Pemrograman Berorientasi Objek
Konstruktor dan Destruktor

Adam Mukharil Bachtiar

Teknik Informatika UNIKOM

Konstruktor dan Destruktor

1. Definisi konstruktor
2. Inisialisasi pada konstruktor
3. Overloading konstruktor
4. Default value
5. Destruktor
6. Alokasi memori dinamis

KONSTRUKTOR

Definisi Konstruktor

1. FUNGSI KHUSUS YANG NAMANYA SAMA DENGAN NAMA CLASS.
2. DIGUNAKAN UNTUK INISIALISASI DAN PEMBERIAN DEFAULT VALUE.
3. DIPANGGIL OTOMATIS KETIKA INSTANSIASI OBJEK.

Definisi Konstruktor

4 TiDAK MENGEMBALiKAN NiLAi.

5 KONSTRUKTOR DAPAT DiOVERLOADiNG.

Implementasi Konstruktor di C++

```
class titik{
  private:
    int x;
    int y;
  public:
    titik(){
      cout<<"Saya konstruktor!"<<endl;
    }
};
```

Implementasi Konstruktor di C++

```
main()
{
        titik a;//kostruktor akan dijalankan
        system("pause");
        return 0;
}
```

Implementasi Konstruktor di Java

```
public class Titik{

   private int x;

   private int y;


   public Titik(){

       System.out.println("Saya konstruktor!");

   }

}
```

Implementasi Konstruktor di Java

```
public class TesTitik {

   public static void main(String[] args) {

        Titik a=new Titik();

    }

}
```

Inisialisasi Dengan Konstruktor

KONSTRUKTOR DAPAT DIGUNAKAN UNTUK MEMBERIKAN NILAI AWAL PADA SUATU ATRIBUT PADA CLASS TERSEBUT. PROSES INISIALISASI INI MIRIP SEPERTI SETTER.

Inisialisasi Dengan Konstruktor di C++

```
class titik{
   private:
      int x;
      int y;

   public:
      titik(int x,int y){
         this->x=x;
         this->y=y;
         cout<<"Konstruktor titik sedang dijalankan!”<<endl;
      }
};
```

INISIALISASI

Inisialisasi Dengan Konstruktor di C++

```
int main(int argc, char *argv[])
{
    titik a(10,10);
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Inisialisasi Dengan Konstruktor di Java

```
class titik{

    private int x;

    private int y;


    public titik(int x,int y){

      this.x=x;

      this.y=y;

       System.out.println("Konstruktor titik dijalankan);

    }

}
```

INISIALISASI

Inisialisasi Dengan Konstruktor di Java

```
public class TesTitik {

   public static void main(String[] args) {

        Titik a=new Titik(10,10);

    }

}
```

Overloading Konstruktor

1. Memiliki konsep yang sama dengan overloading method.
2. Dibedakan berdasarkan parameter (jumlah atau tipe datanya).
3. Pemanggilan konstruktor tergantung pada instansiasi objek.

Overloading Konstruktor di C++

```
class titik{
  private:
    int x;
    int y;

  public:
    titik(){
        cout<<"Konstruktor titik 1 dijalankan!"<<endl;
    }

    titik(int x,int y){
        this->x=x;
        this->y=y;
        cout<<"Konstruktor titik 2 sedang dijalankan!"<<endl;
    }
};
```

Overloading Konstruktor di C++

```
int main(int argc, char *argv[])
{
    titik a;
    cout<<endl;
    titik b(10,10);
    cout<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Overloading Konstruktor di Java

```
public class Titik {
    private int x;
    private int y;

    public Titik() {
        System.out.println("Konstruktor titik 1 dijalankan!");
    }

    public Titik(int x, int y) {
        this.x = x;
        this.y = y;
        System.out.println("Konstruktor titik 2 dijalankan!");
    }
}
```

Overloading Konstruktor di Java

```
public class TesTitik {
   public static void main(String[] args) {
        Titik a=new Titik();
        Titik b=new Titik(10,10);
    }
}
```

Default Value Konstruktor

PARAMETER YANG DIGUNAKAN PADA KONSTRUKTOR DAPAT DIBERIKAN NILAI DEFAULT YANG AKAN DIGUNAKAN JIKA PENGGUNANYA TIDAK MENGISI PARAMETERNYA.

Default Value di C++

```
class titik{
      private:
               int x;
               int y;

      public:
        titik(int x=0,int y=0){
         cout<<"Konstruktor titik dijalankan!”<<endl;
        }
};
```

DEFAULT VALUE

Default Value di C++

```
int main(int argc, char *argv[])
{
    titik a;
    cout<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Default Value di Java

```
class Titik{
    private int x;
    private int y;


    public titik(int x=0,int y=0){
        System.out.println("Saya Konstruktor!");
    }
};
```

Default Value di Java

```
public class TesTitik {

    public static void main(String[] args){

          Titik a=new Titik();

    }

}
```

DESTRUKTOR

Definisi Destruktor

1. FUNGSI KHUSUS YANG NAMANYA SAMA DENGAN NAMA CLASS DAN DIAWALI " ~ ".
2. DIGUNAKAN SEBAGAI PENANDA KETIKA OBJEK DIHAPUS DARI MEMORI.
3. DIPANGGIL OTOMATIS KETIKA OBJEK DIHANCURKAN.

Definisi Destruktor

4. TIDAK MENGEMBALIKAN NILAI DAN TIDAK BERPARAMETER.

5. TIDAK DAPAT DIOVERLOADING.

6. HANYA TERLIHAT JIKA MENGGUNAKAN ALOKASI MEMORI DINAMIS.

Definisi Destruktor

7

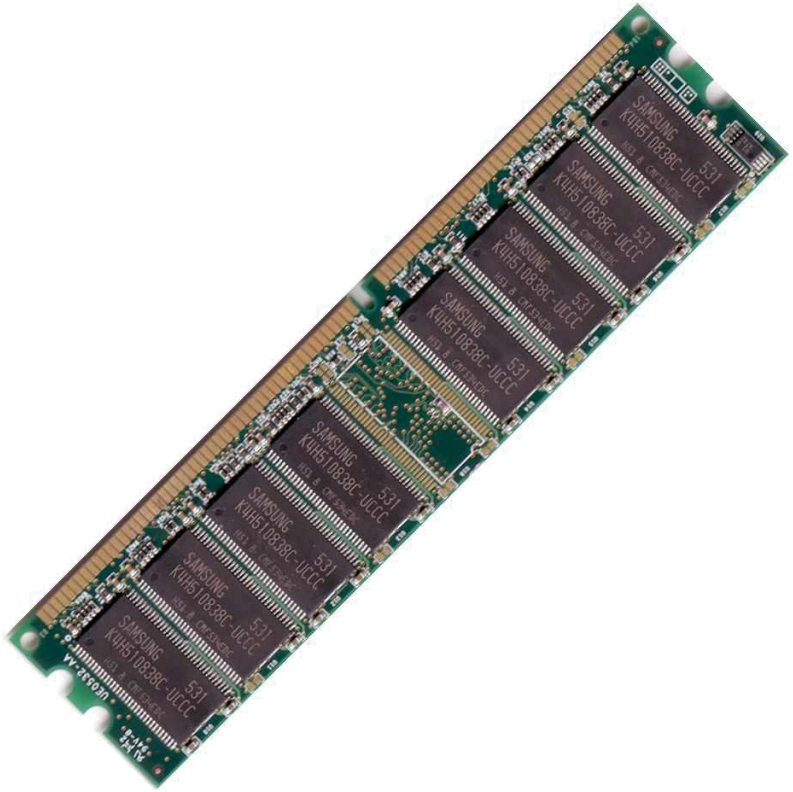
JAVA MEMILIKI FASILITAS GARBAGE COLLECTION SEHINGGA TIDAK PERLU MEMBUAT DESTRUKTOR SECARA EKSPLISIT.

Implementasi Destruktor di C++

```
class titik{
      private:
               int x;
               int y;

      public:
       ~titik(){
          cout<<"Destruktor titik dijalankan!";
        }
};
```

ALOKASI MEMORI DINAMIS



Alokasi Memori Dinamis

CARA INSTANSIASI OBJEK YANG MEMUNGKINKAN PENCIPTAAN DAN PENGHANCURAN OBJEK PADA SAAT PROGRAM MASIH DIJALANKAN.

Format Alokasi Memori Dinamis C++

Format Pemesanan Memori:

<nama_class> <nama_objek>;

<nama_object>=new <nama_class>;

Contoh:

Dosen Dsn1;

Dsn1=new Dosen();

Format Alokasi Memori Dinamis C++

Format Penghapusan Objek:

delete(<nama_objek>);

Contoh:

delete(dsn1);//destruktor akan terlihat di sini