TUTORIAL JAVA 3.7

MARISA W. PARYASTO - 33207002

Date: 13/10/2009.

Daftar Isi

Bagian 1. **Dasar-dasar Java**

1. Kelas dan Objek

1.1. **Tujuan.** Setelah menyelesaikan bagian ini, mahasiswa diharapkan dapat:

(1) Memahami cara mengkompilasi dan menjalankan program dengan menggunakan Java
(2) Mengindentifikasi pesan kesalahan yang terjadi pada saat proses kompilasi dilakukan
(3) Memahami konsep dasar dan penggunaan kelas dalam bahasa pemrograman Java
(4) Memahami konsep objek dan merepresentasikannya dalam bahasa pemrograman
(5) Memahami konsep encapsulation dan inheritance

1.2. **Perangkat.** PC dengan JDK 1.5

1.3. **Materi.**

1.3.1. *Program 1 - Hello World.* Ketiklah program dibawah ini lalu compile dengan menggunakan perintah:

```
javac Hello.java
```

dan jalankan dengan perintah

```
java Hello
```

yang diketik dari console.

```
1 class Hello  {
2     public static void main(String[] args)
3     {
4         System.out.println("Hello World!");
5     }
6 }
```

(1) Jelaskan untuk apakah perintah static, void dan args
(2) Apakah yang terjadi jika nama kelas tidak sama dengan nama file-nya? Jelaskan mengapa demikian.

1.3.2. *Program 2 - Kelas sederhana.* Contoh program sederhana yang sudah menggunakan kelas:

```
1  class Student{
2      private String name;
3      String status;
4      int mark;
5
6      public Student(){
7          name = "";
8          status = "";
9          mark = 0;
10     }
```

```
11
12      public void set_name(String n){
13      name = n;
14      }
15
16      public void set_status(String s){
17          status = s;
18      }
19
20      public void set_mark(int m){
21          mark = m;
22      }
23
24      public void display_data(){
25          System.out.println("Name : " + name);
26          System.out.println("Status : " + status);
27          System.out.println("Mark : " + mark);
28  }
29      public static void main(String args[]){
30          Student me = new Student();
31
32          me.set_name("Chika");
33          me.set_status("active");
34          me.set_mark(90);
35          me.display_data();
36      }
37  }
```

Baris 6-10 pada Program 2 disebut constructor. Apakah guna dari constructor ini?

1.3.3. *Program 3 - Kelas sederhana dengan encapsulation.* Berikut adalah modifikasi dari program sebelumnya, dilengkapi dengan contoh enkapsulasi. Pada baris kedua, variabel dari kelas Student di set menjadi private sehingga variabel tersebut tidak dapat diakses oleh kelas lain. Default dari variabel atau kelas jika tidak didefinisikan access specifiernya adalah public.

```
1   class Student{
2       private String name;
3       String status;
4       int mark;
5
6       public Student(){
7           name = "";
8
9           status = "";
10          mark = 0;
```

```
11      }
12
13      public void set_name(String n){
14          name = n;
15      }
16
17      public void set_data(String s){
18          status = s;
19      }
20
21      public void set_mark(int m){
22          mark = m;
23      }
24
25      public void display_status(){
26          System.out.println("Name : " + name);
27          System.out.println("Status : " + status);
28          System.out.println("Mark : " + mark);      }
29      };
30
31      class TestStudent{
32          public static void main(String args[]){
33          Student me = new Student();
34
35          me.set_name("Chika");
36          me.set_status("active");
37          me.set_mark(90);
38
39          me.name = "ABC";
40          me.status = "zzz";
41
42          me.display_data();      }
43  }
```

Perhatikan baris 39 dan 40. Ketika kompilasi terdapat pesan error untuk baris 39 dan tidak untuk baris 40. Mengapa?

1.3.4. *Program 4 - Inheritance sederhana.* Berikut adalah contoh program inheritance sederhana:

```
1  import java.lang.String;
2
3  class Dad{
4          protected String EyesColor;
5          protected String NoseShape;
6          String Character;
7
```

```
8           Dad(){
9                   EyesColor = "Dark Brown";
10                  NoseShape = "Sharp";
11                  Character = "Nice";
12          }
13
14          void Display(){
15                  System.out.println("Eyes color: " +
                        EyesColor);
16                  System.out.println("Nose shape: " +
                        NoseShape);
17  //      System.out.println("Character : " + Character);
18          }
19  };
20
21  class Me extends Dad{
22          String EyesColor;
23          String NoseShape;
24          String Character;
25
26          void SetCharacter(String c){
27                  Character = c;
28          }
29
30          void Set(String Eyes, String Nose){
31                  EyesColor = Eyes;
32                  NoseShape = Nose;
33          }
34
35          void Display(){
36                  super.Display();
37                  System.out.println("Character : " +
                        Character);
38          }
39  }
40
41  class OtherPeople{ }
42
43  public class DadAndMe{
44          public static void main(String args[]){
45                  Dad myDad = new Dad();
46                  Me I = new Me();
47          OtherPeople You = new OtherPeople();
48
49  //      myDad.Display();
```

```
50                  I.SetCharacter("Shy");
51                  myDad.Character = "Grumpy";
52                  I.Display();
53          }
54  }
```

Coba lengkapi class OtherPeople dan perhatikan bahwa atribut yang dimiliki class Dad tidak terdapat pada class OtherPeople

1.3.5. *Program 5 - Inheritance.* Perhatikan program dibawah ini. Kelas Item merupakan turunan dari kelas Book.

```
1  class Item {
2      String item_name;
3      int item_price; //in USD
4
5      void displayDetails()
6      {
7          System.out.println("Item Name is:"+item_name);
8          System.out.println("Item Price is:"+item_price);
9      }
10 }
11
12 public class Book extends Item {
13     String author_name;
14     float seriesno;
15     public Book(){
16         item_name="Macbeth";
17         item_price=10;
18         author_name="Shakespeare";
19         seriesno=1.0f;
20     }
21     void displayDetails(){
22         System.out.println("Author Name is:"+author_name);
23         System.out.println("Series Number is:"+seriesno);
24     }
25
26     public static void main(String args[]){
27         Book bookobj=new Book();
28         bookobj.displayDetails();
29     }
30 }
```

(1) Apakah hasil keluaran dari program tersebut?
(2) Tambahkan satu baris pada program di atas sehingga class Book dapat menampilkan pula atribut item_name dan item_price dengan memanggil method yang dimiliki oleh class Item.

1.3.6. *Program 6 - Inheritance dengan keyword Super.* Berikut adalah contoh penggunaan keyword Super pada inheritance:

```
1  public class Superclass {
2     public void printMethod() {
3     System.out.println("Printed in Superclass.");
4     }
5  }
6
7  public class Subclass extends Superclass {
8     public void printMethod() { //overrides printMethod in
          Superclass
9     super.printMethod();
10    System.out.println("Printed in Subclass");
11    }
12
13    public static void main(String[] args) {
14       Subclass s = new Subclass();
15       s.printMethod();
16    }
17 }
```

1.3.7. *Program 7 - Inheritance dengan lebih banyak kelas.* Hal apa yang berbeda dari listing program dibawah ini dibandingkan dengan program-program sebelumnya?

```
1  class Toy {
2     int toyId;
3     String toyName;
4     float toyPrice;
5
6     public Toy(int id,String name, float price){
7        toyId=id;
8        toyName=name;
9        toyPrice=price;
10    }
11
12       public void displayDetails(){
13          System.out.println("Toy Id is:"+toyId);
14          System.out.println("Toy Name is:"+toyName);
15          System.out.println("Toy Price is:"+toyPrice);
16       }
17 }
18
19 class Customer {
20    int custId;
21    String custName;
```

```
22      String custAddress;
23
24      public Customer(int id,String name,String address){
25          custId=id;
26          custName=name;
27          custAddress=address;
28      }
29
30      public void displayDetails(){
31          System.out.println("Customer Id is:"+custId);
32          System.out.println("Customer Name is:"+custName);
33          System.out.println("Customer Address is:"+custAddress);
34      }
35  }
36
37  class OnlineCustomer extends Customer {
38      String loginId;
39      String masterCardNo;
40
41      public OnlineCustomer(int cId, String name, String address,
42          String id, String cardno){
43          super(cId,name,address);
44          loginId=id;
45          masterCardNo=cardno;
46      }
47
48      public void displayDetails(){
49          super.displayDetails();
50          System.out.println("Customer login id is:"+loginId);
51          System.out.println("Master Card No is:"+masterCardNo);
52      }
53  }
54
55  public class Trial{
56      public static void main(String args[]){
57          OnlineCustomer cObj=new OnlineCustomer(1001,"Carol","164,
58  ------Redmond Way,Ohio","carol@usa.net","9473884833");
59          Customer custObj = new Customer(001,"Andrew","Wallstreet");
60          Toy tObj=new Toy(1001,"Barbie Doll",40);
61          cObj.displayDetails();
62          tObj.displayDetails();
63          custObj.displayDetails();
64      }
65  }
```

1.3.8. *Program 8 - Inheritance overriding.* Pada contoh program ini ditunjukkan bagaimana cara meng-override suatu kelas:

```
1  public class Animal {
2     public static void testClassMethod() {
3        System.out.println("The class method in Animal.");
4     }
5
6     public void testInstanceMethod() {
7        System.out.println("The instance method in Animal.");
8     }
9  }
10
11 public class Cat extends Animal {
12    public static void testClassMethod() {
13       System.out.println("The class method in Cat.");
14    }
15
16    public void testInstanceMethod() {
17       System.out.println("The instance method in Cat.");
18    }
19
20    public static void main(String[] args) {
21       Cat myCat = new Cat();
22       Animal myAnimal = myCat;
23       Animal.testClassMethod();
24       myAnimal.testInstanceMethod();
25    }
26 }
```

2. Struktur dari Program Java

2.1. **Tujuan.** Setelah menyelesaikan bagian ini, mahasiswa diharapkan dapat:

(1) Memahami struktur program Java
(2) Memodifikasi dan membuat program Java dengan struktur yang baik
(3) Mengidentifikasi kesalahan pada program
(4) Membuat program dengan dokumentasi yang baik
(5) Membuat program applet dan aplikasi

2.2. **Perangkat.** PC dengan JDK 1.5

2.3. **Materi.**

2.3.1. *Program 1.* Contoh program aplikasi sederhana

```
1  public class SimpleApplicationCG{
2  public static void main(String args[]){
3     // Display Hello World! now
4     System.out.println("Hello, this is my first application
          !");
5     }
6  }
```

Contoh program applet sederhana

```
1  import java.awt.*;
2  import java.applet.*;
3  public class SimpleAppletCG extends Applet {
4     public void init(){  // Initialize the canvas
5        Color lightgray=new Color(211,211,211); // customize
6        setBackground(lightgray);  // make mellow background
7        resize(150,10);
8     }
9
10    public void paint(Graphics g){  // Display Hello World!
11       g.drawString("Hello, this is my first applet!",50,25);
12    }
13 }
```

Program .html untuk menjalankan applet

```
1  <HTML>
2     <APPLET CODE=SimpleAppletCG.class WIDTH=100 HEIGHT=100>
3     </APPLET>
4  </HTML>
```

Contoh program applet untuk menampilkan tombol

```
1  import java.applet.*;
2  import java.awt.*;
3  public class SimpleApplet extends Applet{
4     String s = "I'm learning Java";
5     int x = 20;
```

```
6     int y = 50;
7
8     public void init(){
9        add(new Button("Hello"));
10    }
11    public void paint(Graphics g){
12       g.drawString(s, x, y);
13       g.drawLine(x, y+2, x+getFontMetrics(getFont()).
             stringWidth(s), y+2);
14    }
15 }
```

GAMBAR 2.1. SimpleApplet.java

(1) Buatlah program aplikasi yang memberikan hasil yang sama dengan program applet di atas

2.3.2. *Program 2.* Berikut adalah contoh program sederhana untuk membuat sebuah jendela dengan dua buah tombol

```
1  import javax.swing.*;
2
3  public class SampleProgram {
4     static JFrame f1;
5     JPanel p1;
6     JButton b1,b2;
7
8     public SampleProgram(){
9        p1=new JPanel();
10       f1.getContentPane().add(p1);
11       b1=new JButton("Submit");
12       p1.add(b1);
13       b2=new JButton("Cancel");
14       p1.add(b2);
15    }
16
17    public static void main(String args[])
```

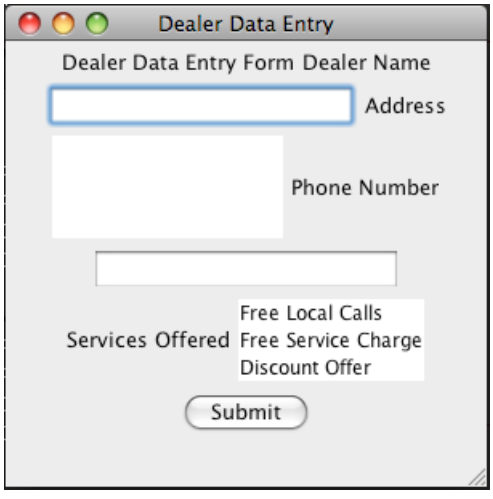
```
18      {
19          f1=new JFrame("Sample Application");
20          SampleProgram obj=new SampleProgram();
21          f1.setSize(300,300);
22          f1.setVisible(true);
23      }
24  }
```

(1) Apa yang terjadi jika keyword "static" pada baris 4 dihilangkan. Beri penjelasan.
(2) Berikan penjelasan mengenai apa yang dilakukan oleh baris ke-10

2.3.3. *Program 3.* Contoh berikut terdiri dari lebih banyak komponen. Perhatikan bagaimana penamaan variabel dan susunan program sehingga mudah untuk ditelusuri.

```
1  import javax.swing.*;
2  import java.awt.*;
3
4  public class Dealer{
5      static JFrame frameObject;
6      JPanel panelObject;
7      JLabel labelTitle;
8      JLabel labelDealerName;
9      JTextField textDealerName;
10     JLabel labelDealerAddress;
11     JTextArea textDealerAddress;
12     JLabel labelDealerPhone;
13     JTextField textDealerPhone;
14     JLabel labelDealerServices;
15     JList listDealerServices;
16     JButton b1;
17
18     public Dealer(){
19         panelObject = new JPanel();
20         frameObject.getContentPane().add(panelObject);
21         //Initialize label controls
22         labelTitle = new JLabel("Dealer Data Entry Form");
23         labelDealerName = new JLabel("Dealer Name");
24         labelDealerAddress = new JLabel("Address");
25         labelDealerPhone = new JLabel("Phone Number");
26         labelDealerServices = new JLabel("Services Offered");
27         //Initialize data entry controls
28         textDealerName = new JTextField(15);
29         textDealerAddress = new JTextArea(4,12);
30         textDealerAddress.setLineWrap(true);
31         textDealerPhone = new JTextField(15);
```

```
32
33          String services[] = {"Free Local Calls",
34          "Free Service Charge", "Discount Offer"};
35          listDealerServices = new JList(services);
36          b1=new JButton("Submit");
37
38          //Add the controls
39          panelObject.add(labelTitle);
40          panelObject.add(labelDealerName);
41          panelObject.add(textDealerName);
42          panelObject.add(labelDealerAddress);
43          panelObject.add(textDealerAddress);
44          panelObject.add(labelDealerPhone);
45          panelObject.add(textDealerPhone);
46          panelObject.add(labelDealerServices);
47          panelObject.add(listDealerServices);
48          panelObject.add(b1);
49      }
50
51      public static void main(String args[]){
52          frameObject=new JFrame("Dealer Data Entry");
53          Dealer dealerObj=new Dealer();
54          frameObject.setSize(300,300);
55          frameObject.setVisible(true);
56      }
57  }
```



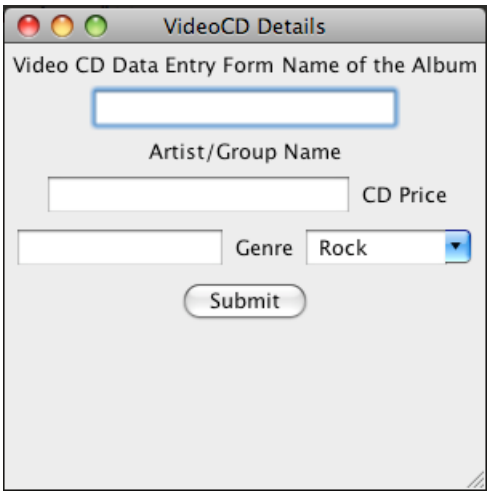
GAMBAR 2.2. Dealer.java

(1) Jelaskan apa yang terjadi jika baris 68 dan 69 dimodifikasi atau dihilangkan? Kesimpulan apa yang dapat diambil? [to be edited]
(2) Adakah perintah lain yang memberikan efek yang sama dengan baris 20? Jika ya, sebutkan.

2.3.4. *Program 4.* Contoh program dengan penggunaan setToolTipText

```
1  import javax.swing.*;
2  import java.awt.*;
3
4  public class VideoCD{
5      static JFrame frameObject;
6      JPanel panelObject;
7      JLabel labelTitle;
8      JLabel labelAlbumName;
9      JLabel labelArtistName;
10     JLabel labelGenre;
11     JLabel labelPrice;
12
13     //variables for data entry controls
14     JTextField textAlbumName;
15     JTextField textArtistName;
16     JComboBox comboGenre;
17     JTextField textPrice;
18     JButton b1;
19
20     public VideoCD(){
21         panelObject = new JPanel();
22         frameObject.getContentPane().add(panelObject);
23
24         //Initialize label controls
25         labelTitle=new JLabel("Video␣CD␣Data␣Entry␣Form");
26         labelAlbumName = new JLabel("Name␣of␣the␣Album");
27         labelArtistName = new JLabel("Artist/Group␣Name");
28         labelGenre = new JLabel("Genre");
29         labelPrice = new JLabel("CD␣Price");
30
31         //Initialize data entry controls
32         textAlbumName = new JTextField(15);
33         textAlbumName.setToolTipText(new String("Enter␣the␣
               name␣of␣the␣album␣here"));
34         textArtistName = new JTextField(15);
35         textArtistName.setToolTipText(new String("Enter␣the␣
               name␣of␣the␣artist/group␣name␣here"));
36         textPrice = new JTextField(10);
```

```
37          textPrice.setToolTipText(new String("Enter␣the␣price␣
                of␣the␣CD␣here"));
38          comboGenre=new JComboBox();
39          comboGenre.setEditable(true);
40          comboGenre.addItem(new String("␣Rock"));
41          comboGenre.addItem(new String("␣Jazz"));
42          comboGenre.addItem(new String("␣Country"));
43          comboGenre.addItem(new String("␣Blues"));
44          comboGenre.addItem(new String("␣Classic"));
45          comboGenre.addItem(new String("␣Alternative"));
46          comboGenre.setToolTipText(new String
47          ("Specify␣the␣target␣age␣group␣and␣the␣gender␣here"));
48          b1=new JButton("Submit");
49
50          //Add the controls
51          panelObject.add(labelTitle);
52          panelObject.add(labelAlbumName);
53          panelObject.add(textAlbumName);
54          panelObject.add(labelArtistName);
55          panelObject.add(textArtistName);
56          panelObject.add(labelPrice);
57          panelObject.add(textPrice);
58          panelObject.add(labelGenre);
59          panelObject.add(comboGenre);
60          panelObject.add(b1);
61      }
62
63      public static void main(String args[])  {
64          frameObject=new JFrame("VideoCD␣Details");
65          VideoCD cdObject=new VideoCD();
66          frameObject.setSize(300,300);
67          frameObject.setVisible(true);
68      }
69  }
```



GAMBAR 2.3. VideoCD

(1) Ubahlah program di atas menjadi program yang menggunakan applet

3. Layout Manager

3.1. **Tujuan.** Setelah menyelesaikan bagian ini, mahasiswa diharapkan dapat:

(1) Menjelaskan perbedaan antara jenis-jenis layout manager yang dimiliki Java
(2) Menggunakan layout manager sesuai kebutuhan

3.2. **Perangkat.** PC dengan JDK 1.5

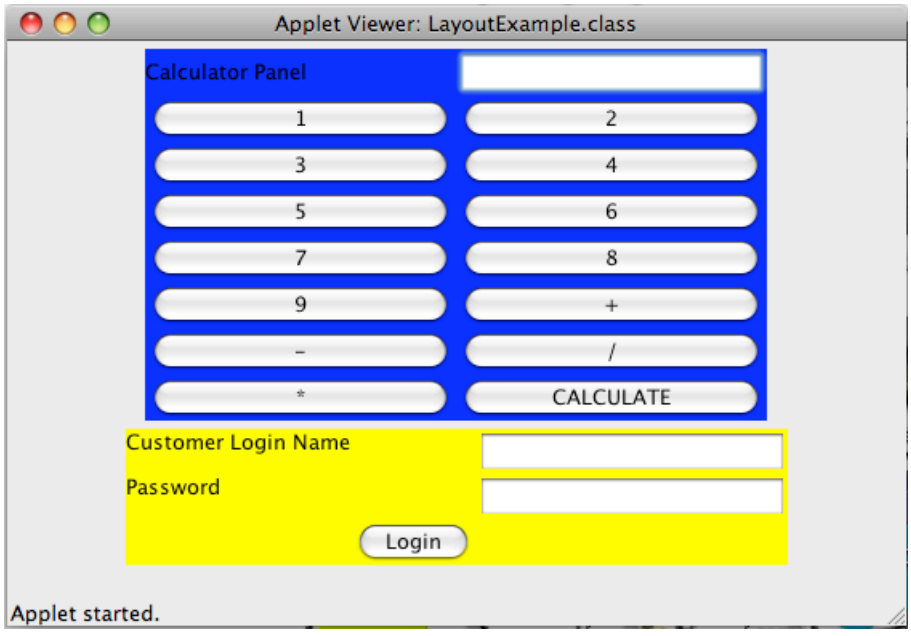
3.3. **Materi.**

3.3.1. *Contoh Layout.* Perhatikan contoh program dibawah ini:

```
1  import javax.swing.*;
2  import java.awt.*;
3  public class LayoutExample extends JApplet {
4     JPanel p1,p3;    JPanel panelObject;
5     JLabel labelCustName;
6     JLabel labelCustPassword;
7     JTextField textCustName;
8     JPasswordField textCustPassword;
9     JButton buttonLogin;
10    GridBagLayout gl;
11    GridBagConstraints gbc;
12    JLabel l1,l2;
13    JTextField tf1;
14    JButton b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14;
15    GridLayout g1;
16    BoxLayout bl1;
17
18    public void init(){
19       g1=new GridLayout(8,2);
20       JPanel p1=new JPanel();
21       p1.setLayout(g1);
22       l2=new JLabel("Calculator Panel");
23       tf1=new JTextField(15);
24       b1=new JButton("1");
25       b2=new JButton("2");
26       b3=new JButton("3");
27       b4=new JButton("4");
28       b5=new JButton("5");
29       b6=new JButton("6");
30       b7=new JButton("7");
31       b8=new JButton("8");
32       b9=new JButton("9");
33       b10=new JButton("+");
34       b11=new JButton("-");
35       b12=new JButton("/");
```

```
36          b13=new JButton("*");
37          b14=new JButton("CALCULATE");
38          p1.add(l2);
39          p1.add(tf1);
40          p1.add(b1);
41          p1.add(b2);
42          p1.add(b3);
43          p1.add(b4);
44          p1.add(b5);
45          p1.add(b6);
46          p1.add(b7);
47          p1.add(b8);
48          p1.add(b9);
49          p1.add(b10);
50          p1.add(b11);
51          p1.add(b12);
52          p1.add(b13);
53          p1.add(b14);
54          p1.setBackground(Color.blue);
55          gl = new GridBagLayout();
56          gbc = new GridBagConstraints();
57          panelObject = new JPanel();
58          panelObject.setLayout(gl);
59
60          //Initialize controls
61          labelCustName = new JLabel("Customer␣Login␣Name");
62          labelCustPassword = new JLabel("Password");
63          textCustName = new JTextField(15);
64          textCustPassword = new JPasswordField(15);
65          buttonLogin=new JButton("Login");
66
67          //Add controls to the panel
68          gbc.anchor = GridBagConstraints.NORTHWEST;
69          gbc.gridx = 1;
70          gbc.gridy = 5;
71          gl.setConstraints(labelCustName,gbc);
72          panelObject.add(labelCustName);
73          gbc.anchor = GridBagConstraints.NORTHWEST;
74          gbc.gridx = 4;
75          gbc.gridy = 5;
76          gl.setConstraints(textCustName,gbc);
77          panelObject.add(textCustName);
78          gbc.anchor = GridBagConstraints.NORTHWEST;
79          gbc.gridx = 1;
80          gbc.gridy = 9;
```

```
81          gl.setConstraints(labelCustPassword,gbc);
82          panelObject.add(labelCustPassword);
83          gbc.anchor = GridBagConstraints.NORTHWEST;
84          gbc.gridx = 4;
85          gbc.gridy = 9;
86          gl.setConstraints(textCustPassword,gbc);
87          panelObject.add(textCustPassword);
88          gbc.anchor=GridBagConstraints.NORTHWEST;
89          gbc.gridx=2;
90          gbc.gridy=13;
91          gl.setConstraints(buttonLogin,gbc);
92          panelObject.add(buttonLogin);
93          panelObject.setBackground(Color.yellow);
94          p3=new JPanel();
95          bl1=new BoxLayout(p3,BoxLayout.X_AXIS);
96          bl1.addLayoutComponent("Calculator Panel",p1);
97          bl1.addLayoutComponent("Text Panel",panelObject);
98          getContentPane().add(p3);
99          p3.add(p1);
100         p3.add(panelObject);
101     }
102 }
```



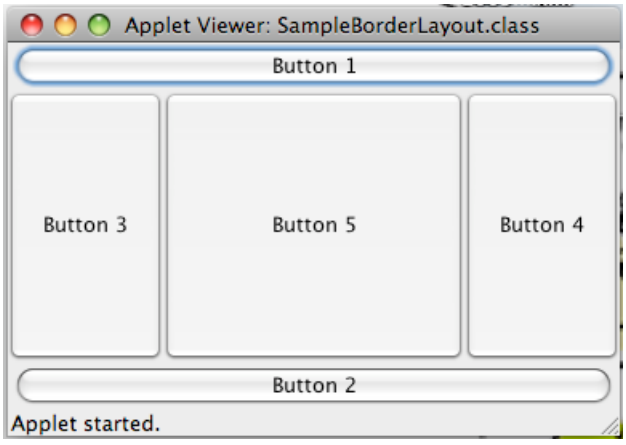
GAMBAR 3.1. Layout Example

(1) Jelaskan perintah pada baris 54 dan 55

(2) Jelaskan mengapa program diatas menggunakan lebih dari satu panel

3.3.2. *Border Layout.* Berikut adalah contoh program dengan menggunakan border layout

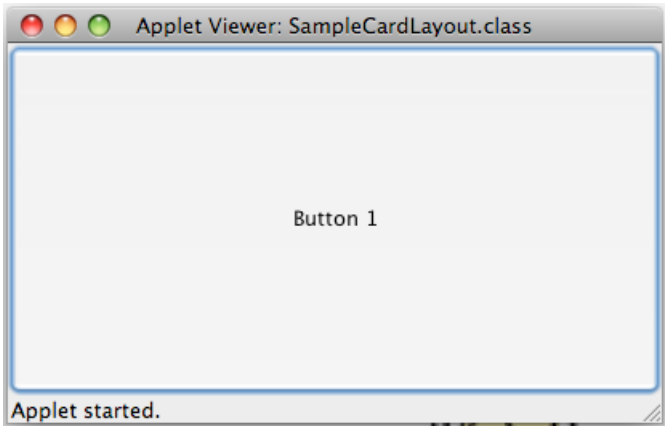
```
1  import java.awt.*;
2  import javax.swing.*;
3
4  public class SampleBorderLayout extends JApplet {
5      JButton b1,b2,b3,b4,b5;
6      BorderLayout bd1;
7
8      public void init() {
9          bd1 = new BorderLayout();
10         JPanel p1 = new JPanel();
11         getContentPane().add(p1);
12         p1.setLayout(bd1);
13         b1 = new JButton("Button 1");
14         b2 = new JButton("Button 2");
15         b3 = new JButton("Button 3");
16         b4 = new JButton("Button 4");
17         b5 = new JButton("Button 5");
18         p1.add("North",b1);
19         p1.add("South",b2);
20         p1.add("West",b3);
21         p1.add("East",b4);
22         p1.add("Center",b5);
23     }
24 }
```



GAMBAR 3.2. Border Layout

3.3.3. *Card Layout.* Berikut adalah contoh program dengan menggunakan card layout

```
1  import javax.swing.*;
2  import java.awt.*;
3
4  public class SampleCardLayout extends JApplet {
5     JButton button1, button2, button3;
6     CardLayout c1;
7     JPanel p1;
8
9     public void init() {
10       c1 = new CardLayout();
11       p1 = new JPanel();
12       p1.setLayout(c1);
13
14       getContentPane().add(p1);
15       button1 = new JButton("Button 1");
16       button2 = new JButton("Button 2");
17       button3 = new JButton("Button 3");
18       p1.add("Button1", button1);
19       p1.add("Button2", button2);
20       p1.add("Button3", button3);
21    }
22 }
```

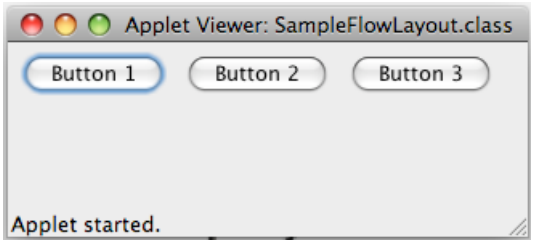


GAMBAR 3.3. Card Layout

3.3.4. *Flow Layout.* Berikut adalah contoh program dengan menggunakan flow layout

```
1  import java.awt.*;
2  import javax.swing.*;
```

```
3
4  public class SampleFlowLayout extends JApplet {
5     JButton b1, b2, b3;
6     FlowLayout f1;
7
8     public void init() {
9        f1 = new FlowLayout(FlowLayout.LEFT);
10       JPanel p1 = new JPanel();
11       getContentPane().add(p1);
12       p1.setLayout(f1);
13       b1 = new JButton("Button 1");
14       b2 = new JButton("Button 2");
15       b3 = new JButton("Button 3");
16       p1.add(b1);
17       p1.add(b2);
18       p1.add(b3);
19    }
20 }
```

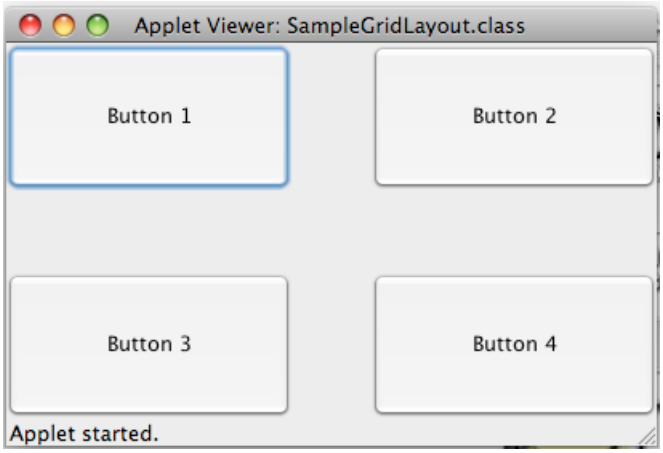


GAMBAR 3.4. Flow Layout

3.3.5. *Grid Layout.* Berikut adalah contoh program dengan menggunakan grid layout

```
1  import java.awt.*;
2  import javax.swing.*;
3  public class SampleGridLayout extends JApplet {
4     JButton b1, b2, b3, b4;
5     GridLayout g1;
6
7     public void init() {
8        g1 = new GridLayout(2,2,50,50);
9        JPanel p1 = new JPanel();
10       getContentPane().add(p1);
11       p1.setLayout(g1);
12       b1 = new JButton("Button 1");
13       b2 = new JButton("Button 2");
```

```
14          b3 = new JButton("Button 3");
15          b4 = new JButton("Button 4");
16          p1.add(b1);
17          p1.add(b2);
18          p1.add(b3);
19          p1.add(b4);
20      }
21  }
```

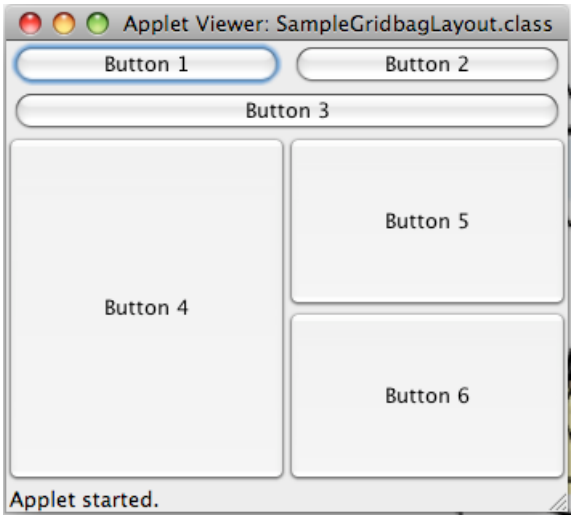


GAMBAR 3.5. Grid Layout

3.3.6. *Gridbag Layout.* Berikut adalah contoh program yang menggunakan gridbag layout

```
1  import java.awt.*;
2  import javax.swing.*;
3  public class GridBagSample extends JApplet {
4      JPanel panelObject;
5      GridBagLayout gbObject;
6      GridBagConstraints gbc;
7      JButton b1,b2,b3,b4,b5,b6;
8
9      public void init() {
10         gbObject = new GridBagLayout();
11         gbc = new GridBagConstraints();
12         panelObject = new JPanel();
13         getContentPane().add(panelObject);
14         panelObject.setLayout(gbObject);
15         JButton b1 = new JButton("Button 1");
16         JButton b2 = new JButton("Button 2");
17         JButton b3 = new JButton("Button 3");
```

```
18          JButton b4 = new JButton("Button 4");
19          JButton b5 = new JButton("Button 5");
20          JButton b6 = new JButton("Button 6");
21          gbc.fill = GridBagConstraints.BOTH;
22          gbc.anchor = GridBagConstraints.CENTER;
23          gbc.gridwidth = 1;
24          gbc.weightx = 1.0;
25          gbObject.setConstraints(b1,gbc);
26          panelObject.add(b1);
27          gbc.gridwidth = GridBagConstraints.REMAINDER;
28          gbObject.setConstraints(b2,gbc);
29          panelObject.add(b2);
30          gbc.gridwidth = GridBagConstraints.REMAINDER;
31          gbObject.setConstraints(b3,gbc);
32          panelObject.add(b3);
33          gbc.weightx = 0.0;
34          gbc.weighty = 1.0;
35          gbc.gridheight = 2;
36          gbc.gridwidth = 1;
37          gbObject.setConstraints(b4,gbc);
38          panelObject.add(b4);
39          gbc.gridwidth = GridBagConstraints.REMAINDER;
40          gbc.gridheight = 1;
41          gbObject.setConstraints(b5,gbc);
42          panelObject.add(b5);
43          gbc.gridwidth = GridBagConstraints.REMAINDER;
44          gbc.gridheight = 1;
45          gbObject.setConstraints(b6,gbc);
46          panelObject.add(b6);
47      }
48  }
```



GAMBAR 3.6. Gridbag Layout

3.3.7. *Box Layout.* Berikut adalah contoh program yang menggunakan box layout

```
1  import java.awt.*;
2  import javax.swing.*;
3
4  public class SampleBoxLayout extends JApplet {
5     JPanel calpanel, logpanel;
6     JPanel mainpanel;
7
8     public void init() {
9        GridLayout gridObj;
10       GridBagLayout gridbagObj;
11       GridBagConstraints gbc;
12       BoxLayout boxObj;
13
14       gridObj = new GridLayout(8,2);
15       calpanel = new JPanel();
16       calpanel.setLayout(gridObj);
17       gridbagObj = new GridBagLayout();
18       gbc = new GridBagConstraints();
19       logpanel = new JPanel();
20       logpanel.setLayout(gridbagObj);
21       mainpanel = new JPanel();
22       boxObj = new BoxLayout(mainpanel,BoxLayout.X_AXIS);
23       boxObj.addLayoutComponent("Calculator Panel", calpanel);
24       boxObj.addLayoutComponent("TextPanel", logpanel);
25       getContentPane().add(mainpanel);
26       mainpanel.add(calpanel);
```

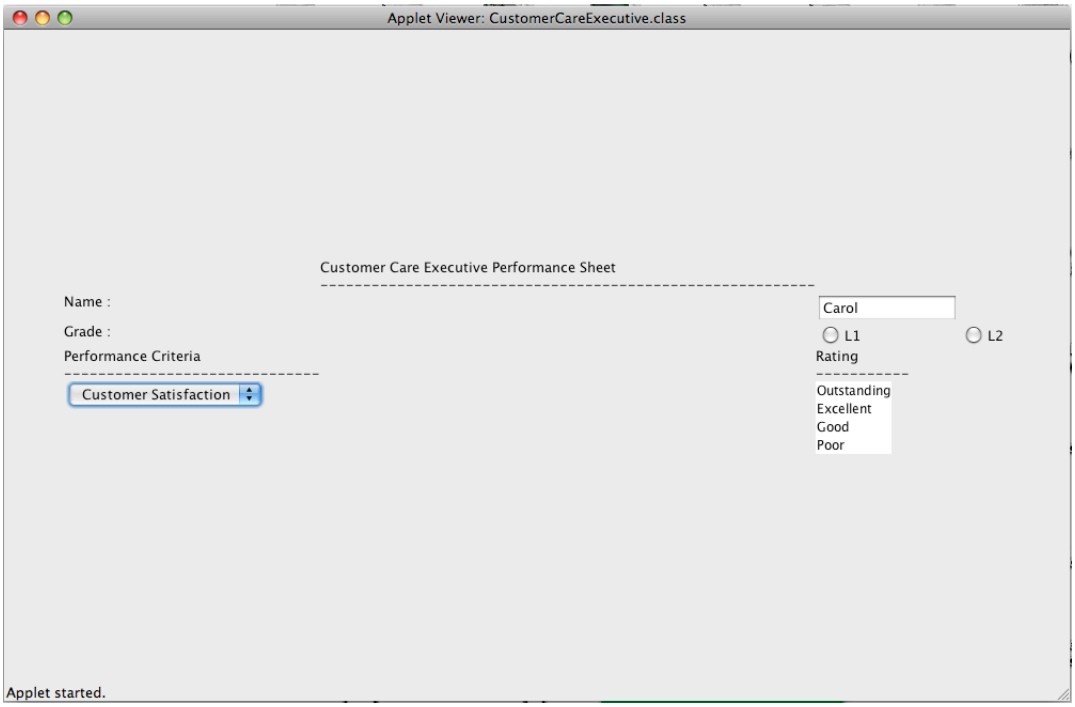
```
27          mainpanel.add(logpanel);
28      }
29  }
```

3.3.8. *More examples.* Berikut adalah contoh program tambahan

```
1   import javax.swing.*;
2   import java.awt.*;
3
4   public class CustomerCareExecutive extends JApplet {
5       //Variables for the controls
6       JPanel panelObject;
7       JLabel labeltitle,labelName,labelGrade,labelPerfCriteria,
            labelRating;
8       JLabel line1,line2, line3;
9       JTextField textName;
10      JRadioButton buttonGrade1,buttonGrade2;
11      JComboBox comboPerfCriteria;
12      JList listRating;
13      GridBagLayout gl;
14      GridBagConstraints gbc;
15
16      public void init() {
17          //initialize the layout
18          gl = new GridBagLayout();
19          gbc = new GridBagConstraints();
20          panelObject = (JPanel)getContentPane();
21          panelObject.setLayout(gl);
22
23          //initialize the controls
24          labeltitle=new JLabel("Customer␣Care␣Executive␣
                Performance␣Sheet");
25          line1=new JLabel("
                ----------------------------------------------------------------
                ");
26          labelName=new JLabel("Name␣:");
27          labelGrade=new JLabel("Grade␣:");
28          labelPerfCriteria=new JLabel("Performance␣Criteria");
29          labelRating=new JLabel("Rating");
30          line2=new JLabel("-----------------------------------");
31          line3=new JLabel("----------");
32          textName=new JTextField(10);
33          textName.setText("Carol");
34          buttonGrade1=new JRadioButton("L1");
35          buttonGrade2=new JRadioButton("L2");
```

```
36          String criteria[] = {"Customer Satisfaction", "
                Productivity"};
37          comboPerfCriteria = new JComboBox(criteria);
38          String rating[]={"Outstanding","Excellent","Good","
                Poor"};
39          listRating=new JList(rating);
40
41          //Apply constraints and add controls to the panel
42          gbc.anchor = GridBagConstraints.NORTHWEST;
43          gbc.gridx = 3;
44          gbc.gridy = 1;
45          gl.setConstraints(labeltitle,gbc);
46          panelObject.add(labeltitle);
47          gbc.anchor = GridBagConstraints.NORTHWEST;
48          gbc.gridx = 3;
49          gbc.gridy = 3;
50          gl.setConstraints(line1,gbc);
51          panelObject.add(line1);
52          gbc.anchor = GridBagConstraints.NORTHWEST;
53          gbc.gridx = 2;
54          gbc.gridy = 9;
55          gl.setConstraints(labelName,gbc);
56          panelObject.add(labelName);
57          gbc.anchor = GridBagConstraints.NORTHWEST;
58          gbc.gridx = 4;
59          gbc.gridy = 9;
60          gl.setConstraints(textName,gbc);
61          panelObject.add(textName);
62          gbc.anchor = GridBagConstraints.NORTHWEST;
63          gbc.gridx = 2;
64          gbc.gridy = 14;
65          gl.setConstraints(labelGrade,gbc);
66          panelObject.add(labelGrade);
67          gbc.anchor = GridBagConstraints.NORTHWEST;
68          gbc.gridx = 4;
69          gbc.gridy = 14;
70          gl.setConstraints(buttonGrade1,gbc);
71          panelObject.add(buttonGrade1);
72          gbc.anchor = GridBagConstraints.NORTHWEST;
73          gbc.gridx = 5;
74          gbc.gridy = 14;
75          gl.setConstraints(buttonGrade2,gbc);
76          panelObject.add(buttonGrade2);
77          gbc.anchor = GridBagConstraints.NORTHWEST;
78          gbc.gridx = 2;
```

```
79          gbc.gridy = 19;
80          gl.setConstraints(labelPerfCriteria,gbc);
81          panelObject.add(labelPerfCriteria);
82          gbc.anchor = GridBagConstraints.NORTHWEST;
83          gbc.gridx = 4;
84          gbc.gridy = 19;
85          gl.setConstraints(labelRating,gbc);
86          panelObject.add(labelRating);
87          gbc.anchor = GridBagConstraints.NORTHWEST;
88          gbc.gridx = 2;
89          gbc.gridy = 22;
90          gl.setConstraints(line2,gbc);
91          panelObject.add(line2);
92          gbc.anchor = GridBagConstraints.NORTHWEST;
93          gbc.gridx = 4;
94          gbc.gridy = 22;
95          gl.setConstraints(line3,gbc);
96          panelObject.add(line3);
97          gbc.anchor = GridBagConstraints.NORTHWEST;
98          gbc.gridx = 2;
99          gbc.gridy = 26;
100         gl.setConstraints(comboPerfCriteria,gbc);
101         panelObject.add(comboPerfCriteria);
102         gbc.anchor = GridBagConstraints.NORTHWEST;
103         gbc.gridx = 4;
104         gbc.gridy = 26;
105         gl.setConstraints(listRating,gbc);
106         panelObject.add(listRating);
107     }
108 }
```



GAMBAR 3.7. More examples

Bagian 2. Pemrograman dalam Java

4. Membuat applet dan application

4.1. **Applet.** Untuk memanggil applet perlu dibuat satu file dalam format html yang berisi file .class yang akan dipanggil, seperti contoh di bawah ini:

```
<HTML>
    <APPLET CODE="DisplayApplet.class" WIDTH="200" HEIGHT="
        200">
    </APPLET>
</HTML>
```

```
import javax.swing.*;
import java.awt.*;

public class DisplayApplet extends JApplet{
    public void paint(Graphics g){
        g.drawString("This is displayed by the paint method"
            ,20,20);
    }
}
```

4.1.2. *Program 2.* Berikut adalah program yang menunjukkan siklus dari suatu applet. Perhatikan counter-counter dari applet tersebut.

```
import javax.swing.*;
import java.awt.*;

public class AppletMethods extends JApplet {
    int initCounter = 0;
    int startCounter = 0;
    int stopCounter = 0;
    int destroyCounter = 0;

    public void init(){
        initCounter++;
        repaint();
    }

    public void start(){
         startCounter++;
         repaint();
    }

    public void stop(){
        stopCounter++;
```

```
22          repaint();
23      }
24
25      public void destroy(){
26          destroyCounter++;
27          repaint();
28      }
29
30      public void paint(Graphics g){
31          g.drawString("init has been invoked "
32          + String.valueOf(initCounter) + " times",20,20);
33          g.drawString("start has been invoked "
34          + String.valueOf(startCounter) + " times",20,35);
35          g.drawString("stop has been invoked "
36          + String.valueOf(stopCounter) + " times",20,50);
37          g.drawString("destroy has been invoked "
38          + String.valueOf(destroyCounter) + " times",20,65);
39      }
40  }
```

```
1   import javax.swing.*;
2
3   public class VCD extends JApplet{
4       //Variable for the panel
5       static JPanel panelObject;
6       int stopCount;
7       int startCount;
8
9       //Variables of labels
10      JLabel    labelVideoCDNo;
11      JLabel    labelVideoCDName;
12      JLabel    labelGenre;
13      JLabel    labelArtist;
14
15      //Label for the image
16      JLabel    labelImagePosition;
17
18      //Label to display count
19      JLabel    labelCount;
20
21      //Variables for data entry controls
22      JTextField    textVideoCDNo;
23      JTextField    textVideoCDName;
24      JComboBox     comboGenre;
25      JTextField    textArtist;
```

```
26
27      public void init(){
28          //  Create panel
29          panelObject = new JPanel();
30          getContentPane().add(panelObject);
31
32          // Initializing labels
33          labelVideoCDNo = new JLabel("Video CD Number");
34          labelVideoCDName = new JLabel(" Name");
35          labelGenre = new JLabel("Package");
36          labelArtist = new JLabel("Artist");
37          labelCount=new JLabel();
38          Icon logoImage = new
39          ImageIcon("c:\\Semester\\Batchcode
40  ......\\Groupname\\CellGO\\Images\\VideoCD.gif");
41          labelImagePosition = new JLabel(logoImage);
42
43          //Initializing TextField
44          textVideoCDNo = new JTextField(15);
45          textVideoCDName = new JTextField(30);
46          textArtist = new JTextField(30);
47          String genres[] = { "Rock", "Pop","Classical","Rap"};
48          comboGenre = new JComboBox(genres);
49
50          //Adding  image to the applet
51          panelObject.add(labelImagePosition);
52
53          //Adding controls for Video CD No
54          panelObject.add(labelVideoCDNo);
55          panelObject.add(textVideoCDNo);
56
57          //Adding controls for Video CD Name
58          panelObject.add(labelVideoCDName);
59          panelObject.add(textVideoCDName);
60
61          //Adding controls for genre
62          panelObject.add(labelGenre);
63          panelObject.add(comboGenre);
64
65          //Adding controls for Customer Age
66          panelObject.add(labelArtist);
67          panelObject.add(textArtist);
68
69          //Adding the label for count
70          panelObject.add(labelCount);
```

```
71      }
72
73      public void start(){
74          //Increment startCount by 1
75          startCount++;
76
77          //Show the updated count
78          String count="Start count is "+startCount
79          +" Stop count is "+stopCount;
80          labelCount.setText(count);
81      }
82
83      public void stop(){
84          //Increment stopCount by 1
85          stopCount++;
86          String count="Stop count is "+stopCount
87          +" Start count is "+startCount;
88          labelCount.setText(count);
89      }
90  }
```

```
1   import java.awt.*;
2   import javax.swing.*;
3
4   public class DailyDiary extends JApplet{
5       JPanel panelObj;
6       JLabel labelTask;
7       JButton buttonStore,buttonDisplay;
8       JComboBox comboTaskList;
9       JLabel labelStatus;
10
11      public void init(){
12          panelObj=new JPanel();
13          getContentPane().add(panelObj);
14
15          Font myfont = new Font("Times New Roman", Font.BOLD,
                18);
16          labelTask =new JLabel("Task List:");
17
18          String taskMessage[]={"9.00 AM Meeting on SEI - CMM",
                "1.00 PM Submit bug report for project to the
                Manager","Book cakes for the Birthday Party", "
                Book tickets for New York flight"};
19
```

```
20          comboTaskList =new JComboBox(taskMessage);
21          comboTaskList.setEditable(true);
22          labelStatus=new JLabel();
23          buttonStore =new JButton("Store Schedule");
24          buttonDisplay =new JButton("Display Status");
25
26          panelObj.add(labelTask);
27          panelObj.add(comboTaskList);
28          panelObj.add(buttonStore);
29          panelObj.add(buttonDisplay);
30          panelObj.add(labelStatus);
31      }
32
33      public void start(){
34          getAppletContext().showStatus("Hi!!! You'll do it");
35      }
36
37      public void stop(){
38          getAppletContext().showStatus("Will be back again to
                remind you");
39          labelStatus.setText("Unknown");
40      }
41
42      public void paint(Graphics g){
43          labelStatus.setText(String.valueOf(comboTaskList.
                getItemCount()));
44      }
45  }
```

```
1   import javax.swing.*;
2
3   public class Dealer extends JApplet{
4       //Variable for the panel
5       JPanel  panelObject;
6
7       //Variables of labels
8       JLabel    labelDealerCellNo;
9       JLabel    labelDealerName;
10      JLabel    labelDealerAddress;
11      JLabel    labelDealerScheme;
12
13      //Variables for data entry controls
14      JTextField    textDealerCellNo;
15      JTextField    textDealerName;
```

```
16      JComboBox     comboDealerScheme;
17      JTextField    textDealerAddress;
18
19      public void init(){
20         // Add appropriate controls to the frame
21         //  Create panel
22         panelObject = new JPanel();
23         getContentPane().add(panelObject);
24
25         //Create and add the appropriate controls
26         // Initializing labels
27         labelDealerCellNo = new JLabel("Cell Number");
28         labelDealerName = new JLabel(" Name");
29         labelDealerScheme = new JLabel("Scheme");
30         labelDealerAddress = new JLabel("Address");
31
32         //Initializing textfield
33         textDealerCellNo = new JTextField(15);
34         textDealerName = new JTextField(30);
35         textDealerAddress = new JTextField(30);
36         String schemes[] = { "Discount", "Standard"};
37         comboDealerScheme = new JComboBox(schemes);
38
39         //Adding controls for cell Number
40         panelObject.add(labelDealerCellNo);
41         panelObject.add(textDealerCellNo);
42
43         //Adding controls for Dealer Name
44         panelObject.add(labelDealerName);
45         panelObject.add(textDealerName);
46
47         //Adding controls for Dealer Package
48         panelObject.add(labelDealerScheme);
49         panelObject.add(comboDealerScheme);
50
51         //Adding controls for Dealer Address
52         panelObject.add(labelDealerAddress);
53         panelObject.add(textDealerAddress);
54      }
55
56      public void destroy(){
57         showFrame();
58      }
59
60      public void showFrame(){
```

```
61          JFrame frame = new JFrame("Dealer Details");
62
63          //Variables of labels
64          JLabel    labelDealCellNo;
65          JLabel    labelDealName;
66          JLabel    labelDealAddress;
67          JLabel    labelDealScheme;
68
69          //Variables for data entry controls
70          JTextField    textDealCellNo=new JTextField(10);
71          JTextField    textDealName=new JTextField(15);
72          JTextField    textDealScheme=new JTextField(15);
73          JTextField    textDealAddress=new JTextField(25);
74          textDealCellNo.setText(textDealerCellNo.getText());
75          textDealName.setText(textDealerName.getText());
76          textDealAddress.setText(textDealerAddress.getText());
77          textDealScheme.setText(String.valueOf
78          (comboDealerScheme.getSelectedItem()));
79          labelDealCellNo = new JLabel("Cell Number");
80          labelDealName = new JLabel(" Name");
81          labelDealScheme = new JLabel("Scheme");
82          labelDealAddress = new JLabel("Address");
83          JPanel panel=new JPanel();
84
85          //add panel to the frame
86          frame.getContentPane().add(panel);
87
88          //Adding controls for cell Number
89          panel.add(labelDealCellNo);
90          panel.add(textDealCellNo);
91
92          //Adding controls for Dealer Name
93          panel.add(labelDealName);
94          panel.add(textDealName);
95
96          //Adding controls for Dealer Package
97          panel.add(labelDealScheme);
98          panel.add(textDealScheme);
99
100         //Adding controls for Dealer Address
101         panel.add(labelDealAddress);
102         panel.add(textDealAddress);
103         frame.setSize(200,200);
104         frame.setVisible(true);
105     }
```

```
106 }
```

```
 1 import javax.swing.*;
 2
 3 public class SampleProgram{
 4    static JFrame f1;
 5    JPanel p1;
 6    JButton b1;
 7
 8    public SampleProgram(){
 9       p1 = new JPanel();
10       f1.getContentPane().add(p1);
11       b1 = new JButton("Submit");
12       p1.add(b1);
13       b1.setText("Cancel");
14       p1.add(b1);
15    }
16
17    public static void main(String args[]){
18       f1 = new JFrame("Sample Application");
19       SampleProgram sp = new SampleProgram();
20       f1.setSize(300,300);
21       f1.setVisible(true);
22    }
23 }
```

```
 1 import javax.swing.*;
 2
 3 public class NewJFrame extends JFrame  {
 4    JButton jButton1, jButton2, jButton3,
 5       jButton4, jButton5, jButton6;
 6
 7    public NewJFrame() {
 8       java.awt.GridBagConstraints gridBagConstraints;
 9       jButton1 = new JButton("1");
10       jButton2 = new JButton("2");
11       jButton3 = new JButton("3");
12       jButton4 = new JButton("4");
13       jButton5 = new JButton("5");
14       jButton6 = new JButton("6");
15
16       getContentPane().setLayout(new java.awt.GridBagLayout
             ());
17       setDefaultCloseOperation(EXIT_ON_CLOSE);
```

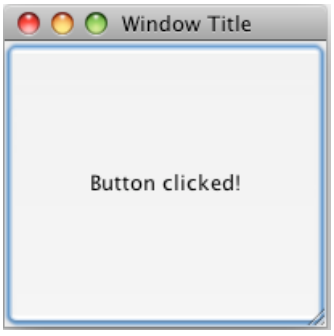
```
18          gridBagConstraints = new java.awt.GridBagConstraints()
                ;
19          gridBagConstraints.gridx = 1;
20          gridBagConstraints.gridy = 1;
21          gridBagConstraints.weightx = 0.5;
22          gridBagConstraints.weighty = 0.5;
23          getContentPane().add(jButton1, gridBagConstraints);
24          gridBagConstraints = new java.awt.GridBagConstraints()
                ;
25          gridBagConstraints.gridx = 3;
26          gridBagConstraints.gridy = 0;
27          gridBagConstraints.gridheight = 3;
28          gridBagConstraints.fill = java.awt.GridBagConstraints.
                VERTICAL;
29          getContentPane().add(jButton2, gridBagConstraints);
30          gridBagConstraints = new java.awt.GridBagConstraints()
                ;
31          gridBagConstraints.gridx = 4;
32          gridBagConstraints.gridy = 0;
33          gridBagConstraints.gridheight = 3;
34          gridBagConstraints.fill = java.awt.GridBagConstraints.
                VERTICAL;
35          getContentPane().add(jButton3, gridBagConstraints);
36          gridBagConstraints = new java.awt.GridBagConstraints()
                ;
37          gridBagConstraints.gridx = 0;
38          gridBagConstraints.gridy = 3;
39          gridBagConstraints.gridwidth = 3;
40          gridBagConstraints.fill = java.awt.GridBagConstraints.
                HORIZONTAL;
41          getContentPane().add(jButton4, gridBagConstraints);
42          gridBagConstraints = new java.awt.GridBagConstraints()
                ;
43          gridBagConstraints.gridx = 3;
44          gridBagConstraints.gridy = 3;
45          gridBagConstraints.fill = java.awt.GridBagConstraints.
                VERTICAL;
46          getContentPane().add(jButton5, gridBagConstraints);
47          gridBagConstraints = new java.awt.GridBagConstraints()
                ;
48          gridBagConstraints.gridx = 4;
49          gridBagConstraints.gridy = 3;
50          getContentPane().add(jButton6, gridBagConstraints);
51          //          pack();
52          setSize(800,600);
```

```
53      }
54      public static void main(String args[]){
55          new NewJFrame().setVisible(true);
56      }
57  }
```

5. Event Handling

5.1. **Program 1.** Contoh program aplikasi untuk membuat satu tombol yang dapat di-klik dengan mengimplementasikan ActionListener.

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  class MyFrame extends JFrame{
6      JButton b1;
7
8      public static void main(String args[]){
9          MyFrame f = new MyFrame();
10     }
11
12     public MyFrame(){
13         super("Window Title");
14         b1 = new JButton("Click Me!");
15
16         getContentPane().add("Center", b1);
17         ButtonListener bListen = new ButtonListener();
18         b1.addActionListener(bListen);
19         setSize(200,200);
20         setVisible(true);
21     }
22
23     class ButtonListener implements ActionListener{
24         public void actionPerformed(ActionEvent evt){
25             JButton source = (JButton)evt.getSource();
26             source.setText("Button clicked!");
27         }
28     }
29 }
```



Gambar 5.1. Button Clicked

5.2. **Program 2.** Contoh applet untuk membuat satu buah tombol yang mengimplementasikan ActionListener. Perhatikan cara mendeklarasikan ActionListener dan menghubungkannya pada tombol dibandingkan dengan Program 1

```
1  import java.awt.event.*;
2  import javax.swing.*;
3
4  public class Welcome extends JApplet implements
       ActionListener{
5      JPanel p1;
6      JButton buttonClickMe;
7
8      public void init(){
9          p1=new JPanel();
10         getContentPane().add(p1);
11         buttonClickMe=new JButton("Click␣Me");
12         p1.add(buttonClickMe);
13         buttonClickMe.addActionListener(this);
14     }
15
16     public void actionPerformed(ActionEvent eventObject){
17         getAppletContext().showStatus("You␣clicked␣the␣Click␣
               Me␣button.");
18     }
19 }
```

GAMBAR 5.2. ButtonAppletBefore

5.3. **Program 3.** Contoh applet dengan satu button yang jika ditekan akan menampilkan angka berurut yang dihasilkan oleh suatu counter. Perhatikan variabel numClicks yang dideklarasikan static. Apakah guna dari variabel static?

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class MyButtonApplet extends JApplet implements ActionListener{
```

GAMBAR 5.3. ButtonAppletAfter

```
6     JPanel panel;
7     JButton button;
8     static int numClicks = 0;
9
10    public void init(){
11       panel = new JPanel();
12       button = new JButton("Click!");
13
14       getContentPane().add(panel);
15       panel.add(button);
16       button.addActionListener(this);
17    }
18
19    public void actionPerformed(ActionEvent e){
20       Object obj = e.getSource();
21       if (obj == button){
22          numClicks++;
23          getAppletContext().showStatus("Number of clicks : " + numClicks);
24       }
25    }
26 }
```

GAMBAR 5.4. Counter Applet

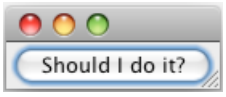
5.4. **Program 4.** Contoh program aplikasi untuk membuat satu button yang bisa di-klik dengan variabel static yang menjadi counter.

```
1  import java.awt.*;
2  import java.awt.event.*;
3
4  public class TestButton implements ActionListener{
5      Frame f;
6      Button b;
7      static int numClick = 0;
8
9      public TestButton(){
10         f = new Frame("Test");
11         b = new Button("Press␣Me!");
12         b.setActionCommand("ButtonPressed");
13     }
14
15     public void LaunchFrame(){
16         b.addActionListener(this);
17         f.add(b,BorderLayout.CENTER);
18         f.pack();
19         f.setVisible(true);
20     }
21
22     public void actionPerformed(ActionEvent e){
23         numClick++;
24         System.out.println("Action␣occured");
25         System.out.println("Button's␣command␣is␣:␣" + e.
               getActionCommand());
26         System.out.println("Number␣of␣button␣clicks␣:␣" +
               numClick);
27     }
28
29     public static void main(String args[]){
30         TestButton tb = new TestButton();
31         tb.LaunchFrame();
32     }
33 }
```

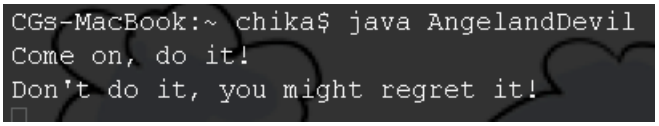
5.5. **Program 5.** Contoh program aplikasi dengan dua buah tombol dengan dua ActionListener

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class AngelandDevil{
```

```
6     JFrame frame;
7
8     public static void main(String args[]){
9        AngelandDevil aad = new AngelandDevil();
10       aad.go();
11    }
12
13    public void go(){
14       frame = new JFrame();
15       JButton button = new JButton("Should I do it?");
16       button.addActionListener(new AngelListener());
17       button.addActionListener(new DevilListener());
18       frame.getContentPane().add(BorderLayout.CENTER, button
             );
19       frame.pack();
20       frame.setVisible(true);
21    }
22
23    class AngelListener implements ActionListener{
24       public void actionPerformed(ActionEvent e){
25       System.out.println("Don't do it, you might regret it!"
             );
26    }
27 }
28
29    class DevilListener implements ActionListener{
30       public void actionPerformed(ActionEvent e){
31          System.out.println("Come on, do it!");
32       }
33    }
34 }
```

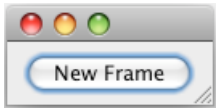


GAMBAR 5.5. Should I do it?



GAMBAR 5.6. Angel and Devil

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class NewFrame extends JFrame implements
       ActionListener {
6      JPanel panel;
7      JButton button;
8
9      NewFrame(){
10         panel = new JPanel();
11         button = new JButton("New Frame");
12         button.addActionListener(this);
13         getContentPane().add(panel);
14         panel.add(button);
15         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16         pack();
17         setVisible(true);
18     }
19
20     public void actionPerformed(ActionEvent e){
21         Object temp = e.getSource();
22         if(temp == button)
23         {
24             JButton b;
25
26             JFrame smallFrame;
27             smallFrame = new JFrame();
28             b = new JButton("Hello!");
29             smallFrame.getContentPane().add(b);
30             smallFrame.pack();
31 //                      smallFrame.setSize(100,100);
32             smallFrame.setVisible(true);
33         }
34     }
35
36     public static void main(String args[]){
37         NewFrame nf = new NewFrame();
38     }
39 }
```

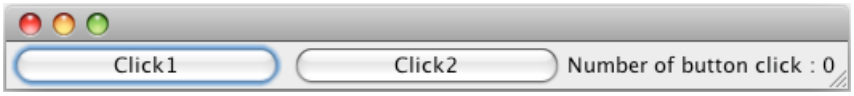


GAMBAR 5.7. NewFrame

GAMBAR 5.8. New Frame Pop

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class ClickMe extends JFrame implements
       ActionListener {
6      static int numClick = 0;
7      JPanel panel;
8      JButton b1, b2;
9      JLabel label;
10     String TEXT = "Number of button click : ";
11
12     ClickMe() {
13         panel = new JPanel(new GridLayout(0,3));
14         b1 = new JButton("Click1");
15         b2 = new JButton("Click2");
16         label = new JLabel(TEXT + "0  ");
17
18         b1.addActionListener(this);
19         b2.addActionListener(this);
20
21         getContentPane().add(panel);
22         panel.add(b1);
23         panel.add(b2);
24         panel.add(label);
25         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26         pack();
27         setVisible(true);
28     }
29
```

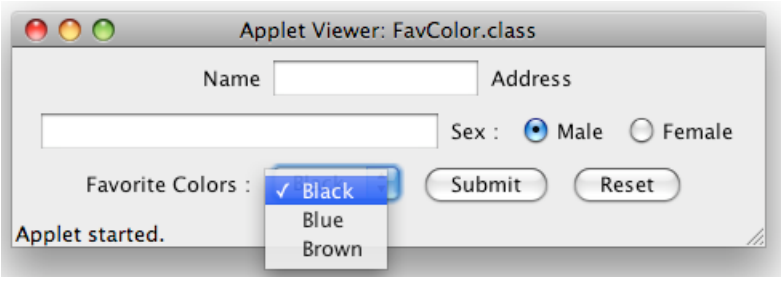
```
30      public void actionPerformed(ActionEvent e){
31          Object temp = e.getSource();
32
33         if(temp == b1){
34              numClick = numClick + 1;
35          }
36          if(temp == b2){
37              numClick = numClick + 2;
38          }
39
40          label.setText(TEXT + numClick);
41      }
42
43      public static void main(String args[]){
44          ClickMe cm = new ClickMe();
45      }
46  }
```



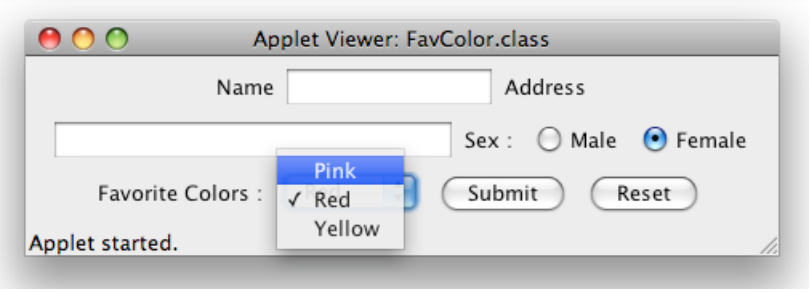
GAMBAR 5.9. Two Button

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class FavColor extends JApplet implements
       ItemListener{
6      JPanel panel;
7      JLabel lName, lAddress, lSex,lFavColors;
8      JTextField tfName, tfAddress;
9      JRadioButton rbMale, rbFemale;
10     JComboBox cb;
11     JButton bSubmit, bReset;
12     ButtonGroup bGroup;
13
14     public void init(){
15         panel = new JPanel();
16         lName = new JLabel("Name");
17         lAddress = new JLabel("Address");
18         lSex = new JLabel("Sex : ");
```

```
19          lFavColors = new JLabel("Favorite Colors : ");
20          tfName = new JTextField(10);
21          tfAddress = new JTextField(20);
22          rbMale = new JRadioButton("Male");
23          rbFemale = new JRadioButton("Female");
24          cb = new JComboBox();
25          bSubmit = new JButton("Submit");
26          bReset = new JButton("Reset");
27          bGroup = new ButtonGroup();
28
29          getContentPane().add(panel);
30          panel.add(lName);
31          panel.add(tfName);
32          panel.add(lAddress);
33          panel.add(tfAddress);
34          panel.add(lSex);
35          panel.add(rbMale);
36          panel.add(rbFemale);
37          panel.add(lFavColors);
38          panel.add(cb);
39          panel.add(bSubmit);
40          panel.add(bReset);
41          bGroup.add(rbMale);
42          bGroup.add(rbFemale);
43          rbMale.addItemListener(this);
44          rbFemale.addItemListener(this);
45      }
46
47      public void itemStateChanged(ItemEvent ev){
48          Object obj=ev.getSource();
49          if(obj == rbMale){
50              cb.removeAllItems();
51              cb.addItem("Black");
52              cb.addItem("Blue");
53              cb.addItem("Brown");
54          }
55          if(obj == rbFemale){
56              cb.removeAllItems();
57              cb.addItem("Pink");
58              cb.addItem("Red");
59              cb.addItem("Yellow");
60          }
61      }
62  }
```



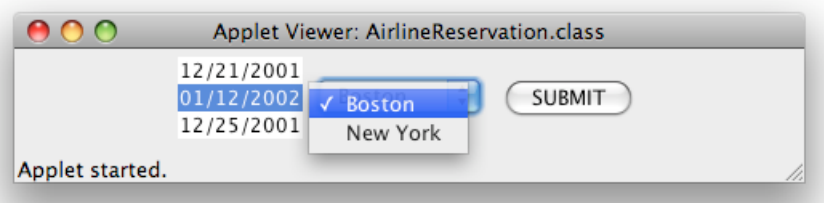
GAMBAR 5.10. FavColor Male



GAMBAR 5.11. FavColor Female

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4
5  public class AirLineReservation extends JApplet
6     implements FocusListener{
7     String date[]={"12/21/2001","01/12/2002","12/25/2001"};
8     String dest[]={"Chicago","Boston","New York",
9     "California","Atlanta","Mexico"};
10    JPanel panelObj;
11    JComboBox destCombo;
12    JList dateList;
13    JButton submit;
14
15    public void init(){
16       dateList=new JList(date);
17       dateList.addFocusListener(this);
18       destCombo=new JComboBox();
19       submit=new JButton("SUBMIT");
20       panelObj=new JPanel();
```

```
21
22          getContentPane().add(panelObj);
23          panelObj.add(dateList);
24          panelObj.add(destCombo);
25          panelObj.add(submit);
26      }
27
28      public void focusLost(FocusEvent e){
29          Object temp=e.getSource();
30
31          if(temp==dateList){
32              destCombo.removeAllItems();
33
34              int selection=dateList.getSelectedIndex();
35              destCombo.addItem(dest[selection]);
36              destCombo.addItem(dest[selection+1]);
37          }
38      }
39
40      public void focusGained(FocusEvent e){
41          //Will not be handled
42      }
43  }
```

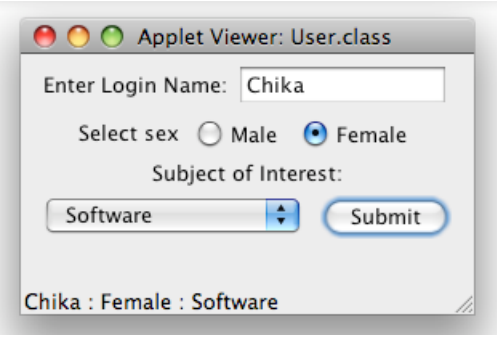


GAMBAR 5.12. Airline Reservation

```
1   import java.awt.*;
2   import javax.swing.*;
3   import java.awt.event.*;
4
5   public class User extends JApplet implements
6   FocusListener,ItemListener,ActionListener{
7       //Declaration for data entry controls
8       JPanel p1;
9       JLabel l1,l3,l4;
10      JTextField tf1;
```

```
11      JRadioButton male;
12      JRadioButton female;
13      JComboBox c1;
14      JButton b1;
15      String subjectarr[]={"Software","Movies",
16      "Literature","Society and People"};
17      String name, sex, subject;
18
19      public void init(){
20         p1=new JPanel();
21         getContentPane().add(p1);
22         l1=new JLabel("Enter Login Name:");
23         l3=new JLabel("Select sex");
24         l4=new JLabel("Subject of Interest:");
25         tf1=new JTextField(10);
26         male=new JRadioButton("Male");
27         female=new JRadioButton("Female");
28         c1=new JComboBox(subjectarr);
29         b1=new JButton("Submit");
30
31         p1.add(l1);
32         p1.add(tf1);
33         p1.add(l3);
34         p1.add(male);
35         p1.add(female);
36         p1.add(l4);
37         p1.add(c1);
38         p1.add(b1);
39         male.addItemListener(this);
40         female.addItemListener(this);
41         tf1.addFocusListener(this);
42         c1.addActionListener(this);
43         b1.addActionListener(this);
44      }
45
46      public void itemStateChanged(ItemEvent ev){
47         Object obj=ev.getSource();
48         if(obj==male)
49            sex="Male";
50         if(obj==female)
51            sex="Female";
52         if(obj==c1){
53            subject=String.valueOf(c1.getSelectedItem());
54            getAppletContext().showStatus(subject);
55         }
```

```
56      }
57
58      public void focusLost(FocusEvent e){
59          Object temp=e.getSource();
60          if(temp==tf1){
61              String strName=tf1.getText();
62              if(strName.length()!=0)
63                  name=strName;
64              else
65                  getAppletContext().showStatus("Name is empty");
66          }
67      }
68
69      public void focusGained(FocusEvent e){
70          //Will not be handled
71      }
72
73      public void actionPerformed(ActionEvent e){
74          Object obj=e.getSource();
75
76          if(obj==b1){
77              subject=String.valueOf(c1.getSelectedItem());
78              String record=name+" : "+sex+" : "+subject;
79              getAppletContext().showStatus(record);
80          }
81      }
82  }
```

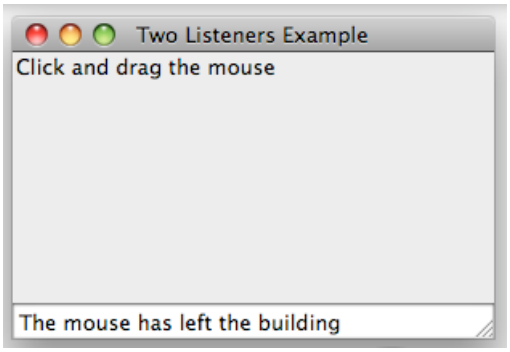


GAMBAR 5.13

```
1  import java.awt.*;
2  import java.awt.event.*;
```

```
3
4  public class TwoListener implements MouseMotionListener,
       MouseListener {
5      Frame f;
6      TextField tf;
7
8      public TwoListener(){
9          f = new Frame("Two␣Listeners␣Example");
10         tf = new TextField(30);
11     }
12
13     public void launchFrame(){
14         Label label = new Label("Click␣and␣drag␣the␣mouse");
15         f.add(label, BorderLayout.NORTH);
16         f.add(tf, BorderLayout.SOUTH);
17         f.addMouseMotionListener(this);
18         f.addMouseListener(this);
19         f.setSize(300,200);
20         f.setVisible(true);
21     }
22
23     public void mouseDragged(MouseEvent e){
24         String s = "Mouse␣dragging␣:␣X␣=␣" + e.getX() + "␣Y␣=␣
               " + e.getY();
25         tf.setText(s);
26     }
27
28     public void mouseEntered(MouseEvent e){
29         String s = "The␣mouse␣entered";
30         tf.setText(s);
31     }
32
33     public void mouseExited(MouseEvent e){
34         String s = "The␣mouse␣has␣left␣the␣building";
35         tf.setText(s);
36     }
37
38     public void mouseMoved(MouseEvent e){}
39     public void mousePressed(MouseEvent e){}
40     public void mouseClicked(MouseEvent e){}
41     public void mouseReleased(MouseEvent e){}
42
43     public static void main(String args[]){
44         TwoListener two = new TwoListener();
45         two.launchFrame();
```

```
46      }
47  }
```

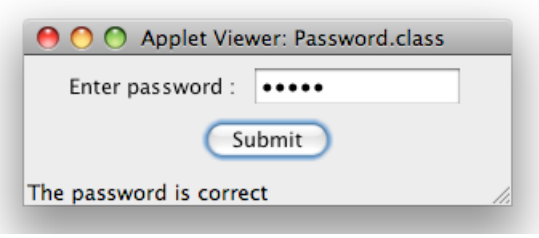


GAMBAR 5.14. TwoListener

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class Password extends JApplet implements
       ActionListener{
6      JPanel panel;
7      JLabel lPassword;
8      JPasswordField tfPassword;
9      JButton bSubmit;
10 //       JButton bReset;
11
12     public void init(){
13         panel = new JPanel();
14         getContentPane().add(panel);
15         lPassword = new JLabel("Enter password :");
16         tfPassword = new JPasswordField(10);
17         bSubmit = new JButton("Submit");
18 //                 bReset = new JButton("Reset");
19
20         panel.add(lPassword);
21         panel.add(tfPassword);
22         panel.add(bSubmit);
23 //               panel.add(bReset);
24
25         bSubmit.addActionListener(this);
26 //                 bReset.addActionListener(this);
```

```
27      }
28
29      public void actionPerformed(ActionEvent e){
30          Object temp=e.getSource();
31          boolean isCorrect = true;
32
33          if(temp == bSubmit){
34              char[] correctPassword = {'c','h','i','k','a'};
35
36              if (tfPassword.getPassword().length
37                  != correctPassword.length){
38                  isCorrect = false;
39              }
40              else{
41                  for (int i = 0; i < tfPassword.getPassword().
                        length;
42                  i++){
43                      if (tfPassword.getPassword()[i] !=
44                          correctPassword[i]){

                                isCorrect = false;
45                      }
46                  }
47              }
48
49              if (isCorrect == true){
50                  getAppletContext().showStatus("The password is
                        correct");
51              }
52              else{
53                  getAppletContext().showStatus("The password is
                        incorrect");
54              }
55          }
56      }
57  }
```



GAMBAR 5.15. Password

6. Exception Handling

```
1  public class HelloWorld
2  {
3          public static void main(String args[])
4          {
5                  int i = 0;
6                  String[] greetings = {"Hello World!", "No, I mean it!", "HELLO WC
7                  while (i < 4)
8  //                 while (i < 3)
9                  {
10                         try
11                         {
12                                 System.out.println(greetings[i]);
13                                 i++;
14                         }
15                         catch (ArrayIndexOutOfBoundsException e)
16                         {
17                                 System.out.println("Re-setting Index Value");
18                                 i = 0;
19                         }
20                         finally
21                         {
22                                 System.out.println("This is always printed");
23                         }
24                 }
25         }
26 }
```

```
1  public class TryCatch
2  {
3          public static void main(String args[])
4          {
5                  int array[] = {0,0};
6                  int num1, num2, result = 0;
7                  num1 = 100;
8                  num2 = 0;
9                  try
10                 {
11                         result = num1/num2;
12                         System.out.println(num1/array[2]);
13                 }
14                 catch (ArithmeticException e)
15                 {
16                         System.out.println("Error! Division by zero");
```

```
17                  }
18                  catch (ArrayIndexOutOfBoundsException e)
19                  {
20                          System.out.println("Error! Array out of bounds!");
21                  }
22                  catch (Exception e)
23                  {
24                          System.out.println("Some other error");
25                  }
26                  try
27                  {
28                          System.out.println(num1/array[2]);
29                  }
30                  catch (ArrayIndexOutOfBoundsException e)
31                  {
32                          System.out.println("Error! Array out of bounds!");
33                  }
34                  catch (Exception e)
35                  {
36                          System.out.println("Some other error");
37                  }
38                  System.out.println("The result is :" + result);
39          }
40  }
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class IllegalAgeException extends Exception
{
        public String getMessage()
        {
                return "Invalid age. The customer should not be provided with a c
        }
}
public class CustomerNew extends JApplet implements ActionListener
{
        int custAge;
        JPanel panelObject;
        JButton buttonAccept;
        JLabel labelCustAge;
        JTextField textCustAge;
        void setAge(int age) throws IllegalAgeException
        {
                if ((age < 20) || (age > 60))
```

```
                        throw new IllegalAgeException();
                custAge = age;
        }
        public void init()
        {
                panelObject = (JPanel)getContentPane();
                panelObject.setLayout(new FlowLayout());
                labelCustAge = new JLabel("Enter customer age : ");
                textCustAge = new JTextField(5);
                buttonAccept = new JButton("Store Details");
                panelObject.add(labelCustAge);
                panelObject.add(textCustAge);
                panelObject.add(buttonAccept);
                buttonAccept.addActionListener(this);
        }
        public void actionPerformed(ActionEvent evt)
        {
                Object obj = evt.getSource();
                if (obj == buttonAccept)
                {
                        CustomerNew custObj = new CustomerNew();
                        int age = Integer.parseInt(textCustAge.getText());
                        try
                        {
                                custObj.setAge(age);
                                getAppletContext().showStatus("Valid entry for cu
                        }
                        catch (IllegalAgeException e)
                        {
                                getAppletContext().showStatus(e.getMessage());
                        }
                }

        }
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class IllegalAgeException extends Exception
{
        public String getMessage()
        {
                return "Invalid age. The customer should not be provided with a c
        }
```

```
}
public class CustomerNew extends JApplet implements ActionListener
{
        int custAge;
        JPanel panelObject;
        JButton buttonAccept;
        JLabel labelCustAge;
        JTextField textCustAge;
        void setAge(int age) throws IllegalAgeException
        {
                if ((age < 20) || (age > 60))
                        throw new IllegalAgeException();
                custAge = age;
        }
        public void init()
        {
                panelObject = (JPanel)getContentPane();
                panelObject.setLayout(new FlowLayout());
                labelCustAge = new JLabel("Enter customer age : ");
                textCustAge = new JTextField(5);
                buttonAccept = new JButton("Store Details");
                panelObject.add(labelCustAge);
                panelObject.add(textCustAge);
                panelObject.add(buttonAccept);
                buttonAccept.addActionListener(this);
        }
        public void actionPerformed(ActionEvent evt)
        {
                Object obj = evt.getSource();
                if (obj == buttonAccept)
                {
                        CustomerNew custObj = new CustomerNew();
                        int age = Integer.parseInt(textCustAge.getText());
                        try
                        {
                                custObj.setAge(age);
                                getAppletContext().showStatus("Valid entry for cu
                        }
                        catch (IllegalAgeException e)
                        {
                                getAppletContext().showStatus(e.getMessage());
                        }
                }

        }
```

```
}

class CustomerCareExecutive
{
        String executiveName;
        int rating;
        public void displayDetails()
        {
                System.out.println(executiveName);
                System.out.println(rating);
        }
}
public class ExecutiveCollection
{
        CustomerCareExecutive exObjects[];

        public ExecutiveCollection()
    {
                exObjects = new CustomerCareExecutive[3];
/*
                for (int i=0; i < 3 ; i++)
                {
                        exObjects[i] = new CustomerCareExecutive();
                }
*/
                try
                {
           exObjects[0].executiveName="One Corp.";
           exObjects[0].rating = 10;
           exObjects[1].executiveName="Two Inc.";
           exObjects[1].rating = 9;
           exObjects[2].executiveName="Three";
           exObjects[2].rating = 8;
                }
                catch(NullPointerException e)
                {
                        System.out.println("Memory for the array has not been allo
                }
        }

        public void displayCollection()
        {
                try
                {
                        for(int ctr=0;ctr < 4; ctr++)
```

```
                        {
                                exObjects[ctr].displayDetails();
                        }
                }
                catch(ArrayIndexOutOfBoundsException e)
                {
                        System.out.println("Array out of bounds!");
                }
                catch(NullPointerException e)
                {
                        System.out.println("Memory not allocated for the array!")
                }
        }

        public static void main(String args[])
        {
                ExecutiveCollection collectionObj;
                collectionObj = new ExecutiveCollection();
//              collectionObj.displayCollection();
                System.out.println("All Records displayed");
        }
}
```

Bagian 3. Packages and Streams

7. Thread and Multithreading

```
1  public class ThreadDemo
2  {
3          public static void main(String[] args)
4          {
5                  Counter c1 = new Counter("C1"); //di
                       implementasi di counter.java
6                  Thread c2 = new Counter("C2");  //upcasting
7                  Thread c3 = new Thread("C3");   //gag ada
                       gunanya, krn tdk diimplementasi
8                  c1.start();
9                  c2.start();
10                 c3.start();
11         }
12 }
```

```
public class ThreadDemo1
{
        public static void main(String[] args)
        {
                Counter1 c1 = new Counter1("C1",Thread.MAX_PRIORITY);
//both
                Thread c2 = new Counter1("C2",Thread.MIN_PRIORITY);
//are acceptable
                c1.start();
                c2.start();
        }
}
```

```
import javax.swing.*;
import java.awt.Dimension;
import java.awt.BorderLayout;
public class ThreadDemo2
{
        public static void main(String[] args)
        {
                Counter2 c1 = new Counter2("C1");
                Counter2 c2 = new Counter2("C2");
                JFrame frame = new JFrame();
                frame.setSize(new Dimension(800,120));
                frame.getContentPane().add(c1,BorderLayout.NORTH);
                frame.getContentPane().add(c2,BorderLayout.SOUTH);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
                //frame.pack();  //di 1.3 blm ada setPreferredSize
                frame.setVisible(true);
                Thread t1 = new Thread(c1);
                Thread t2 = new Thread(c2);
                t1.setPriority(Thread.MAX_PRIORITY);
                t2.setPriority(Thread.MIN_PRIORITY);
                t1.start();
                t2.start();
                try     {
                        Thread.sleep(20000);
                }
                catch(java.lang.InterruptedException e){
                                e.printStackTrace();
                }
                t1.setPriority(Thread.MIN_PRIORITY);
                t2.setPriority(Thread.MAX_PRIORITY);
        }
}

import javax.swing.*;
import java.awt.Dimension;
import java.awt.BorderLayout;
import java.awt.Font;
public class ThreadDemo3
{
        public static void main(String[] args)
        {
                Counter3 c1 = new Counter3("C1");
                Counter2 c2 = new Counter2("C2");
                JFrame frame = new JFrame();
                frame.setSize(new Dimension(800,160));
                frame.getContentPane().add(c1,BorderLayout.NORTH);
                frame.getContentPane().add(c2,BorderLayout.CENTER);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                //frame.pack();
                frame.setVisible(true);
                final Thread t1 = new Thread(c1);       //diakses dari inner class
                Thread t2 = new Thread(c2);
                t1.setPriority(Thread.MAX_PRIORITY);
                t2.setPriority(Thread.MIN_PRIORITY);
                Waiter w = new Waiter("Waiter")
                {
                        public void run(){
                                while(true){
                                        setText(getName() + " : "+number +" @ "+S
```

```
                                        number++;
                                        try{
                                                t1.join();
//menunggu t1 selesai baru t3 dijalankan
                                                Thread.sleep(5);
                                        }catch(java.lang.InterruptedException e){
                                                e.printStackTrace();
                                        }
                                }
                        }
                };
                frame.getContentPane().add(w,BorderLayout.SOUTH);
                Thread t3 = new Thread(w);
                t1.start();
                t2.start();
                t3.start();

        }
}
abstract class Waiter extends JLabel implements Runnable
{
        int number;
                //private String name;
        public Waiter(String name){
                        setName(name);
                        Font f = new Font("Arial",Font.BOLD,35);
                        setFont(f);
        }
}

//package simple;
public class ThreadTester
{
        public static void main(String[] args)
        {
                HelloRunner r = new HelloRunner();
                Thread t = new Thread(r);
                t.start();
        }
}
class HelloRunner implements Runnable
{
        int i;
        public void run()
        {
```

```
                i = 0;
                while (true)
                {
                        System.out.println("Hello " + i++);
                        if (i == 50)
                        {
                                break;
                        }
                }
        }
}

public class PrintMessage extends Thread
{
        String name;
    PrintMessage(String name)
    {
                this.name=name;
    }

        public void run()
    {
                for (; ;)
        {
                        System.out.println("Hello! " + name + " Welcome to the Wo
            try
                        {
                                sleep(5000);
                        }
                        catch(InterruptedException e)
                        {
                                System.out.println("Exception was thrown:" + e);
            }
                }
    }

        public static void main(String arg[])
    {
                PrintMessage thread1=new PrintMessage("Thread1");
        PrintMessage thread2=new PrintMessage("Thread2");
        PrintMessage thread3=new PrintMessage("Thread3");
        thread1.start();
        thread2.start();
        thread3.start();
    }
```

```
}

public class SimpleThreads {
    //Display a message, preceded by the name of the current thread
static void threadMessage(String message) {         String threadName = Thread.cu
System.out.format("%s: %s%n", threadName, message);      }
    private static class MessageLoop implements Runnable {
public void run() {             String importantInfo[] = {
"Mares eat oats",                 "Does eat oats",
"Little lambs eat ivy",                 "A kid will eat ivy too"
};             try {                for (int i = 0; i < importantInfo.length; i+
//Pause for 4 seconds                    Thread.sleep(4000);
//Print a message                       threadMessage(importantInfo[i]);
}            } catch (InterruptedException e) {
threadMessage("I wasn't done!");             }         }
}
    public static void main(String args[]) throws InterruptedException {
        //Delay, in milliseconds before we interrupt MessageLoop
//thread (default one hour).          long patience = 1000 * 60 * 60;
        //If command line argument present, gives patience in seconds.
if (args.length > 0) {             try {                  patience = Long.parseLong
} catch (NumberFormatException e) {                 System.err.println("Argument
System.exit(1);             }
        }
        threadMessage("Starting MessageLoop thread");
long startTime = System.currentTimeMillis();          Thread t = new Thread(new Me
t.start();
        threadMessage("Waiting for MessageLoop thread to finish");
//loop until MessageLoop thread exits          while (t.isAlive()) {
threadMessage("Still waiting...");             //Wait maximum of 1 second for Mes
//finish.             t.join(1000);              if (((System.currentTimeMillis() -
t.isAlive()) {                  threadMessage("Tired of waiting!");
t.interrupt();                  //Shouldn't be long now -- wait indefinitely
t.join();             }
        }         threadMessage("Finally!");     } }

public class HelloRunnable implements Runnable {
    public void run() {         System.out.println("Hello from a thread!");
}
    public static void main(String args[]) {         (new Thread(new HelloRunnabl
}
}

package simple;
class HelloRunner implements Runnable
{
```

```
        int i;
        public void run()
        {
                i = 0;
                while (true)
                {
                        System.out.println("Hello " + i++);
                        if (i == 50)
                        {
                                break;
                        }
                }
        }
}

public class HelloThread extends Thread {
    public void run() {         System.out.println("Hello from a thread!");
}
    public static void main(String args[]) {         (new HelloThread()).start();
}
}

public class Counter extends Thread
{
        private int number;
        //private String name;
        public Counter(String name){
                setName(name);
        }
        public void run(){
                while(true){
                        System.out.println(getName() + " : "+number +" @ "+System
                        number++;
                        try{
                                sleep(2000);
                        }catch(java.lang.InterruptedException e){
//exception khusus untuk thread
                                e.printStackTrace();
                        }
                }
        }
}

public class Counter1 extends Thread
{
        private int number;
```

```
        //private String name;
        public Counter1(String name,int priority){
                setPriority(priority);
                setName(name);
        }
        public void run(){
                while(true){
                        System.out.println(getName() + " : "+number +" @ "+System
                        number++;
                        try{
                                sleep(2);
                        }catch(java.lang.InterruptedException e){
                                e.printStackTrace();
                        }
                }
        }
}

import javax.swing.JLabel;
import java.awt.Font;
public class Counter2 extends JLabel implements Runnable
{
        private int number;
        //private String name;
        public Counter2(String name){
                setName(name);
                Font f = new Font("Arial",Font.BOLD,35);
                setFont(f);
        }
        public void run(){
                while(true){
                        setText(getName() + " : "+number +" @ "+System.currentTim
                        number++;
                        try{
                                Thread.sleep(5);
                        }catch(java.lang.InterruptedException e){
                                e.printStackTrace();
                        }
                }
        }
}

import javax.swing.JLabel;
import java.awt.Font;
public class Counter3 extends JLabel implements Runnable
```

```
{
        private int number;
        //private String name;
        public Counter3(String name){
                setName(name);
                Font f = new Font("Arial",Font.BOLD,35);
                setFont(f);
        }
        public void run(){
                while(number < 1000){
                        setText(getName() + " : "+number +" @ "+System.currentTim
                        number++;
                        try{
                                Thread.sleep(5);
                        }catch(java.lang.InterruptedException e){
                                e.printStackTrace();
                        }
                }
        }
}

import javax.swing.JLabel;
import java.awt.Font;
public class Counter4 extends JLabel implements Runnable
{
        private int number;
        //private String name;
        public Counter4(String name){
                setName(name);
                Font f = new Font("Arial",Font.BOLD,35);
                setFont(f);
        }
        public void run(){
                while(number < 1000){
                        setText(getName() + " : "+number +" @ "+System.currentTim
                        number++;
                        try{
                                Thread.sleep(5);
                        }catch(java.lang.InterruptedException e){
                                e.printStackTrace();
                        }
                }
        }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Customer extends JApplet implements ActionListener
{
        String custName;
        String custPassword;
        JPanel panelObject;
        JLabel labelCustName;
        JLabel labelCustPassword;
        JTextField textCustName;
        JPasswordField textCustPassword;
        JButton buttonLogin;

        void setPassword(String password)
        {
                custPassword = password;
        }

        public void init()
        {
                panelObject = (JPanel)getContentPane();
                panelObject.setLayout(new FlowLayout());
                labelCustName = new JLabel("Customer Login Name");
                labelCustPassword = new JLabel("Password");
                textCustName = new JTextField(15);
                textCustPassword = new JPasswordField(15);
                buttonLogin=new JButton("Login");
                panelObject.add(labelCustName);
                panelObject.add(textCustName);
                panelObject.add(labelCustPassword);
                panelObject.add(textCustPassword);
                panelObject.add(buttonLogin);
                buttonLogin.addActionListener(this);
        }

        public void actionPerformed(ActionEvent evt)
        {
                Object obj=evt.getSource();
                if(obj==buttonLogin)
                {
                        String password=new String(textCustPassword.getPassword()
                        try
                        {
                                if(password.length()<6||password.length()>10)
```

```
                                        throw new PasswordException();
                        }
                        catch(PasswordException e)
                        {
                                System.out.println(e);
                        }

                }
        }
}
class PasswordException extends Exception
{
        public String toString()
        {
                return "Exception : Password length not correct. Password should
        }
}
```

```
import java.util.Random;
public class Consumer implements Runnable {     private Drop drop;
    public Consumer(Drop drop) {          this.drop = drop;
}
    public void run() {          Random random = new Random();
for (String message = drop.take(); ! message.equals("DONE");
message = drop.take()) {              System.out.format("MESSAGE RECEIVED: %s%n",
try {                  Thread.sleep(random.nextInt(5000));
} catch (InterruptedException e) {}          }     } }
```

```
final public class ImmutableRGB {
    //Values must be between 0 and 255.     final private int red;
final private int green;     final private int blue;     final private String nam
    private void check(int red, int green, int blue) {
if (red < 0 || red > 255                  || green < 0 || green > 255
|| blue < 0 || blue > 255) {             throw new IllegalArgumentException();
}     }
    public ImmutableRGB(int red, int green, int blue, String name) {
check(red, green, blue);          this.red = red;          this.green = green;
this.blue = blue;          this.name = name;     }
    public int getRGB() {          return ((red << 16) | (green << 8) | blue);
}
    public String getName() {          return name;     }
    public ImmutableRGB invert() {          return new ImmutableRGB(255 - red, 255
"Inverse of " + name);     } }
```

```
class NameRunnable implements Runnable
{
```

```
        public void run()
        {
                for (int x = 1 ; x <= 3; x++ )
                {
                        System.out.println(x +   u" Run by " + Thread.currentThrea
                        try
                        {
                                Thread.sleep(1000);
                        }
                        catch (InterruptedException e) {}
                }
        }
}
public class ManyNames
{
        public static void main(String[] args)
        {
                NameRunnable nr = new NameRunnable();
                Thread one = new Thread(nr);
                one.setName("Fred");
                Thread two = new Thread(nr);
                two.setName("Lucy");
                Thread three = new Thread(nr);
                three.setName("Ricky");
                one.start();
                two.start();
                three.start();
        }
}

import java.util.Random;
public class Producer implements Runnable {     private Drop drop;
    public Producer(Drop drop) {          this.drop = drop;
}
    public void run() {          String importantInfo[] = {
"Mares eat oats",             "Does eat oats",             "Little lambs eat ivy"
"A kid will eat ivy too"          };          Random random = new Random();
        for (int i = 0; i < importantInfo.length; i++) {
drop.put(importantInfo[i]);                 try {
Thread.sleep(random.nextInt(5000));                 } catch (InterruptedException e)
}          drop.put("DONE");      } }

public class ProducerConsumerExample {      public static void main(String[] args)
Drop drop = new Drop();          (new Thread(new Producer(drop))).start();
(new Thread(new Consumer(drop))).start();      } }
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.beans.*;
import java.util.Random;
public class ProgressBarDemo extends JPanel
                             implements ActionListener,
                                        PropertyChangeListener
{
    private JProgressBar progressBar;
    private JButton startButton;
    private JTextArea taskOutput;
//    private Task task;
/*
    class Task extends SwingWorker<Void, Void> {
        @Override
        public Void doInBackground() {
            Random random = new Random();
            int progress = 0;
            //Initialize progress property.
            setProgress(0);
            while (progress < 100) {
                //Sleep for up to one second.
                try {
                    Thread.sleep(random.nextInt(1000));
                } catch (InterruptedException ignore) {}
                //Make random progress.
                progress += random.nextInt(10);
                setProgress(Math.min(progress, 100));
            }
            return null;
        }
                @Override
        public void done() {
            Toolkit.getDefaultToolkit().beep();
            startButton.setEnabled(true);
            setCursor(null); //turn off the wait cursor
            taskOutput.append("Done!\n");
        }
    }
*/
        public ProgressBarDemo() {
        super(new BorderLayout());
        //Create the demo's UI.
        startButton = new JButton("Start");
```

```
        startButton.setActionCommand("start");
        startButton.addActionListener(this);
        progressBar = new JProgressBar(0, 100);
        progressBar.setValue(0);
        progressBar.setStringPainted(true);
        taskOutput = new JTextArea(5, 20);
        taskOutput.setMargin(new Insets(5,5,5,5));
        taskOutput.setEditable(false);
        JPanel panel = new JPanel();
        panel.add(startButton);
        panel.add(progressBar);
//          add(panel, BorderLayout.PAGE_START);
                add(panel);
        add(new JScrollPane(taskOutput), BorderLayout.CENTER);
        setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
        //Create and set up the window.
        JFrame frame = new JFrame("ProgressBarDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //Create and set up the content pane.
        JComponent newContentPane = new ProgressBarDemo();
//          newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);
        //Display the window.
        frame.pack();
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent evt)
        {
/*
        startButton.setEnabled(false);
        setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
        //Instances of javax.swing.SwingWorker are not reusuable, so
        //we create new instances as needed.
        task = new Task();
        task.addPropertyChangeListener(this);
        task.execute();
*/
    }
    public void propertyChange(PropertyChangeEvent evt)
        {
        if ("progress" == evt.getPropertyName()) {
            int progress = ((Integer)evt.getNewValue()).intValue();
            progressBar.setValue(progress);
//            taskOutput.append(String.format(
//                    "Completed %d%% of task.\n", task.getProgress()));
```

```
        }
    }
    public static void main(String[] args)
        {
                new ProgressBarDemo();
        }
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.beans.*;
import java.util.Random;
public class ProgressBarDemo2 extends JPanel
                              implements ActionListener,
                                         PropertyChangeListener {
    private JProgressBar progressBar;
    private JButton startButton;
    private JTextArea taskOutput;
    private Task task;
    class Task extends SwingWorker<Void, Void> {
        /*
         * Main task. Executed in background thread.
         */
        @Override
        public Void doInBackground() {
            Random random = new Random();
            int progress = 0;
            //Initialize progress property.
            setProgress(0);
            //Sleep for at least one second to simulate "startup".
            try {
                Thread.sleep(1000 + random.nextInt(2000));
            } catch (InterruptedException ignore) {}
            while (progress < 100) {
                //Sleep for up to one second.
                try {
                    Thread.sleep(random.nextInt(1000));
                } catch (InterruptedException ignore) {}
                //Make random progress.
                progress += random.nextInt(10);
                setProgress(Math.min(progress, 100));
            }
            return null;
        }
```

```
        /*
         * Executed in event dispatch thread
         */
        public void done() {
            Toolkit.getDefaultToolkit().beep();
            startButton.setEnabled(true);
            taskOutput.append("Done!\n");
        }
    }
    public ProgressBarDemo2() {
        super(new BorderLayout());
        //Create the demo's UI.
        startButton = new JButton("Start");
        startButton.setActionCommand("start");
        startButton.addActionListener(this);
        progressBar = new JProgressBar(0, 100);
        progressBar.setValue(0);
        //Call setStringPainted now so that the progress bar height
        //stays the same whether or not the string is shown.
        progressBar.setStringPainted(true);
        taskOutput = new JTextArea(5, 20);
        taskOutput.setMargin(new Insets(5,5,5,5));
        taskOutput.setEditable(false);
        JPanel panel = new JPanel();
        panel.add(startButton);
        panel.add(progressBar);
        add(panel, BorderLayout.PAGE_START);
        add(new JScrollPane(taskOutput), BorderLayout.CENTER);
        setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
    }
    /**
     * Invoked when the user presses the start button.
     */
    public void actionPerformed(ActionEvent evt) {
        progressBar.setIndeterminate(true);
        startButton.setEnabled(false);
        //Instances of javax.swing.SwingWorker are not reusuable, so
        //we create new instances as needed.
        task = new Task();
        task.addPropertyChangeListener(this);
        task.execute();
    }
    /**
     * Invoked when task's progress property changes.
     */
```

```
    public void propertyChange(PropertyChangeEvent evt) {
        if ("progress" == evt.getPropertyName()) {
            int progress = (Integer) evt.getNewValue();
            progressBar.setIndeterminate(false);
            progressBar.setValue(progress);
            taskOutput.append(String.format(
                        "Completed %d%% of task.\n", progress));
        }
    }
    /**
     * Create the GUI and show it. As with all GUI code, this must run
     * on the event-dispatching thread.
     */
    private static void createAndShowGUI() {
        //Create and set up the window.
        JFrame frame = new JFrame("ProgressBarDemo2");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //Create and set up the content pane.
        JComponent newContentPane = new ProgressBarDemo2();
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);
        //Display the window.
        frame.pack();
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

```
import java.util.concurrent.locks.Lock; import java.util.concurrent.locks.Reentra
public class Safelock {     static class Friend {         private final String na
private final Lock lock = new ReentrantLock();
        public Friend(String name) {             this.name = name;
}
        public String getName() {             return this.name;
}
```

```
        public boolean impendingBow(Friend bower) {
Boolean myLock = false;              Boolean yourLock = false;
try {                  myLock = lock.tryLock();
yourLock = bower.lock.tryLock();              } finally {
if (! (myLock && yourLock)) {                      if (myLock) {
lock.unlock();                        }                    if (yourLock) {
bower.lock.unlock();                             }                 }
}             return myLock && yourLock;          }
public void bow(Friend bower) {              if (impendingBow(bower)) {
try {                     System.out.format("%s: %s has bowed to me!%n",
this.name, bower.getName());                          bower.bowBack(this);
} finally {                     lock.unlock();
bower.lock.unlock();                    }             } else {
System.out.format("%s: %s started to bow to me, but" +
" saw that I was already bowing to him.%n",
this.name, bower.getName());                    }         }
        public void bowBack(Friend bower) {             System.out.format("%s: %s
this.name, bower.getName());          }     }
    static class BowLoop implements Runnable {          private Friend bower;
private Friend bowee;
        public BowLoop(Friend bower, Friend bowee) {
this.bower = bower;              this.bowee = bowee;          }
public void run() {              Random random = new Random();
for (;;) {                  try {                      Thread.sleep(random.nextInt(
} catch (InterruptedException e) {}                  bowee.bow(bower);
}         }     }
    public static void main(String[] args) {          final Friend alphonse = new
final Friend gaston = new Friend("Gaston");          new Thread(new BowLoop(alphons
new Thread(new BowLoop(gaston, alphonse)).start();      } }

public class SleepMessages {     public static void main(String args[]) throws In
String importantInfo[] = {             "Mares eat oats",
"Does eat oats",             "Little lambs eat ivy",
"A kid will eat ivy too"         };
        for (int i = 0; i < importantInfo.length; i++) {
//Pause for 4 seconds              Thread.sleep(4000);
//Print a message              System.out.println(importantInfo[i]);
}     } }

class SynchronizedCounter {     private int c = 0;
    public synchronized void increment() {          c++;      }
    public synchronized void decrement() {          c--;      }
    public synchronized int value() {          return c;      }
}
```

```
public class SynchronizedRGB {
    //Values must be between 0 and 255.     private int red;
private int green;     private int blue;     private String name;
    private void check(int red, int green, int blue) {
if (red < 0 || red > 255                  || green < 0 || green > 255
|| blue < 0 || blue > 255) {             throw new IllegalArgumentException();
}     }
    public SynchronizedRGB(int red, int green, int blue, String name) {
check(red, green, blue);          this.red = red;          this.green = green;
this.blue = blue;          this.name = name;     }
    public void set(int red, int green, int blue, String name) {
check(red, green, blue);          synchronized (this) {
this.red = red;              this.green = green;              this.blue = blue;
this.name = name;          }     }
    public synchronized int getRGB() {          return ((red << 16) | (green << 8)
}
    public synchronized String getName() {          return name;
}
    public synchronized void invert() {          red = 255 - red;
green = 255 - green;          blue = 255 - blue;          name = "Inverse of " + nam
} }
```

```
public class Deadlock {     static class Friend {          private final String nam
public Friend(String name) {              this.name = name;
}          public String getName() {              return this.name;
}          public synchronized void bow(Friend bower) {
System.out.format("%s: %s has bowed to me!%n",
this.name, bower.getName());              bower.bowBack(this);
}          public synchronized void bowBack(Friend bower) {
System.out.format("%s: %s has bowed back to me!%n",
this.name, bower.getName());          }     }
    public static void main(String[] args) {          final Friend alphonse = new
final Friend gaston = new Friend("Gaston");          new Thread(new Runnable() {
public void run() { alphonse.bow(gaston); }          }).start();
new Thread(new Runnable() {              public void run() { gaston.bow(alphonse);
}).start();     } }
```

```
public class Drop {     //Message sent from producer to consumer.
private String message;     //True if consumer should wait for producer to send m
//if producer should wait for consumer to retrieve message.
private boolean empty = true;
```

```
    public synchronized String take() {         //Wait until message is available
while (empty) {             try {                 wait();
} catch (InterruptedException e) {}         }         //Toggle status.
empty = true;         //Notify producer that status has changed.
notifyAll();         return message;     }
    public synchronized void put(String message) {         //Wait until message h
while (!empty) {             try {                 wait();
} catch (InterruptedException e) {}         }         //Toggle status.
empty = false;         //Store message.         this.message = message;
//Notify consumer that status has changed.         notifyAll();
} }

import java.util.Date;
import java.util.Calendar;
import java.util.GregorianCalendar;
import javax.swing.*;
import java.awt.*;
public class CustomerApplet extends JApplet implements Runnable
{
        JPanel panelObject;
    JLabel labelAnimation;
        String dealerInfo[]={"CellSoft, Inc. - Service charge free for 1 year","S
        int counter;
        Thread t1;

        public void init()
    {
                labelAnimation=new JLabel("     ");
                panelObject=new JPanel();
                panelObject=(JPanel)getContentPane();
                panelObject.setLayout(new FlowLayout());
                panelObject.setBackground(Color.blue);
                 panelObject.add(labelAnimation);
                t1= new Thread(this);
        t1.start(); //Starting thread
        }
        public void run() // body of the thread
    {
        for(;;)
        {
                        display();// This method displays
                          // date
            try
            {
                                t1.sleep(1000);
```

```
                }
            catch(InterruptedException e)
            {
                                showStatus("Thread interrupted");
                        }
                }
         }

          public void display() //displays date and
     {                          //time on the status bar
                Font f=new Font("Times New Roman",Font.BOLD,28);
                labelAnimation.setText(dealerInfo[counter]);
            counter++;
            if(counter>=3)
                counter=0;
     }
}

class NameRunnable implements Runnable
{
        public void run()
        {
                for (int x = 1 ; x <= 3; x++ )
                {
                        System.out.println(x + " Run by " +
                                Thread.currentThread().getName());
                }
        }
}
public class ManyNames0
{
        public static void main(String[] args)
        {
                NameRunnable nr = new NameRunnable();
                Thread one = new Thread(nr);
                one.setName("Fred");
                Thread two = new Thread(nr);
                two.setName("Lucy");
                Thread three = new Thread(nr);
                three.setName("Ricky");
                one.start();
                two.start();
                three.start();
        }
}
```

8. Java I/O Stream Classes

8.1. **Program 1.** Using random access file

```
1  try {
2     File f = new File("filename");
3     RandomAccessFile raf = new RandomAccessFile(f, "rw");
                        // Read a character
4     char ch = raf.readChar();                  // Seek to end of
          file
5     raf.seek(f.length());                  // Append to the end
6     raf.writeChars("aString");
7     raf.close();
8  }
9  catch (IOException e) {
10 }
```

File output demo

```
/*  *
* FileOutputDemo  *
* Demonstration of FileOutputStream and
* PrintStream classes  *
*/

import java.io.*;
class FileOutputDemo  {
   public static void main(String args[])         {
      FileOutputStream out; // declare a file output object
      PrintStream p; // declare a print stream object
      try {
         // Create a new file output stream
         // connected to "myfile.txt"
         out = new FileOutputStream("myfile.txt");
         // Connect print stream to the output stream
         p = new PrintStream( out );
         p.println ("This is written to a file");
         p.close();
      }
catch (Exception e){
         System.err.println ("Error writing to file");
}
}
}
```

File input demo

```
/*  *  * FileInputDemo  * Demonstrates FileInputStream and  * DataInputStream
*/ import java.io.*;
```

```
class FileInputDemo  {          public static void main(String args[])
{                  // args.length is equivalent to argc in C
if (args.length == 1)                {
try                    {                                    // Open the file that is
// command line parameter                                     FileInputStream fstream
FileInputStream(args[0]);
                                  // Convert our input stream to a
// DataInputStream                                  DataInputStream in =
new DataInputStream(fstream);
                                  // Continue to read lines while
// there are still some left to read
while (in.available() !=0)                                    {
// Print file line to screen
System.out.println (in.readLine());
}
                                  in.close();
}                           catch (Exception e)
{                                 System.err.println("File input error");
}               }                   else
System.out.println("Invalid parameters");        }
}
```

Create directory create new file

```
//This program creates a new directory   // and inside it creates a new .txt file
import java.io.*;          public class File1  {    public static void main ( Strin
]  args )  throws IOException {              //write the pathe name in the file co
File ab = new File ( "C:\\NewPath\\New" ) ;          //call the File??s boolean me
)    //to create a new directory for you   //in your specied path and get the patl
//by calling the getAbsolutePath ( )  method          if ( ab.mkdirs (
) )   {      System.out.println ( "New directory is created at: "
+ ab.getAbsolutePath ( ) ) ;    }    else  {       System.out.println ( "Already
}          //Now create a .txt file in the above created directory
File bc = new File ( ab, "uni.txt" ) ;          if ( bc.createNewFile (
) )   {      System.out.println ( "New text file is created at: "
+ bc.getAbsolutePath ( ) ) ;    }    else {      System.out.println ( "Not creat
}           }     }
```

Contoh create new file

```
public static void main  ( String [  ]  args )       {
String filePath="c:/tmp/foo.txt";       File newFile = new File ( filePath ) ;
try        {          if ( newFile.createNewFile ( ) )
{            System.out.println ( "New File Created." ) ;
}          else          {            System.out.println ( "File already exists."
}         }        catch  ( IOException ioe )         {
System.out.println ( "IOException = " + ioe.getMessage (  )
) ;        }       }
```

Contoh cek apakah file bisa ditulisi atau tidak

```
public class ff  {            public static void main ( String [
] args )  throws Exception {        File af_FileToTest = new File ( "C:/my.JPG"
boolean lb_IsWritable = false;             if  ( af_FileToTest.exists (
)  )  {          lb_IsWritable = af_FileToTest.canWrite ( ) ;
}  else  {          throw new FileNotFoundException ( af_FileToTest + " not exists
}             System.out.println ( lb_IsWritable ) ;
}     }
```

Contoh lihat isi direktori

```
File directory = new File ( "C:\\JavaSource" ) ;   File [ ]
    filesInDir = directory.listFiles ( ) ;        if (
  filesInDir != null )  {       int length = filesInDir.
  length;            for  ( int i = 0; i  <  length; ++i
  )  {           File f = filesInDir [ i ] ;
  if  ( f.isFile ( ) )  {              if  ( f.
  canRead ( )  )                System.out.println (
  "Can␣Read␣File:␣" + f.getName ( ) ) ;
  else                System.out.println ( "Can␣NOT␣Read
  ␣File:␣" + f.getName ( ) ) ;           }
  else if  ( f.isDirectory ( ) )  {
  System.out.println ( "Directory:␣" + f.getName ( ) )
  ;           }             }     }
```

Contoh read file

```
package MyProject
import java.io.BufferedInputStream;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
/**
* This program reads a text file line by line and print to
    the console. It uses
* FileOutputStream to read the file.
*
*/

public class FileInput {
   public static void main(String[] args) {
      File file = new File("C:\\MyFile.txt");
      FileInputStream fis = null;
      BufferedInputStream bis = null;
      DataInputStream dis = null;
```

```
20        try {
21           fis = new FileInputStream(file);
22           // Here BufferedInputStream is added for fast
                 reading.
23           bis = new BufferedInputStream(fis);
24           dis = new DataInputStream(bis);
25           // dis.available() returns 0 if the file does not
                 have more lines.
26           while (dis.available() != 0) {
27           // this statement reads the line from the file and
                 print it to
28           // the console.
29           System.out.println(dis.readLine());        }
30           // dispose all the resources after using them.
31           fis.close();
32           bis.close();
33           dis.close();
34        }
35        catch (FileNotFoundException e) {
36           e.printStackTrace();
37        }
38        catch (IOException e) {
39        e.printStackTrace();
40        }
41     }
42  }
```

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4  import java.io.*;
5  public class CustomerCareExecutive extends JFrame implements
       ActionListener,ItemListener
6  {
7           //Variable for the panel
8           JPanel panelObject;
9           JLabel l1,l2,l3,l4,l5,l6,l7,l8;
10          JTextField tf1;
11          JRadioButton c1,c2;
12          JComboBox cb1;
13          JList li1;
14          GridBagLayout gl;
15          GridBagConstraints gbc;
16          String criteria[] = {"Customer Satisfaction", "
                Productivity"};
```

```
17          String rating[]={"Outstanding","Excellent","Good","
                Poor"};
18          JButton b1;
19          String grade=new String();
20          public CustomerCareExecutive() {
21                  super("Customer Care Executive Window");
22                  gl = new GridBagLayout();
23                  gbc = new GridBagConstraints();
24                  panelObject = (JPanel)getContentPane();
25                  panelObject.setLayout(gl);
26
27              l1=new JLabel("Customer Care Executive
                    Performance Sheet");
28  l8=new JLabel("
    ----------------------------------------------------------------------
    ");
29                  l2=new JLabel("Name :");
30                  l3=new JLabel("Grade :");
31                  l4=new JLabel("Performance Criteria");
32                  l5=new JLabel("Rating");
33                  l6=new JLabel("
                        ----------------------------------");
34                  l7=new JLabel("----------");
35                  tf1=new JTextField(10);
36                  tf1.setText("Carol");
37
38                  c1=new JRadioButton("L1");
39                  c2=new JRadioButton("L2");
40                  c1.addActionListener(this);
41                  c2.addActionListener(this);
42
43                  cb1 = new JComboBox(criteria);
44                  li1=new JList(rating);
45
46                  b1=new JButton("Submit");
47                  b1.addActionListener(this);
48
49                  gbc.anchor = GridBagConstraints.NORTHWEST;
50                  gbc.gridx = 3;
51                  gbc.gridy = 1;
52                  gl.setConstraints(l1,gbc);
53                  panelObject.add(l1);
54
55
56                  gbc.gridx = 3;
```

```
57                  gbc.gridy = 3;
58                  gl.setConstraints(l8,gbc);
59                  panelObject.add(l8);
60
61                  gbc.gridx = 2;
62                  gbc.gridy = 9;
63                  gl.setConstraints(l2,gbc);
64                  panelObject.add(l2);
65
66
67                  gbc.gridx = 4;
68                  gbc.gridy = 9;
69                  gl.setConstraints(tf1,gbc);
70                  panelObject.add(tf1);
71
72
73                  gbc.gridx = 2;
74                  gbc.gridy = 14;
75                  gl.setConstraints(l3,gbc);
76                  panelObject.add(l3);
77
78                  gbc.gridx = 4;
79                  gbc.gridy = 14;
80                  gl.setConstraints(c1,gbc);
81                  panelObject.add(c1);
82                  c1.addItemListener(this);
83
84                  gbc.gridx = 5;
85                  gbc.gridy = 14;
86                  gl.setConstraints(c2,gbc);
87                  panelObject.add(c2);
88                  c2.addItemListener(this);
89
90                  gbc.gridx = 2;
91                  gbc.gridy = 19;
92                  gl.setConstraints(l4,gbc);
93                  panelObject.add(l4);
94
95                  gbc.gridx = 4;
96                  gbc.gridy = 19;
97                  gl.setConstraints(l5,gbc);
98                  panelObject.add(l5);
99
100                 gbc.gridx = 2;
101                 gbc.gridy = 22;
```

```
102                 gl.setConstraints(l6,gbc);
103                 panelObject.add(l6);
104
105
106                 gbc.gridx = 4;
107                 gbc.gridy = 22;
108                 gl.setConstraints(l7,gbc);
109                 panelObject.add(l7);
110
111                 gbc.gridx = 2;
112                 gbc.gridy = 26;
113                 gl.setConstraints(cb1,gbc);
114                 panelObject.add(cb1);
115
116                 gbc.gridx = 4;
117                 gbc.gridy = 26;
118                 gl.setConstraints(li1,gbc);
119                 panelObject.add(li1);
120                 gbc.gridx = 5;
121                 gbc.gridy = 30;
122                 gl.setConstraints(b1,gbc);
123                 panelObject.add(b1);
124                 setSize(500,500);
125                 setVisible(true);
126
127         }
128         public static void main(String args[]){
129                 new CustomerCareExecutive();
130                 }
131         public void actionPerformed(ActionEvent evt)
132         {
133                 Object obj=evt.getSource();
134                 if(obj==b1)
135                 {
136                         String name=tf1.getText();
137                         int selection;
138                         String perfcriteria=String.valueOf(
                                cb1.getSelectedItem());
139                         selection=li1.getSelectedIndex();
140
141                         String execrating=String.valueOf(li1
                                .getSelectedValue());
142
143
```

```
144                         String record=name+" : "+grade+" : "
                                +perfcriteria+" : "+execrating+"
                                ; ";
145
146                         try
147                         {
148
149                         RandomAccessFile logFile=new
                                RandomAccessFile("Executive.txt"
                                ,"rw");
150                         logFile.seek(logFile.length());
151                         logFile.writeBytes(record);
152                         }
153                         catch(IOException ev)
154                         {
155                                 System.out.println("Cannot
                                        write to log file"+ev);
156                         }
157
158                 }
159
160         }
161           public void itemStateChanged(ItemEvent ev)
162           {
163                   Object obj=ev.getSource();
164                   if(obj==c1)
165                           grade="L1";
166                   if(obj==c2)
167                           grade="L2";
168           }
169   }
```

```
1  //LoginExecutive.java
2  //Use the stream class code given below as a guidance for
       reading data from the file
3  //Code needs to be modified to display the data on the
       screen
4  //Code has to be converted to an applet
5  import java.io.*;
6  import java.util.*;
7  public class CustomerCareExecutive1
8  {
9         BufferedReader bfreader;
10        FileInputStream fsreader;
11        InputStreamReader inputreader;
```

```
12              public CustomerCareExecutive()
13          {
14              try
15              {
16                  // Reading data to a file
17                  fsreader=new FileInputStream("Executive.txt");
18                  inputreader=new InputStreamReader(fsreader);
19                  bfreader=new BufferedReader(inputreader);
20                  String record=new String();
21                  while((record=bfreader.readLine())!=null)
22                          System.out.println(record);
23                  fsreader.close();
24              }
25              catch(FileNotFoundException fn)
26              {
27                   System.out.println("The specified file does
                        not exist");
28              }
29              catch(IOException fn)
30              {
31                   System.out.println("Error performing IO
                        Operation");
32              }
33
34          }
35          public static void main(String args[])
36              {
37                   CustomerCareExecutive lnexobj;
38                  lnexobj=new CustomerCareExecutive();
39          }
40  }
```

```
1   import java.io.*;
2   public class Dealer
3   {
4           String dealerName;
5           String dealerAddress;
6           String dealerPhone;
7           String dealerServices;
8           InputStreamReader keyReader;
9           BufferedReader bfReader;
10          FileOutputStream fsWriter;
11          public Dealer()
12          {
```

```
13                  try
14                  {
15                          keyReader = new InputStreamReader(
                                System.in);
16                          bfReader = new BufferedReader(
                                keyReader);
17                          System.out.println("Dealer␣Name␣:␣")
                                ;
18                          dealerName = bfReader.readLine();
19                          System.out.println("Address␣:␣");
20                          dealerAddress = bfReader.readLine();
21                          System.out.println("Phone␣Number␣:␣"
                                );
22                          dealerPhone = bfReader.readLine();
23                          System.out.println("Services␣Offered
                                ␣:␣");
24                          dealerServices = bfReader.readLine()
                                ;
25                          keyReader.close();
26                          bfReader.close();
27                          fsWriter = new FileOutputStream("E
                                :\\chikana\\codes\\java\\sem1\\
                                lesson8\\Dealer.txt", true);
28                          String temp = dealerName + ":" +
                                dealerAddress + ":" +
                                dealerPhone + ":" +
                                dealerServices;
29                          fsWriter.write(temp.getBytes());
30                          System.out.println("Finished␣writing
                                ␣dealer␣details␣to␣the␣file");
31                          fsWriter.close();
32                  }
33                  catch(FileNotFoundException fn)
34                  {
35                          System.out.println("The␣specified␣
                                file␣does␣not␣exist");
36                  }
37                  catch(IOException fn)
38                  {
39                          System.out.println("Error␣perfoming␣
                                IO␣Operation");
40                  }
41          }
42          public static void main(String args[])
43          {
```

```
44                  Dealer dealerObj;
45                  dealerObj = new Dealer();
46          }
47  }
```

```
 1  import java.awt.*;
 2  import java.awt.event.*;
 3  import javax.swing.*;
 4  import java.io.*;
 5  public class Login extends JApplet implements ActionListener
 6  {
 7          JPanel panel;
 8          JLabel lCustName;
 9          JTextField tfCustName;
10          JLabel lPassword;
11          JPasswordField pfPassword;
12          JButton bLogin;
13          public void init()
14          {
15                  panel = new JPanel();
16                  getContentPane().add(panel);
17                  lCustName = new JLabel("Customer␣Login␣Name␣
                        :␣");
18                  tfCustName = new JTextField(10);
19                  lPassword = new JLabel("Password␣:␣");
20                  pfPassword = new JPasswordField(10);
21                  bLogin = new JButton("Login");
22                  panel.add(lCustName);
23                  panel.add(tfCustName);
24                  panel.add(lPassword);
25                  panel.add(pfPassword);
26                  panel.add(bLogin);
27                  bLogin.addActionListener(this);
28          }
29          public void actionPerformed(ActionEvent evt)
30          {
31                  Object obj = evt.getSource();
32                  if (obj == bLogin)
33                  {
34                          String entry = tfCustName.getText()
                                + ":" + new String(pfPassword.
                                getPassword());
35
36                          try
```

```
37                              {
38                                      RandomAccessFile logFile = new
                                            RandomAccessFile("E:\\
                                           chikana\\codes\\java\\sem1
                                           \\lesson8\\customer.txt",
                                           "rw");
39                                      logFile.seek(logFile.length
                                            ());
40                                      logFile.writeBytes(entry +
                                            System.getProperty("line
                                            .separator"));
41                              }
42                              catch(IOException e)
43                              {
44                                      showStatus("Cannot write on
                                            to the log file " + e);
45                              }
46                      }
47              }
48  }
```

9. Server - Client Applications

```
1  import java.io.*;
2  import java.net.*;
3  public class MyServer
4  {
5          public static void main(String[] args) throws
               IOException
6          {
7                  ServerSocket server = new ServerSocket(1001)
                       ;
8          System.out.println("Server Ready");
9          Socket clientSocket = server.accept();
10         BufferedReader is = new BufferedReader (new
               InputStreamReader(clientSocket.getInputStream())
               );
11         DataOutputStream os = new DataOutputStream(
               clientSocket.getOutputStream());
12         String line = is.readLine();
13         System.out.print("We received: " + line);
14         os.writeBytes("Hello too Client!!!\n");
15         os.close();
16         is.close();
17         clientSocket.close();
18         }
19 }
```

```
1  import java.io.*;
2  import java.net.*;
3  public class HelloServer
4  {
5          public static void main(String[] args) throws
               IOException
6          {
7                  ServerSocket server = new ServerSocket(9999)
                       ;
8          System.out.println("Server Ready");
9          Socket clientSocket = server.accept();
10         BufferedReader is = new BufferedReader (new
               InputStreamReader(clientSocket.getInputStream())
               );
11         DataOutputStream os = new DataOutputStream(
               clientSocket.getOutputStream());
12         String line = is.readLine();
13         System.out.print("We received: " + line);
```

```
14          os.writeBytes("Hello too Client!!!\n");
15          os.close();
16          is.close();
17          clientSocket.close();
18          }
19  }
```

```
1   import java.io.*;
2   import java.net.*;
3   public class HelloClient
4   {
5           public static void main(String[] args)
6           throws IOException,NumberFormatException{
7           PrintWriter out;
8           BufferedReader in;
9           Socket client;
10                  if(args.length == 0)
11              client = new Socket("localhost",9999);
12          else
13              client = new Socket(args[0], Integer.parseInt(
                    args[1]));
14          in = new BufferedReader (new InputStreamReader(
                client.getInputStream()));
15          out = new PrintWriter(client.getOutputStream());
16          out.println("Hello Server !!!");
17          out.flush();
18          System.out.print(in.readLine());
19          in.close();
20          out.close();
21          client.close();
22          }
23  }
```

```
1   import java.net.*;
2   import java.io.*;
3   public class SimpleClient
4   {
5           public static void main(String args[])
6           {
7                   try
8                   {
9                           Socket s1 = new Socket("127.0.0.1",
                                5432);
```

```
10                          BufferedReader br = new
                                BufferedReader(new
                                InputStreamReader(s1.
                                getInputStream()));
11                          System.out.println(br.readLine());
12                          br.close();
13                          s1.close();
14                  }
15                  catch(ConnectException connExc)
16                  {
17                          System.err.println("Could not
                                connect to the server.");
18                  }
19                  catch(IOException e)
20                  {
21                  }
22          }
23  }
```

```
 1  import java.awt.*;
 2  import javax.swing.*;
 3  import java.awt.event.*;
 4  import java.io.*;
 5  import java.net.*;
 6  class User extends Object implements Serializable
 7  {
 8          String name;
 9          String password=new String();
10  }
11  public class SimpleClientApp extends JApplet implements
        ActionListener
12  {
13          JPanel panel;
14
15          JLabel labelName;
16          JLabel labelPassword;
17
18          JTextField textName;
19          JPasswordField textPassword;
20          JButton submit;
21          GridBagLayout gbObject;
22          GridBagConstraints gbc;
23          public void init()
24          {
25                  gbObject = new GridBagLayout();
```

```
26                  gbc = new GridBagConstraints();
27                  panel = (JPanel)getContentPane();
28                  panel.setLayout(gbObject);
29
30                  labelName = new JLabel("Login Name");
31                  labelPassword = new JLabel("Password");
32                  textName = new JTextField(15);
33                  textPassword = new JPasswordField(10);
34                  submit=new JButton("Submit");
35
36                  gbc.anchor = GridBagConstraints.NORTHWEST;
37                  gbc.gridx = 1;
38                  gbc.gridy = 5;
39                  gbObject.setConstraints(labelName,gbc);
40                  panel.add(labelName);
41
42                  gbc.anchor = GridBagConstraints.NORTHWEST;
43                  gbc.gridx = 4;
44                  gbc.gridy = 5;
45                  gbObject.setConstraints(textName,gbc);
46                  panel.add(textName);
47                  gbc.anchor = GridBagConstraints.NORTHWEST;
48                  gbc.gridx = 1;
49                  gbc.gridy = 8;
50                  gbObject.setConstraints(labelPassword,gbc);
51                  panel.add(labelPassword);
52                  gbc.anchor = GridBagConstraints.NORTHWEST;
53                  gbc.gridx = 4;
54                  gbc.gridy = 8;
55                  gbObject.setConstraints(textPassword,gbc);
56                  panel.add(textPassword);
57                  gbc.anchor = GridBagConstraints.NORTHWEST;
58                  gbc.gridx = 4;
59                  gbc.gridy = 17;
60                  gbObject.setConstraints(submit,gbc);
61                  panel.add(submit);
62                  submit.addActionListener(this);
63          }
64          public void actionPerformed(ActionEvent ev)
65          {
66                  Object obj=ev.getSource();
67                  if(obj == submit)
68                  {
69                          User userObj = new User();
70                          userObj.name = textName.getText();
```

```
71                          userObj.password = new String(
                                textPassword.getPassword());
72
73                          try
74                          {
75                                  Socket toServer;
76                                  toServer=new Socket
                                        ("127.0.0.1",1001);
77
78                                  ObjectOutputStream
                                        streamToServer=new
                                        ObjectOutputStream(
                                        toServer.getOutputStream
                                        ());
79                                  streamToServer.writeObject((
                                        User)userObj);
80                                  BufferedReader bf = new
                                        BufferedReader(new
                                        InputStreamReader(
                                        toServer.getInputStream
                                        ()));
81                                  String loginID = bf.readLine
                                        ();
82
83                                  getAppletContext().
                                        showStatus("Your login
                                        ID is : " + loginID);
84                                  streamToServer.close();
85                                  bf.close();
86                          }
87                          catch(Exception e)
88                          {
89                                  getAppletContext().
                                        showStatus("Exception..
                                        in connecting to server
                                        ");
90                          }
91                  }
92          }
93  }
```

```
1  import java.net.*;
2  import java.io.*;
3  public class SimpleServer
4  {
```

```
5           public static void main(String args[])
6           {
7                   ServerSocket s = null;
8                   System.out.println("Server listening... ");
9                   try
10                  {
11                          s = new ServerSocket(5432);
12                  }
13                  catch(IOException e)
14                  {
15                          e.printStackTrace();
16                  }
17                  while (true)
18                  {
19                          try
20                          {
21                                  Socket s1 = s.accept();
22                                  OutputStream s1out = s1.
                                        getOutputStream();
23                                  BufferedWriter bw = new
                                        BufferedWriter(new
                                        OutputStreamWriter(s1out
                                        ));
24                                  bw.write("Hello Net World!")
                                        ;
25                                  bw.close();
26                                  s1.close();
27                          }
28                          catch(IOException e)
29                          {
30                                  e.printStackTrace();
31                          }
32                  }
33          }
34  }
```

```
1   import java.awt.*;
2   import java.io.*;
3   import java.net.*;
4   class Customer implements Serializable
5   {
6           String custName;
7           String custPassword;
8   }
9   public class Server extends Thread
```

```
10 {
11         ServerSocket serverSocket;
12         public Server()
13         {
14                 try
15                 {
16                         serverSocket = new ServerSocket
                               (1001);
17                 }
18                 catch(IOException e)
19                 {
20                         fail(e, "Could not start server.");
21                 }
22                 System.out.println("Server started...");
23                 this.start();
24         }
25         public static void fail(Exception e, String str)
26         {
27                 System.out.println(str + "." + e);
28         }
29         public void run()
30         {
31                 try
32                 {
33                         while(true)
34                         {
35                                 Socket client = serverSocket
                                       .accept();
36                                 Connection con = new
                                       Connection(client);
37                         }
38                 }
39                 catch(IOException e)
40                 {
41                         fail(e, "Not listening!");
42                 }
43         }
44         public static void main(String args[])
45         {
46                 new Server();
47         }
48 }
49 class Connection extends Thread
50 {
51         protected Socket netClient;
```

```
52          protected ObjectInputStream fromClient;
53          protected PrintStream toClient;
54          public Connection(Socket client)
55          {
56                  netClient = client;
57                  try
58                  {
59                          fromClient = new ObjectInputStream(
                                netClient.getInputStream());
60                          toClient = new PrintStream(netClient
                                .getOutputStream());
61                  }
62                  catch(IOException e)
63                  {
64                          try
65                          {
66                                  netClient.close();
67                          }
68                          catch(IOException e1)
69                          {
70                                  System.err.println("Unable
                                        to set up streams" + e1)
                                        ;
71                                  return;
72                          }
73                  }
74                  this.start();
75          }
76          public void run()
77          {
78                  Customer clientMessage;
79                  try
80                  {
81                          for( ; ; )
82                          {
83                                  clientMessage = (Customer)
                                        fromClient.readObject();
84                                  if (clientMessage == null)
85                                          break;
86                                  toClient.println("Received
                                        from : " + clientMessage
                                        .custName);
87                          }
88                  }
89                  catch(IOException e)
```

```
90                  {}
91                  catch(ClassNotFoundException e1)
92                  {
93                          System.out.println("Error in reading
                                object " + e1);
94                  }
95                  finally
96                  {
97                          try
98                          {
99                                  netClient.close();
100                         }
101                         catch(IOException e)
102                         {}
103                 }
104         }
105 }
```

```
1  import java.io.*;
2  import java.net.*;
3  import java.awt.*;
4  import java.awt.event.*;
5  import javax.swing.*;
6  public class ServerApp extends JFrame
7  {
8          private JTextField enterField;
9          private JTextArea displayArea;
10         private ObjectOutputStream output;
11         private ObjectInputStream input;
12         private ServerSocket server;
13         private Socket connection;
14         private int counter = 1;
15         public ServerApp()
16         {
17                 super("Server");
18                 Container container = getContentPane();
19                 enterField = new JTextField();
20                 enterField.setEditable(false);
21                 enterField.addActionListener(new
                       ActionListener()
22                 {
23                         public void actionPerformed(
                               ActionEvent event)
24                         {
```

```
25                                  sendData(event.
                                        getActionCommand());
26                                  enterField.setText("");
27                              }
28                      }
29
30                      container.add(enterField, BorderLayout.NORTH
                            );
31                      displayArea = new JTextArea();
32                      container.add(new JScrollPane(displayArea),
                            BorderLayout.CENTER);
33                      setSize(300,150);
34                      setVisible(true);
35              }
36              public void runServer()
37              {
38                      try
39                      {
40                              server = new ServerSocket(12345,
                                    100);
41                              while (true)
42                              {
43                                      try
44                                      {
45                                              waitForConnection();
46                                              getStreams();
47                                              processConnection();
48                                      }
49                                      catch(EOFException
                                            eofException)
50                                      {
51                                              System.err.println("
                                                    Server
                                                    terminated
                                                    connection");
52                                      }
53                                      finally
54                                      {
55                                              closeConnection();
56                                              ++counter;
57                                      }
58                              }
59                      }
60                      catch(IOException ioException)
61                      {
```

```
62                      ioException.printStackTrace();
63              }
64      }
65      private void waitForConnection() throws IOException
66      {
67              displayMessage("Waiting for connection\n");
68              connection = server.accept();
69              displayMessage("Connection " + counter + "
                    received from : " + connection.
                    getInetAddress().getHostName());
70      }
71      private void getStreams() throws IOException
72      {
73              output = new ObjectOutputStream(connection.
                    getOutputStream());
74              output.flush();
75              input = new ObjectInputStream(connection.
                    getInputStream());
76              displayMessage("\nGot I/O streams\n");
77      }
78      private void processConnection() throws IOException
79      {
80              String message = "Connection successful";
81              sendData(message);
82              setTextFieldEditable(true);
83              do
84              {
85                      try
86                      {
87                              message = (String)input.
                                    readObject();
88                              displayMessage("\n" +
                                    message);
89                      }
90                      catch(ClassNotFoundException
                            classNotFoundException)
91                      {
92                              displayMessage("\nUnknown
                                    object type received");
93                      }
94              } while (!message.equals("CLIENT >>>
                    TERMINATE"));
95      }
96      private void closeConnection()
97      {
```

```
98                displayMessage("\nTerminating connection\n")
                      ;
99                setTextFieldEditable(false);
100               try
101               {
102                       output.close();
103                       input.close();
104                       connection.close();
105               }
106               catch(IOException ioException)
107               {
108                       ioException.printStackTrace();
109               }
110       }
111       private void sendData(String message)
112       {
113               try
114               {
115                       output.writeObject("SERVER >>> " +
                             message);
116                       output.flush();
117                       displayMessage("\nSERVER >>> " +
                             message);
118               }
119               catch (IOException ioException)
120               {
121                       displayArea.append("\nError writing
                             object");
122               }
123       }
124       private void displayMessage(final String
             messageToDisplay)
125       {
126               SwingUtilities.invokeLater(new Runnable()
127               {
128                       public void run()
129                       {
130                               displayArea.append(
                                     messageToDisplay);
131                               displayArea.setCaretPosition
                                     (displayArea.getText().
                                     length());
132                       }
133               }
134               );
```

```
135         }
136         private void setTextFieldEditable(final boolean
                editable)
137         {
138                 SwingUtilities.invokeLater(new Runnable()
139                 {
140                         public void run()
141                         {
142                                 enterField.setEditable(
                                        editable);
143                         }
144                 }
145                 );
146         }
147         public static void main(String args[])
148         {
149                 ServerApp application = new ServerApp();
150                 application.setDefaultCloseOperation(JFrame.
                        EXIT_ON_CLOSE);
151                 application.runServer();
152         }
153 }
```

```
 1 import java.awt.event.*;
 2 import java.io.*;
 3 import java.net.*;
 4 //The Customer class needs to implement Serializable
 5 class Customer implements Serializable
 6 {
 7         String custName;
 8         String custPassword;
 9 }
10
11 public class AppServer extends Thread
12 {
13         ServerSocket serverSocket;
14         public AppServer()
15         {
16         try
17         {
18             serverSocket = new ServerSocket(1001);
19         }
20         catch(IOException e)
21         {
22             fail(e, "Could not start server.");
```

```
23          }
24          System.out.println("Server started . . .");
25          this.start();     // Starts the thread
26
27          }
28            public static void fail(Exception e, String str)
29            {
30                      System.out.println(str + "." + e);
31            }
32
33              public void run()
34          {
35          try
36          {
37              while(true)
38              {
39                  Socket client = serverSocket.accept();
40                  Connection con = new Connection(client);
41              }
42          }
43          catch(IOException e)
44          {
45              fail(e, "Not listening");
46          }
47          }
48              public static void main(String args[])
49          {
50                 new AppServer();
51          }
52
53  }
54  class Connection extends Thread
55  {
56    protected Socket netClient;
57    protected ObjectInputStream fromClient;
58    protected PrintStream toClient;
59    public Connection(Socket client)
60    {
61          netClient = client;
62          try
63          {
64                              fromClient = new ObjectInputStream(
                                    netClient.getInputStream());
65                              toClient = new PrintStream(netClient
                                    .getOutputStream());
```

```
66          }
67          catch(IOException e)
68           {
69              try
70                {
71                  netClient.close();
72                }
73                catch(IOException e1)
74                {
75                  System.err.println("Unable to set up streams"
                        + e1);
76                  return;
77                }
78           }
79           this.start();
80      }
81      public void run()
82      {
83         Customer clientMessage;
84         try
85         {
86             for(;;)
87             {
88                clientMessage = (Customer)fromClient.readObject
                    ();
89                if(clientMessage == null)
90                        break;
91                // Send acknowledgement to the client
92                            toClient.println("Received from : "
                                + clientMessage.custName);
93                RandomAccessFile fobj=new RandomAccessFile("E:\\
                    Customer.txt","rw");
94                            fobj.seek(fobj.length());
95                            fobj.writeBytes(clientMessage.
                                custName + " : " + clientMessage
                                .custPassword + " ");
96
97                if (!(clientMessage.custName.equals(
                    clientMessage.custPassword)))
98                                    toClient.println("Received
                                        from:" + clientMessage.
                                        custName);
99                else
```

```
100                                     toClient.println("Name and
                                            password can not be same
                                            ");
101
102                 }
103             }
104             catch(IOException e)
105             {}
106             catch(ClassNotFoundException e1)
107              {
108                System.out.println("Error in reading object "+e1);
109              }
110             finally
111             {
112               try
113                {
114                  netClient.close();
115                }
116                catch(IOException e)
117                        {}
118                }
119                }
120         }

        import javax.swing.*;
        import java.awt.event.*;
        import java.io.*;
        import java.net.*;
        //The User class needs to implement Serializable
        class User extends Object implements Serializable
        {
                String userName;
                String password=new String();
                String dob=new String();
                String subj=new String();
        }
        public class AppServer1 extends Thread
        {
               ServerSocket serverSocket;
               public AppServer1()
               {
               try
               {
                   serverSocket = new ServerSocket(1001);
               }
```

```
        catch(IOException e)
        {
            fail(e,"Could not start server.");
        }
        System.out.println("Server started . . .");
        this.start();     // Starts the thread

        }
 public static void fail(Exception e, String str)
 {
        System.out.println(str + "." + e);
 }
        public void run()
        {
        try
        {
            while(true)
            {
                Socket client = serverSocket.accept();
                Connection con = new Connection(client);
            }
        }
        catch(IOException e)
        {
            fail(e,"Not listening");
        }
        }
        public static void main(String args[])
        {
        new AppServer();
        }

}
class Connection extends Thread
{
  protected Socket netClient;
  protected ObjectInputStream fromClient;
  protected PrintStream toClient;
  public Connection(Socket client)
  {
        netClient = client;
        try
        {
           fromClient = new ObjectInputStream(netClient.getInputStream());
               toClient = new PrintStream(netClient.getOutputStream());
```

```
        }
        catch(IOException e)
         {
            try
              {
                netClient.close();
              }
              catch(IOException e1)
              {
                System.err.println("Unable to set up streams" + e1);
                return;
              }
         }
         this.start();
   }
   public void run()
   {
      String clientMessage;
      try
      {
          for(;;)
          {
              User clientdet = (User)fromClient.readObject();
              if(clientdet == null)
              break;
              RandomAccessFile fobj=new RandomAccessFile("User.txt","rw");
              fobj.seek(fobj.length());
              fobj.writeBytes(clientdet.userName+":"+clientdet.password+":"+clientd
              toClient.println(clientdet.userName+"@cellgo.com");
          }
      }
      catch(Exception e)
      {}
      finally
      {
        try
         {
           netClient.close();
         }
         catch(IOException e)
    {}
    }
 }
}
```

```
import java.awt.event.*;
import java.io.*;
import java.net.*;
class Login implements Serializable
{
        String loginName;
        String loginPassword;
}

public class LoginServer extends Thread
{
        ServerSocket serverSocket;

        public LoginServer()
    {
                try
                {
                        serverSocket = new ServerSocket(1001);
                }
                catch(IOException e)
                {
                        fail(e, "Could not start server.");
                }
                System.out.println("Server started . . .");
                this.start();     // Starts the thread
    }
        public static void fail(Exception e, String str)
        {
                System.out.println(str + "." + e);
        }

        public void run()
    {
       try
       {
           while(true)
           {
               Socket client = serverSocket.accept();
               Connection con = new Connection(client);
           }
       }
       catch(IOException e)
       {
           fail(e, "Not listening");
       }
```

```
        }

        public static void main(String args[])
    {
                new LoginServer();
    }
}
class Connection extends Thread
{
        protected Socket netClient;
        protected ObjectInputStream fromClient;
        protected PrintStream toClient;
        public Connection(Socket client)
        {
                netClient = client;
                try
                {
                        fromClient = new ObjectInputStream(netClient.getInputStrea
                        toClient = new PrintStream(netClient.getOutputStream());
                }
                catch(IOException e)
        {
                        try
                        {
                                netClient.close();
            }
            catch(IOException e1)
            {
                                System.err.println("Unable to set up streams" + e
                                return;
            }
        }
        this.start();
        }
        public void run()
        {
                Login clientMessage;
                try
                {
                        for(;;)
                        {
                                clientMessage = (Login)fromClient.readObject();
                                if(clientMessage == null)
                                        break;
                                toClient.println("Received from : " + clientMessag
```

```
/*
                                RandomAccessFile fobj=new RandomAccessFile("Login
                                fobj.seek(fobj.length());
                                fobj.writeBytes(clientMessage.loginName + " : " +

                                if(!(clientMessage.loginName.equals(clientMessage
                                        toClient.println("Received from : " + clie
                                else
                                        toClient.println("Name and password can n
*/
                        }
                }
                catch(IOException e)
                {}
                catch(ClassNotFoundException e1)
                {
                        System.out.println("Error in reading object "+e1);
                }
            finally
                {
                        try
                        {
                                netClient.close();
                        }
                        catch(IOException e)
                        {}
                }
        }
}

import java.awt.*;
import java.io.*;
import javax.swing.*;
public class DirList extends JFrame
{
        JTextArea ta;
        String path;
        public static void main(String args[])
        {

                DirList listObj=new DirList(args[0]);

                listObj.setVisible(true);
                listObj.setSize(300,300);
        }
```

```
        public DirList(String fileName)
        {
                super("Directory List");
                path=fileName;
                File fileObj=new File(path);
                ta=new JTextArea(10,10);
                getContentPane().add(ta,BorderLayout.CENTER);

                if(fileObj.isDirectory())
                        displayList(fileObj);
                else{ if(fileObj.isFile())
                        {
                         try{
                            FileInputStream inputFile = new FileInputStream(fileO
                            int i=inputFile.available();
                            byte b;
                            for(;i>0;i--){
                                b=(byte)inputFile.read();
                                System.out.println(b);
                                }
                            System.exit(0);
                             }catch(IOException e){
                                   System.out.println("Error reading file");
                                   }
                        }
                 }
        }
        void displayList(File fobj)
        {
                String dirContents[]=fobj.list();
                ta.append("--------------");
                for(int i=0;i<dirContents.length;i++)
                {

                ta.append(dirContents[i]);
                ta.append(" , ");
                }


        }
}

import java.io.*;
import java.net.*;
import java.awt.*;
```

```
import java.awt.event.*;
import javax.swing.*;
public class ClientApp extends JFrame
{
        private JTextField enterField;
        private JTextArea displayArea;
        private ObjectOutputStream output;
        private ObjectInputStream input;
        private String message = "";
        private String chatServer;
        private Socket client;
        public ClientApp(String host)
        {
                super("Client");
                chatServer = host;
                Container container = getContentPane();
                enterField = new JTextField();
                enterField.setEditable(false);
                enterField.addActionListener( new ActionListener()
                {
                        public void actionPerformed(ActionEvent event)
                        {
                                sendData(event.getActionCommand());
                                enterField.setText("");
                        }
                }
                );
                container.add(enterField, BorderLayout.NORTH);
                displayArea = new JTextArea();
                container.add(new JScrollPane(displayArea), BorderLayout.CENTER);
                setSize(300,150);
                setVisible(true);
        }
        private void runClient()
        {
                try
                {
                        connectToServer();
                        getStreams();
                        processConnection();
                }
                catch(EOFException eofException)
                {
                        System.err.println("Client terminated connection");
                }
```

```
        catch(IOException ioException)
        {
                ioException.printStackTrace();
        }
        finally
        {
                closeConnection();
        }
}
private void connectToServer() throws IOException
{
        displayMessage("Attempting connection\n");
        client = new Socket(InetAddress.getByName(chatServer), 12345);
        displayMessage("Connected to : " + client.getInetAddress().getHost
}
private void getStreams() throws IOException
{
        output = new ObjectOutputStream(client.getOutputStream());
        output.flush();
        input = new ObjectInputStream(client.getInputStream());
        displayMessage("\nGot I/O streams\n");
}
private void processConnection() throws IOException
{
        setTextFieldEditable(true);
        do
        {
                try
                {
                        message = (String)input.readObject();
                        displayMessage("\n" + message);
                }
                catch(ClassNotFoundException classNotFoundException)
                {
                        displayMessage("\nUnknown object type received");
                }
        } while(!message.equals("SERVER >>> TERMINATE"));
}
private void closeConnection()
{
        displayMessage("\nClosing connection");
        setTextFieldEditable(false);
        try
        {
                output.close();
```

```
                input.close();
                client.close();
        }
        catch(IOException ioException)
        {
                ioException.printStackTrace();
        }
}
private void sendData(String message)
{
        try
        {
                output.writeObject("CLIENT >>> " + message);
                output.flush();
                displayMessage("\nCLIENT >>> " + message);
        }
        catch(IOException ioException)
        {
                displayArea.append("\nError writing object");
        }
}
private void displayMessage(final String messageToDisplay)
{
        SwingUtilities.invokeLater(new Runnable()
        {
                public void run()
                {
                        displayArea.append(messageToDisplay);
                        displayArea.setCaretPosition(displayArea.getText(
                }
        }
        );
}
private void setTextFieldEditable(final boolean editable)
{
        SwingUtilities.invokeLater( new Runnable()
        {
                public void run()
                {
                        enterField.setEditable(editable);
                }
        }
        );
}
```

```
        public static void main(String args[])
        {
                ClientApp application;
                if (args.length == 0)
                        application = new ClientApp("127.0.0.1");
                else
                        application = new ClientApp(args[0]);
                application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                application.runClient();
        }
}

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
class Customer extends Object implements java.io.Serializable
{
     String custName;
     String custPassword;
}
public class CustomerApplet extends JApplet
{
        JPanel panelObject;
        JLabel labelCustName;
        JLabel labelCustPassword;
        JTextField textCustName;
        JPasswordField textCustPassword;
    JButton buttonLogin;
        GridBagLayout gl;
        GridBagConstraints gbc;
        public void init()
        {
                gl = new GridBagLayout();
                gbc = new GridBagConstraints();
                panelObject = (JPanel)getContentPane();
                panelObject.setLayout(gl);

                labelCustName = new JLabel("Customer Login Name");
                labelCustPassword = new JLabel("Password");
                textCustName = new JTextField(15);
        textCustPassword = new JPasswordField(15);
        buttonLogin=new JButton("Login");
```

```
            gbc.anchor = GridBagConstraints.NORTHWEST;
            gbc.gridx = 1;
            gbc.gridy = 5;
            gl.setConstraints(labelCustName,gbc);
            panelObject.add(labelCustName);

            gbc.anchor = GridBagConstraints.NORTHWEST;
            gbc.gridx = 4;
            gbc.gridy = 5;
            gl.setConstraints(textCustName,gbc);
            panelObject.add(textCustName);
            gbc.anchor = GridBagConstraints.NORTHWEST;
            gbc.gridx = 1;
            gbc.gridy = 9;
            gl.setConstraints(labelCustPassword,gbc);
            panelObject.add(labelCustPassword);
            gbc.anchor = GridBagConstraints.NORTHWEST;
            gbc.gridx = 4;
            gbc.gridy = 9;
            gl.setConstraints(textCustPassword,gbc);
            panelObject.add(textCustPassword);
    gbc.anchor=GridBagConstraints.NORTHWEST;
    gbc.gridx=2;
    gbc.gridy=13;
    gl.setConstraints(buttonLogin,gbc);
    panelObject.add(buttonLogin);

    LoginAction loginrequest=new LoginAction();
    buttonLogin.addActionListener(loginrequest);
    }
class LoginAction implements ActionListener
{
            public void actionPerformed(ActionEvent evt)
            {
                    Object obj=evt.getSource();
        if(obj==buttonLogin)
        {
           Customer data=new Customer();
                    data.custName = textCustName.getText();
                    data.custPassword =new String(textCustPassword.getPassv
                    try
                    {
                                Socket toServer;
                                toServer = new Socket("127.0.0.1",1001);
                                ObjectOutputStream streamToServer=new
```

```
                                        ObjectOutputStream (toServer.getOutputStre
streamToServer.writeObject((Customer)data);

                                        BufferedReader fromServer = new BufferedR
                                        String status = fromServer.readLine();
                                        getAppletContext().showStatus(status);
                                        streamToServer.close();
                                        fromServer.close();

                                    }
                                catch(InvalidClassException e)
                                {
                                     showStatus("The Customer class is invalid" + e);
                                }
                                catch(NotSerializableException e)
                                {
                                     showStatus("The object is not serializable" + e);
                                }
                                catch(IOException e)
                                {
                                     showStatus("Cannot write to the server" + e);
                                }
                            }
                    }
    }
}

//DealerApplet.java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
class Dealer
{
     String dealerName;
     String dealerPassword;
}
public class DealerApplet extends JApplet
{
        JPanel panelObject;
        JLabel labelDealerName;
        JLabel labelDealerPassword;
        JTextField textDealerName;
        JPasswordField textDealerPassword;
```

```
JButton buttonLogin;
  GridBagLayout gl;
  GridBagConstraints gbc;
  public void init()
  {
          //Initialize the layout variables
          gl = new GridBagLayout();
          gbc = new GridBagConstraints();
          panelObject = (JPanel)getContentPane();
          panelObject.setLayout(gl);

      //Initialize controls
          labelDealerName = new JLabel("Dealer Login Name");
          labelDealerPassword = new JLabel("Password");
          textDealerName = new JTextField(15);
      textDealerPassword = new JPasswordField(15);
      buttonLogin=new JButton("Login");

          //Add controls to the panel
          gbc.anchor = GridBagConstraints.NORTHWEST;
          gbc.gridx = 1;
          gbc.gridy = 5;
          gl.setConstraints(labelDealerName,gbc);
          panelObject.add(labelDealerName);

          gbc.anchor = GridBagConstraints.NORTHWEST;
          gbc.gridx = 4;
          gbc.gridy = 5;
          gl.setConstraints(textDealerName,gbc);
          panelObject.add(textDealerName);
          gbc.anchor = GridBagConstraints.NORTHWEST;
          gbc.gridx = 1;
          gbc.gridy = 9;
          gl.setConstraints(labelDealerPassword,gbc);
          panelObject.add(labelDealerPassword);
          gbc.anchor = GridBagConstraints.NORTHWEST;
          gbc.gridx = 4;
          gbc.gridy = 9;
          gl.setConstraints(textDealerPassword,gbc);
          panelObject.add(textDealerPassword);
      gbc.anchor=GridBagConstraints.NORTHWEST;
      gbc.gridx=2;
      gbc.gridy=13;
      gl.setConstraints(buttonLogin,gbc);
      panelObject.add(buttonLogin);
```

```
            LoginAction loginrequest=new LoginAction();
            buttonLogin.addActionListener(loginrequest);
        }
      class LoginAction implements ActionListener
      {
            public void actionPerformed(ActionEvent evt)
            {
                  Object obj=evt.getSource();
                  if(obj==buttonLogin)
                  {
                  }
            }
       }
}

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
class Customer extends Object implements java.io.
   Serializable
{
     String custName;
     String custPassword;
}
public class CustomerApplet extends JApplet
{
        JPanel panelObject;
        JLabel labelCustName;
        JLabel labelCustPassword;
        JTextField textCustName;
        JPasswordField textCustPassword;
      JButton buttonLogin;
        GridBagLayout gl;
        GridBagConstraints gbc;
        public void init()
        {
                //Initialize the layout variables
                gl = new GridBagLayout();
                gbc = new GridBagConstraints();
                panelObject = (JPanel)getContentPane();
                panelObject.setLayout(gl);
```

```
        //Initialize controls
            labelCustName = new JLabel("Customer Login
                Name");
            labelCustPassword = new JLabel("Password");
            textCustName = new JTextField(15);
          textCustPassword = new JPasswordField(15);
          buttonLogin=new JButton("Login");

            //Add controls to the panel
            gbc.anchor = GridBagConstraints.NORTHWEST;
            gbc.gridx = 1;
            gbc.gridy = 5;
            gl.setConstraints(labelCustName,gbc);
            panelObject.add(labelCustName);

            gbc.anchor = GridBagConstraints.NORTHWEST;
            gbc.gridx = 4;
            gbc.gridy = 5;
            gl.setConstraints(textCustName,gbc);
            panelObject.add(textCustName);
            gbc.anchor = GridBagConstraints.NORTHWEST;
            gbc.gridx = 1;
            gbc.gridy = 9;
            gl.setConstraints(labelCustPassword,gbc);
            panelObject.add(labelCustPassword);
            gbc.anchor = GridBagConstraints.NORTHWEST;
            gbc.gridx = 4;
            gbc.gridy = 9;
            gl.setConstraints(textCustPassword,gbc);
            panelObject.add(textCustPassword);
          gbc.anchor=GridBagConstraints.NORTHWEST;
          gbc.gridx=2;
          gbc.gridy=13;
          gl.setConstraints(buttonLogin,gbc);
          panelObject.add(buttonLogin);

          LoginAction loginrequest=new LoginAction();
          buttonLogin.addActionListener(loginrequest);
     }
class LoginAction implements ActionListener
{
   public void actionPerformed(ActionEvent evt)
   {
          Object obj=evt.getSource();
          if(obj==buttonLogin)
```

```
{
    Customer data=new Customer();
        data.custName = textCustName.getText();
        data.custPassword =new String(
            textCustPassword.getPassword());
        try
        {
                    Socket toServer;
                    toServer = new Socket
                        ("192.1.1.150",1001);
                    ObjectOutputStream
                        streamToServer=new
     ObjectOutputStream (toServer.getOutputStream
        ());

        streamToServer.writeObject((Customer)
        data);


     BufferedReader fromServer=new BufferedReader
        (new
        InputStreamReader(toServer.
        getInputStream()));
     String status=fromServer.readLine();
     getAppletContext().showStatus(status);
                    streamToServer.close();
     fromServer.close();

    }
        catch(InvalidClassException e)
        {
             showStatus("The Customer class is
                invalid" + e);
        }
        catch(NotSerializableException e)
        {
             showStatus("The object is not
                serializable" + e);
        }
        catch(IOException e)
        {
             showStatus("Cannot write to the
                server" + e);
        }
  }
```

```
        }
    }
}

import java.io.*;
import java.net.*;
class Bill
{
      static int billno=0;
      public synchronized int genBillNo()
      {
            billno++;
            return billno;
      }
}
public class BillServer extends Thread
{
          ServerSocket ss;
            public BillServer()
          {
                try { ss=new ServerSocket(1001); }catch(
                   IOException e){}
                this.start();
          }
          public void run()
          {
                    while(true){
                          try {
                          Socket s=ss.accept();
                    ClientThread clientthread=new
                       ClientThread(s); }
                          catch(IOException e){}
                          }
          }
            public static void main(String args[])
            {
                        new BillServer();
            }
}
class ClientThread extends Thread
{
            protected Socket netClient;
          PrintStream ps;
          int bno;
          public ClientThread(Socket s)
```

```
        {
                netClient=s;
            try{ps=new PrintStream(s.getOutputStream())
                ;}catch(IOException e){}
                this.start();
          }
          public void run()
        {
            Bill obj=new Bill();
            bno=obj.genBillNo();
            try{ps.println(String.valueOf(bno));}catch(
                Exception e){}
        }
}

import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
//The Message class needs to implement serializable
class Customer implements java.io.Serializable
{
        String custName;
        String custPassword;
}
//Code for the AppServer class
public class AppServer implements Runnable
{
        ServerSocket server;
        Socket fromClient;
        Thread serverThread;

      public AppServer()
        {

            try
                {
                        server = new ServerSocket(1001);
                        serverThread = new Thread(this);
                        serverThread.start();
                }
                catch(Exception e)
                {
                        System.out.println("Cannot start the
                            thread" + e);
```

```
                }
        }

        public static void main(String args[])
        {
                new AppServer();
        }
        public void run()
        {
                try
                {
                        while(true)
                        {
                                //Listening to the clients
                                    request
                                fromClient = server.accept()
                                    ;
                                //Creating the connect
                                    object
                                Connect con = new Connect(
                                    fromClient);
                        }
                }
                catch(Exception e)
                {
                   System.out.println("Cannot listen to the
                       client" + e);
                }
        }
}
//Code for the connect class
class Connect extends Thread
{
      PrintStream streamToClient;
        Customer data;
        ObjectInputStream streamFromClient;

      static Vector vector = new Vector(1,1);
      static int messageCount;//To count the total number of
          messages
                              //stored
      private int localMsgCount;// To count local messages
        public Connect(Socket inFromClient)
        {
                //Retrieving the clients stream
```

```
                try
                {
                        streamFromClient = new
                            ObjectInputStream(inFromClient.
                            getInputStream());
                        streamToClient= new PrintStream(
                            inFromClient.getOutputStream());



                }
                catch(Exception e)
                {
                System.out.println("Cannot get the client
                    stream" + e);
                }

                finally
                {
                  try
                  {
                  inFromClient.close();
                  }
                  catch(IOException e)
                  {}
                }
                this.start();


     }
     public void run()
      {
             try
                     {

data=(Customer)streamFromClient.readObject();
streamToClient.print(data.custName+" Connected");

for(;;)
   {
```

```
                                                    data=(
                                                        Customer
                                                        )
                                                        streamFromClient
                                                        .
                                                        readObject
                                                        ();
        writeMessage(data);//store message
        String message=readMessage();//read messages stored
        streamToClient.print(message);
        }
    }
    catch(InvalidClassException e)
    {
          System.out.println("Cannot serialize the applicant
              class" + e);
    }
    catch(NotSerializableException e)
    {
          System.out.println("The object is not serializable"
              + e);
    }
    catch(IOException e)
    {
          System.out.println("Cannot read from the client
              stream" + e);
    }
    catch(ClassNotFoundException e)
    {
        System.out.println("Customer class could not be found
            "+e);
    }

}
     //Method to store message
     synchronized void writeMessage(Customer cust)
     {
         vector.addElement(cust);
         ++messageCount;
         ++localMsgCount;
         notifyAll();
      }
      //Method to retrieve message
      synchronized String readMessage()
      {
```

```
        String str=" ";
        while(localMsgCount>=messageCount)
        {
            try
            {
              wait();
            }
            catch(InterruptedException e)
             {}
         }
         for(int i=localMsgCount;i<=messageCount; i++)
         {
                  str=str+vector.elementAt(i);
            }
            notifyAll();
            return str;
        }
}

import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
//The Message class needs to implement serializable
class Message implements java.io.Serializable
{
        String msgFrom;
        String msg;
        String msgTo;
}
//Code for the AppServer class
public class AppServer implements Runnable
{
        ServerSocket server;
        Socket fromClient;
        Thread serverThread;
    public AppServer()
        {
                try
                {
                        server = new ServerSocket(1004);
                        serverThread = new Thread(this);
                        serverThread.start();
                }
                catch(Exception e)
```

```
                {
                        System.out.println("Cannot start the
                            thread " + e);
                }
        }
        public static void main(String args[])
        {
                new AppServer();
        }
        public void run()
        {
                try
                {
                        while(true)
                        {
                                fromClient = server.accept()
                                    ;
                                Connect con = new Connect(
                                    fromClient);
                        }
                }
                catch(Exception e)
                {
                   System.out.println("Cannot listen to the
                       client" + e);
                }
        }
}
class Connect extends Thread
{
        PrintStream streamToClient;
        Message data;
    //To store user name
    String fromUser;
    Socket inFromClient;
        ObjectInputStream streamFromClient;
    static Vector vector = new Vector(1,1);
    static int messageCount;//To count the total number of
       messages stored
    private int localMsgCount;// To count local messages
        public Connect(Socket inFromClients)
        {
                inFromClient=inFromClients;
                //Retrieving the clients stream
                try
```

```
            {
                    streamFromClient = new
                        ObjectInputStream(inFromClient.
                        getInputStream());
            streamToClient= new PrintStream(inFromClient.
                getOutputStream());
            }
            catch(Exception e)
            {
                    System.out.println("Cannot get the
                        client stream" + e);
            }

            this.start();
    }
public void run()
{
            try
            {
                    data=(Message)streamFromClient.
                        readObject();
            //Check for valid login
            if(!(data.msgFrom.equals(data.msg))&&(data.msg.
                length()>6&&data.msg.length()<10))
            {
                            String str="Aaron"+":"+"
                                Malcolm"+":"+"Michelle
                                "+"\n";
                streamToClient.print(str);//Send the list of
                     sales executives
                fromUser=data.msgFrom;
            }
            else
                    {
                            streamToClient.print("Check
                                Password");
                inFromClient.close();
                    }
            for(;;)
                    {
                            data=(Message)
                                streamFromClient.
                                readObject();
                            vector.addElement((Message)
                                data);
```

```
            messageCount++;
                            if(localMsgCount<=
                                messageCount){
            Message data1=(Message)vector.elementAt(
                localMsgCount);
            //If message is for this user send it
                accross
            if(data1.msgTo.equals(fromUser))
            {
                                        Message data2=(
                                            Message)vector.
                                            elementAt(
                                            localMsgCount);
                                        streamToClient.print
                                            (data2.msgFrom
                                            +":"+data2.msg
                                            +"\n");
            }
            streamToClient.print("No new message"+"\n");
            localMsgCount++;
        }
            }
    }
        catch(InvalidClassException e)
        {
                    System.out.println("Cannot serialize
                        the applicant class" + e);
          }
          catch(NotSerializableException e)
          {
                    System.out.println("The object is
                        not serializable" + e);
          }
          catch(IOException e)
          {
                    System.out.println("Cannot read from
                        the client stream" + e);
          }
          catch(ClassNotFoundException e)
          {
                  System.out.println("Dealer class could
                      not be found"+e);
          }

     }
```

```
}

import java.util.*;
import java.io.*;
class Message
{
        String from;
        String message;
        public Message(String from, String message)
        {
                this.from = from;
                this.message = message;
        }
}
class MyVector extends Vector
{
        BufferedReader standardInput;
        static int counter;
        int instanceCounter;
        public MyVector()
        {
                super(1,1);
                standardInput = new BufferedReader(new
                   InputStreamReader(System.in));
        }
        synchronized void put()
        {
                System.out.println("Please enter your name
                   and the message : ");
                String from = standardInput.read();
                String message = standardInput.read();
                Message msg = new Message(from, message);
                addElement((Message)msg);
                counter++;
                instanceCounter++;
                notify();
        }
        synchronized void get()
        {
                while (instanceCounter >= counter)
                {
                        try
                        {
                                wait();
```

```
                        }
                        catch(InterruptedException e)
                        {
                                System.out.println("Thread
                                    interrupted " + e);
                        }
                }
                Message msg = (Message)elementAt(counter-1);
                System.out.println(msg.from + ":" + msg.
                    message);
                counter++;
                notify();
        }
}
class MyThread1 implements Runnable
{
        MyVector myVector;
        public MyThread1(MyVector myVector)
        {
                this.myVector = myVector;
                new Thread(this).start();
        }
        public void run()
        {
                while(true)
                {
                        myVector.put();
                }
        }
}
class MyThread2 implements Runnable
{
        MyVector myVector;
        public MyThread2(MyVector myVector)
        {
                this.myVector = myVector;
                new Thread(this).start();
        }
        public void run()
        {
                while(true)
                {
                        myVector.get();
                }
        }
```

```
        public static void main(String args[])
        {
                MyVector myVector = new MyVector();
                new MyThread1(myVector);
                new MyThread2(myVector);
        }
}


import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
//Code for the AppServer class
public class AppServerUGP implements Runnable
{
        ServerSocket server;
        Socket fromClient;
        Thread serverThread;
        public AppServerUGP()
        {
                try
                {
                        server = new ServerSocket(1004);
                        serverThread = new Thread(this);
                        serverThread.start();
                }
                catch(Exception e)
                {
                        System.out.println("Cannot start the
                            thread " + e);
                }
        }
        public static void main(String args[])
        {
                new AppServerUGP();
        }
        public void run()
        {
                try
                {
                        while(true)
                        {
                                fromClient = server.accept()
                                    ;
```

```
                                Connect con = new Connect(
                                    fromClient);
                            }
                        }
                        catch(Exception e)
                        {
                            System.out.println("Cannot listen to the
                                client" + e);
                        }
                }
}
class Connect extends Thread
{
        ObjectOutputStream streamToClient;
        Login data;

        Socket inFromClient;
        ObjectInputStream streamFromClient;
        static Vector vector = new Vector(1,1);

        public Connect(Socket inFromClients)
        {
            inFromClient=inFromClients;
            //Retrieving the clients stream
            try
            {
            streamFromClient = new ObjectInputStream(
                inFromClient.getInputStream());
            streamToClient= new ObjectOutputStream(
                inFromClient.getOutputStream());
            data=(Login)streamFromClient.readObject();
            vector.addElement((String)data.name);
            streamToClient.writeObject((String)"Valid");
            //Write all users who are online
            for(int i=0;i<vector.size();i++)
            streamToClient.writeObject((String)vector.
                elementAt(i));
            //Inform client that there are no more
                messages
            streamToClient.writeObject((String)"Over");
            }
            catch(Exception e)
            {
            System.out.println("Cannot get the client
                stream" + e);
```

```
                }

        }


}

import java.util.*;
public class VectorTest extends Vector
{
        public VectorTest()
        {
                super(1,1);
        }
        public void addInteger(int integer)
        {
                addElement(new Integer(integer));
        }
        public void addString(String str)
        {
                addElement(str);
        }
        public void displayVector()
        {
                Object object;
                int length = size();
                System.out.println("Number of Vector
                    elements : " + length);
                System.out.println("They are : ");
                for (int i = 0; i < length; i++)
                {
                        object = elementAt(i);
                        System.out.println(object.toString()
                            );
                }
        }
        public static void main(String args[])
        {
                VectorTest vector = new VectorTest();
                int num = 100;
                String string = new String("I love Java");
                vector.addInteger(num);
                vector.addString(string);
                vector.displayVector();
        }
```

}

Pustaka

[1] Sun Java Tutorials
[2] Bruce Eckel, Thinking in Java - 2nd Edition, Prentice Hall, 2000