Pemrograman Berorientasi Objek
Pengantar OOP

Adam Mukharil Bachtiar

Teknik Informatika UNIKOM

Deskripsi Mata Kuliah

1. Sifat : Wajib
2. SKS : 3 SKS Teori + Uji Coba

   2 SKS Homework
3. Prasyarat : Algoritma 1 dan 2 + Struktur Data

Silabus Mata Kuliah

1. Pengantar OOP
2. ADT
3. Class 1
4. Class 2
5. Konstruktor dan Destruktor
6. Friend
7. Inheritance
8. Polimorphisme
9. Abstract Class
10. Interface
11. Package
12. Teknologi Java Lainnya.

Penilaian

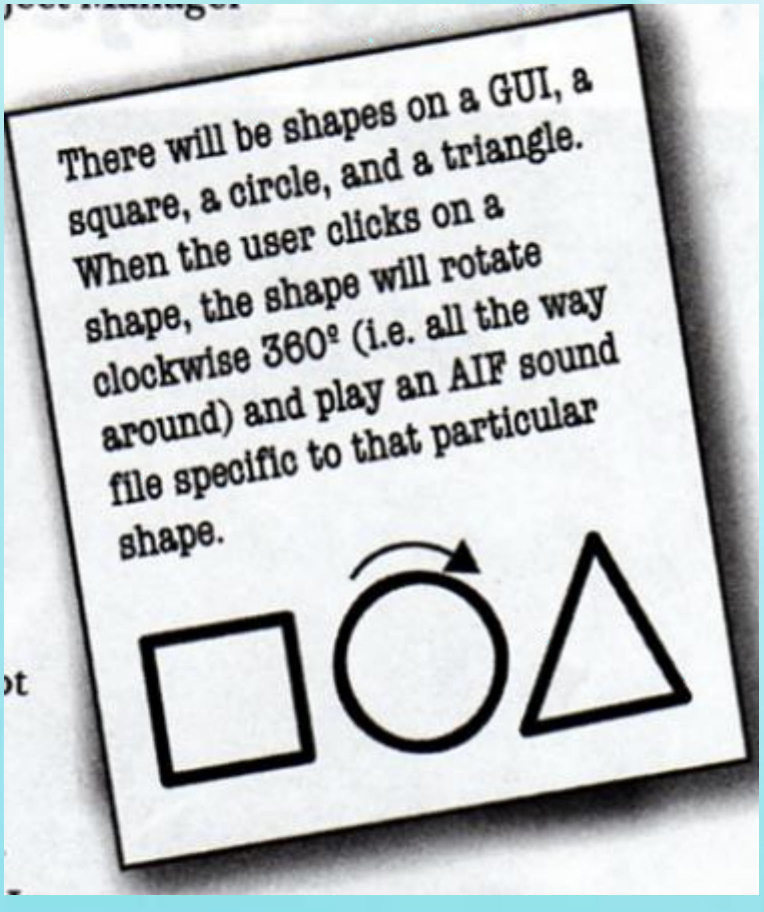
30 % (tugas+quiz) + 30% UTS + 40% UAS

INDEKS	NILAI
A	$80 \leq NA \leq 100$
B	$68 \leq NA \leq 79$
C	$56 \leq NA \leq 67$
D	$45 \leq NA \leq 55$
E	$0 \leq NA \leq 44$

Pengantar OOP

1. Why we need OOP?
2. The differences of procedural and OOP.
3. OO and OO System.
4. Tools

Why we need OOP?


There will be shapes on a GUI, a square, a circle, and a triangle. When the user clicks on a shape, the shape will rotate clockwise 360º (i.e. all the way around) and play an AIF sound file specific to that particular shape.

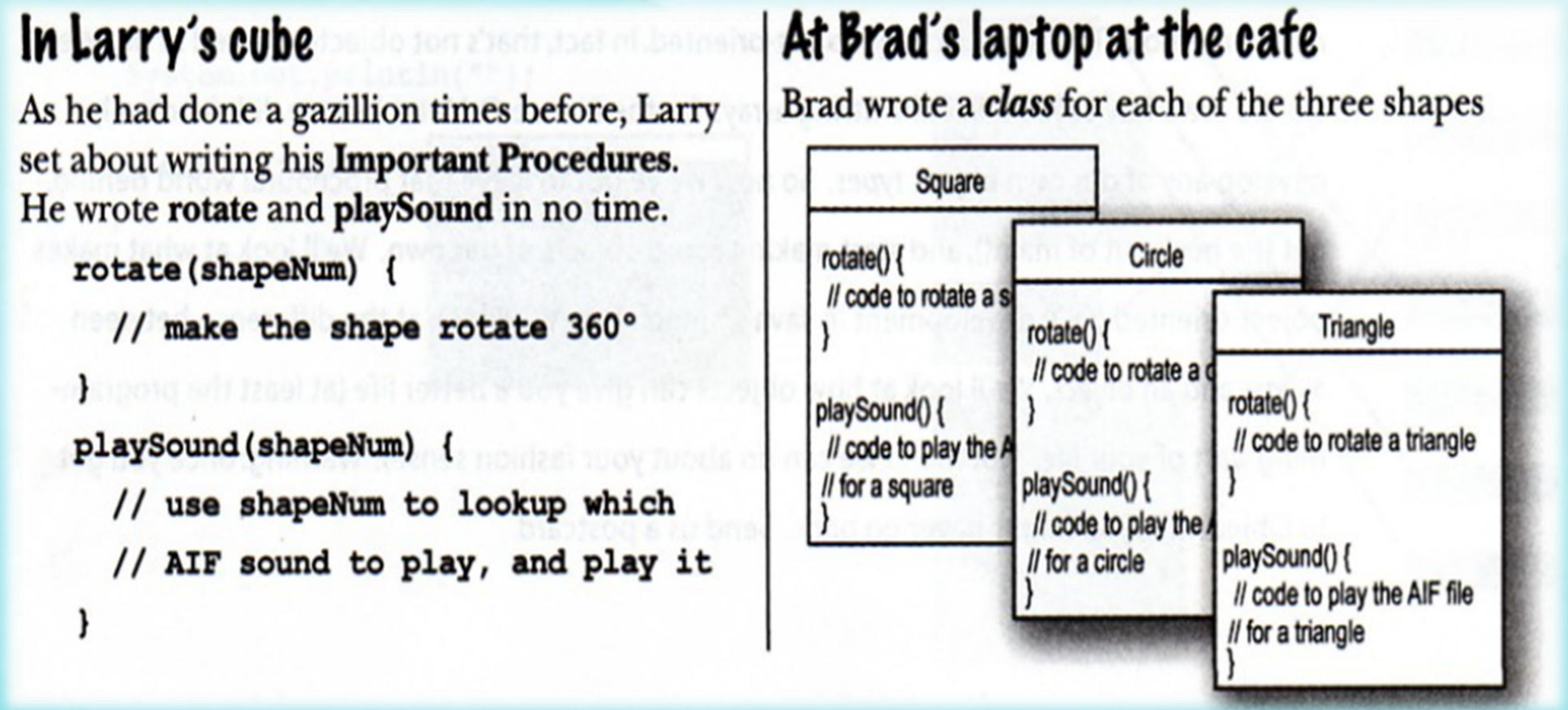
Why we need OOP? (cont'd)

Ada 2 orang bernama Larry dan Brad yang diberikan spec yang sama untuk merebutkan sebuah kursi dari bosnya!

Larry = think procedural

Brad = think OOP

Why we need OOP? (cont’d)



In Larry’s cube

As he had done a gazillion times before, Larry set about writing his **Important Procedures**. He wrote **rotate** and **playSound** in no time.

```
rotate(shapeNum) {
  // make the shape rotate 360°
}
playSound(shapeNum) {
  // use shapeNum to lookup which
  // AIF sound to play, and play it
}
```

At Brad’s laptop at the cafe

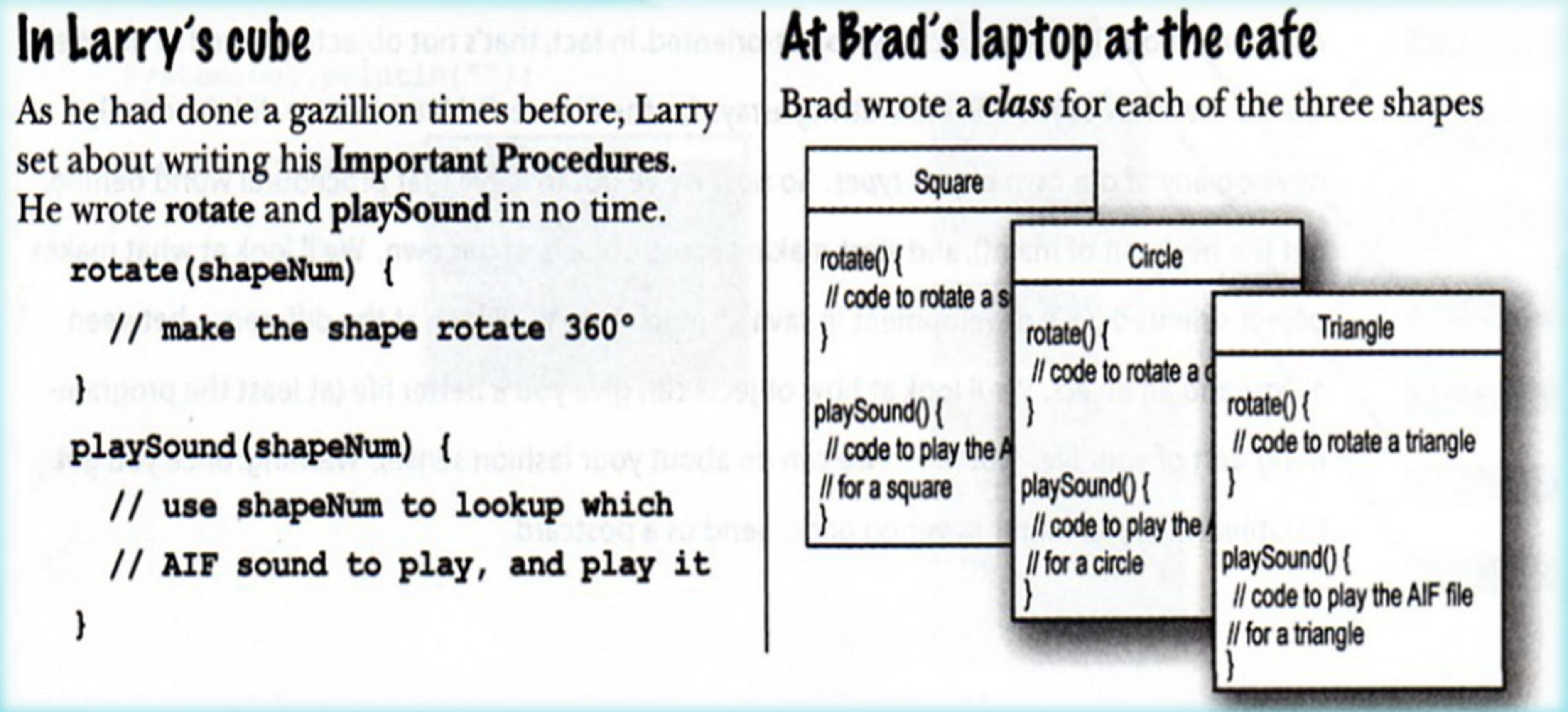
Brad wrote a ***class*** for each of the three shapes

```
Square
rotate() {
 // code to rotate a s
}
playSound() {
 // code to play the A
 // for a square
}
```

```
Circle
rotate() {
 // code to rotate a c
}
playSound() {
 // code to play the A
 // for a circle
}
```

```
Triangle
rotate() {
 // code to rotate a triangle
}
playSound() {
 // code to play the AIF file
 // for a triangle
}
```

Why we need OOP? (cont'd)



In Larry's cube

As he had done a gazillion times before, Larry set about writing his **Important Procedures**. He wrote **rotate** and **playSound** in no time.

```
rotate(shapeNum) {
  // make the shape rotate 360°
}
playSound(shapeNum) {
  // use shapeNum to lookup which
  // AIF sound to play, and play it
}
```

At Brad's laptop at the cafe

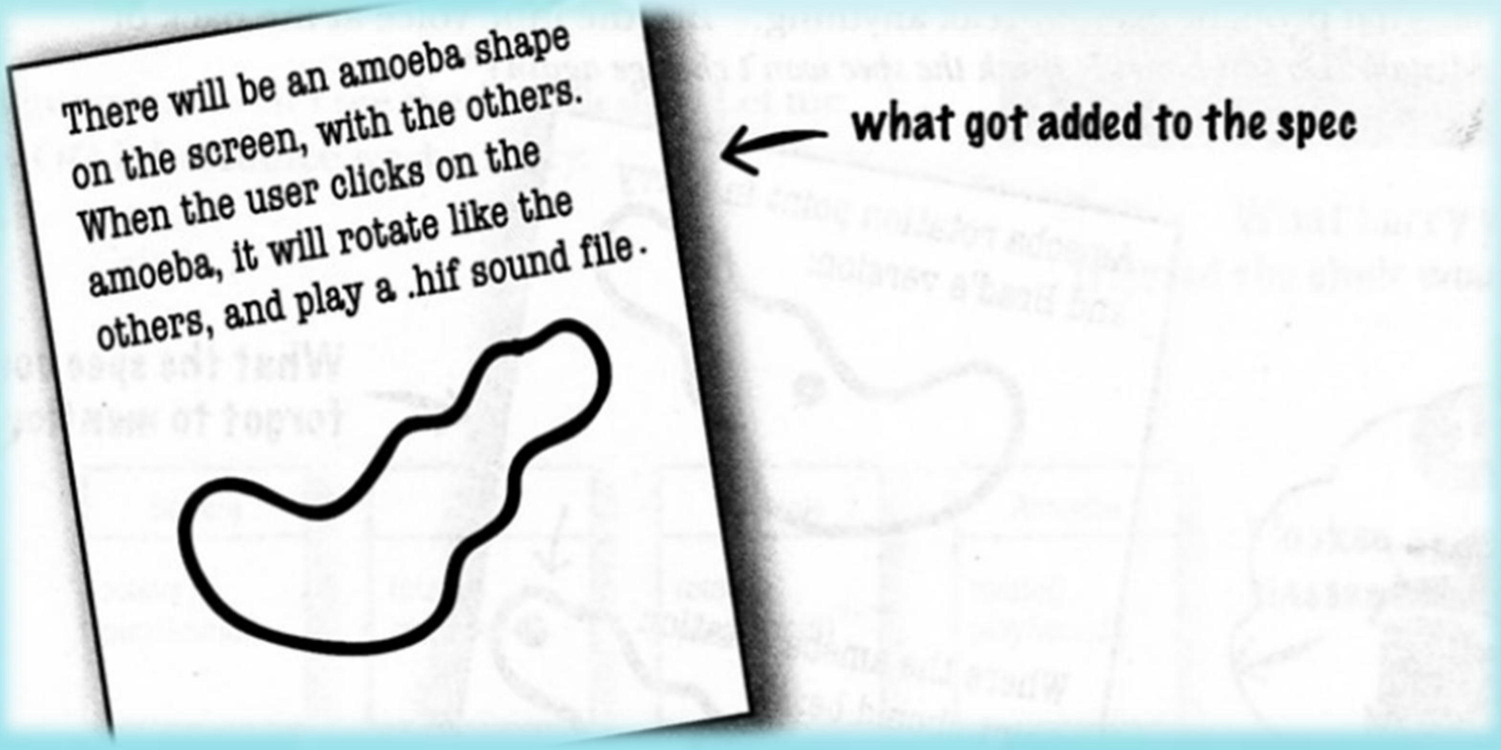
Brad wrote a ***class*** for each of the three shapes

```
Square
rotate() {
 // code to rotate a s
}
playSound() {
 // code to play the A
 // for a square
}
```

```
Circle
rotate() {
 // code to rotate a c
}
playSound() {
 // code to play the A
 // for a circle
}
```

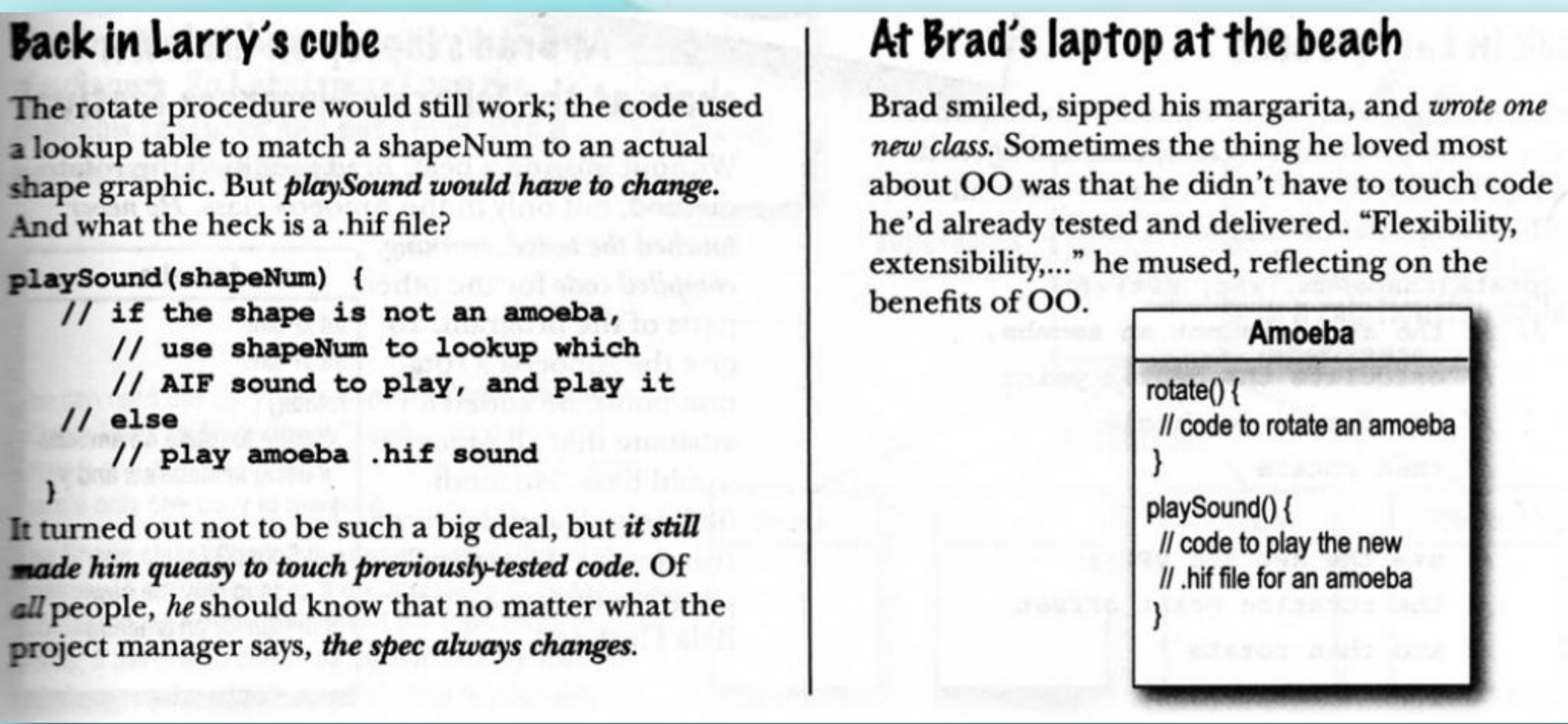
```
Triangle
rotate() {
 // code to rotate a triangle
}
playSound() {
 // code to play the AIF file
 // for a triangle
}
```

Why we need OOP? (cont'd)


There will be an amoeba shape on the screen, with the others. When the user clicks on the amoeba, it will rotate like the others, and play a .hif sound file.

what got added to the spec

Why we need OOP? (cont'd)



Back in Larry's cube

The rotate procedure would still work; the code used a lookup table to match a shapeNum to an actual shape graphic. But ***playSound would have to change.*** And what the heck is a .hif file?

```
playSound(shapeNum) {
   // if the shape is not an amoeba,
      // use shapeNum to lookup which
      // AIF sound to play, and play it
   // else
      // play amoeba .hif sound
  }
```

It turned out not to be such a big deal, but ***it still made him queasy to touch previously-tested code***. Of *all* people, *he* should know that no matter what the project manager says, ***the spec always changes.***

At Brad's laptop at the beach

Brad smiled, sipped his margarita, and *wrote one new class.* Sometimes the thing he loved most about OO was that he didn't have to touch code he'd already tested and delivered. "Flexibility, extensibility,..." he mused, reflecting on the benefits of OO.

Amoeba

```
rotate() {
  // code to rotate an amoeba
}
playSound() {
  // code to play the new
  // .hif file for an amoeba
}
```

Kira-kira siapa pemenangnya?

The Differences of Procedural and OOP

PROSEDURAL	OOP
Fokus terhadap cara komputer menyelesaikan suatu tugas	Fokus terhadap objek yang sedang digunakan

OO (Object Oriented)

Suatu paradigma yang menggunakan objek dengan identitas yang membungkus propertis dan operasi, melewatkan pesan, dan inheritance untuk menyelesaikan domain permasalahan.

OO System

Sebuah sistem yang dibangun berdasarkan metode berorientasi objek.

Tools

1. JDK versi terbaru
2. Netbeans 6.9 atau 7
3. Dev C++