Course: Object Based Modeling
Code: CS-33105
Branch: MCA-3

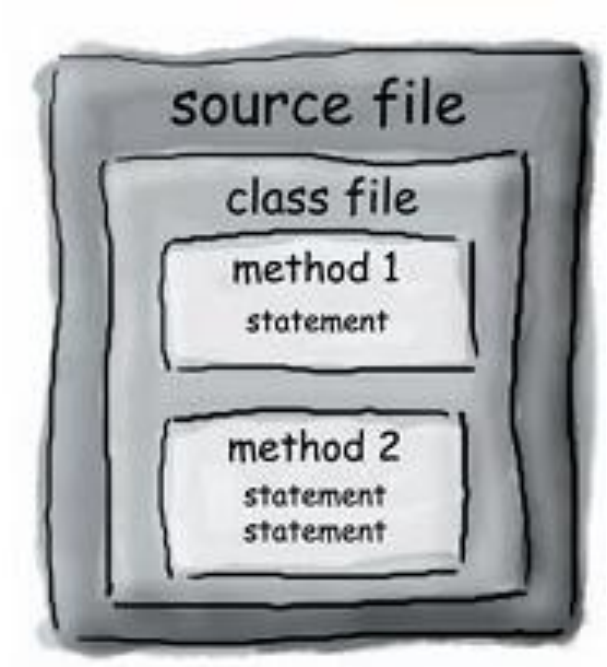
Lecture – 2

Dr. J Sathish Kumar (JSK)
(Faculty & Coordinator)

Department of Computer Science and Engineering

Motilal Nehru National Institute of Technology Allahabad,
Prayagraj-211004

An Introduction to Java



Put a class in a source file.

Put methods in a class.

Put statements in a method.

What goes in a source file?

```
public class Dog {

}
```

class

What goes in a class?

```
public class Dog {
   void bark() {
   }
}
```

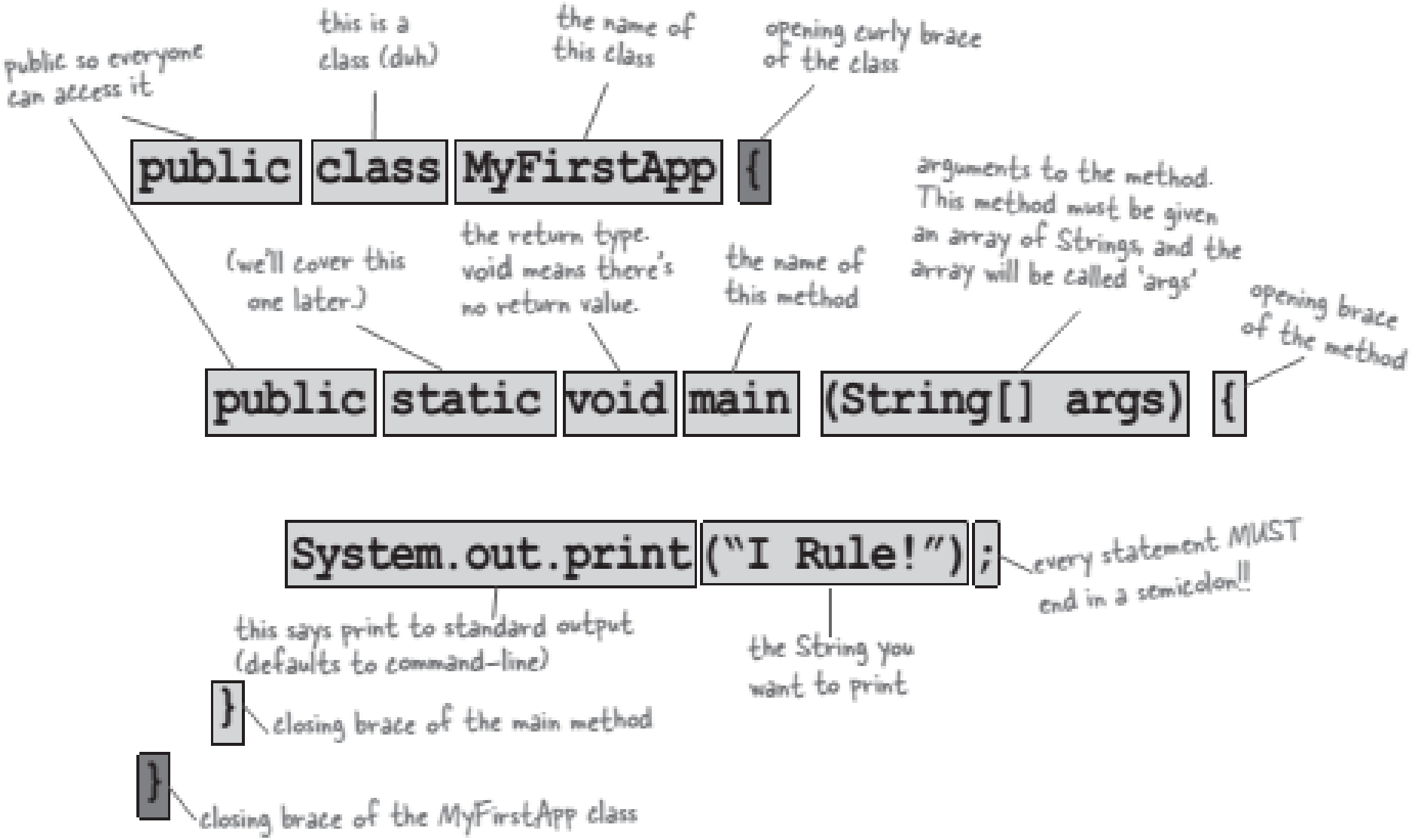
method

What goes in a method?

```
public class Dog {
   void bark() {
     statement1;
     statement2;
   }
}
```

statements

Anatomy of a class



Compile and Execute

1 Save

```
MyFirstApp.java
```

2 Compile

```
javac MyFirstApp.java
```

3 Run

File Edit Window Help Scream

```
%java MyFirstApp
I Rule!
The World
```

Syntax

- Each statement must end in a semicolon.
  - x = x + 1**;**
- A single-line comment begins with two forward slashes.
  - x = 22;
    // this line disturbs me
- Most white space doesn't matter.
  - x = 3 ;
- Variables are declared with a **name** and a **type**
  - int weight;
    //*type:* int, *name:* weight
- Classes and methods must be defined within a pair of curly braces.
  - public void go()
    {
    // amazing code here
    }

Statements

Declarations, assignments, method calls, etc.

```
int x = 3;
String name = "Dirk";
x = x * 17;
System.out.print("x is " + x);
double d = Math.random();
// this is a comment
```

Branching: *if/else* tests

```
if (x == 10) {
   System.out.print("x must be 10");
} else {
   System.out.print("x isn't 10");
}
if ((x < 3) & (name.equals("Dirk"))) {
   System.out.println("Gently");
}
System.out.print("this line runs no matter what");
```

Loops: *for* and *while*

```
while (x > 12) {
    x = x -1;
}

for (int x = 0; x < 10; x = x + 1) {
    System.out.print("x is now " + x);
}
```

Class Exercise #1

```
public class Loopy {
   public static void main (String[] args) {
     int x = 1;
     System.out.println("Before the Loop");
     while (x < 4) {
       System.out.println("In the loop");
       System.out.println("Value of x is " + x);
         x = x + 1;
     }
     System.out.println("This is after the loop");
   }
}
```

this is the output

```
% java Loopy
Before the Loop
In the loop
Value of x is 1
In the loop
Value of x is 2
In the loop
Value of x is 3
This is after the loop
```

Class Exercise #2

```
class IfTest {
  public static void main (String[] args) {
    int x = 3;
    if (x == 3) {
      System.out.println("x must be 3");
    }
    System.out.println("This runs no matter what");
  }
}
```

```
% java IfTest
x must be 3
This runs no matter what
```

Class Exercise #3

Given the output:

```
% java DooBee
DooBeeDooBeeDo
```

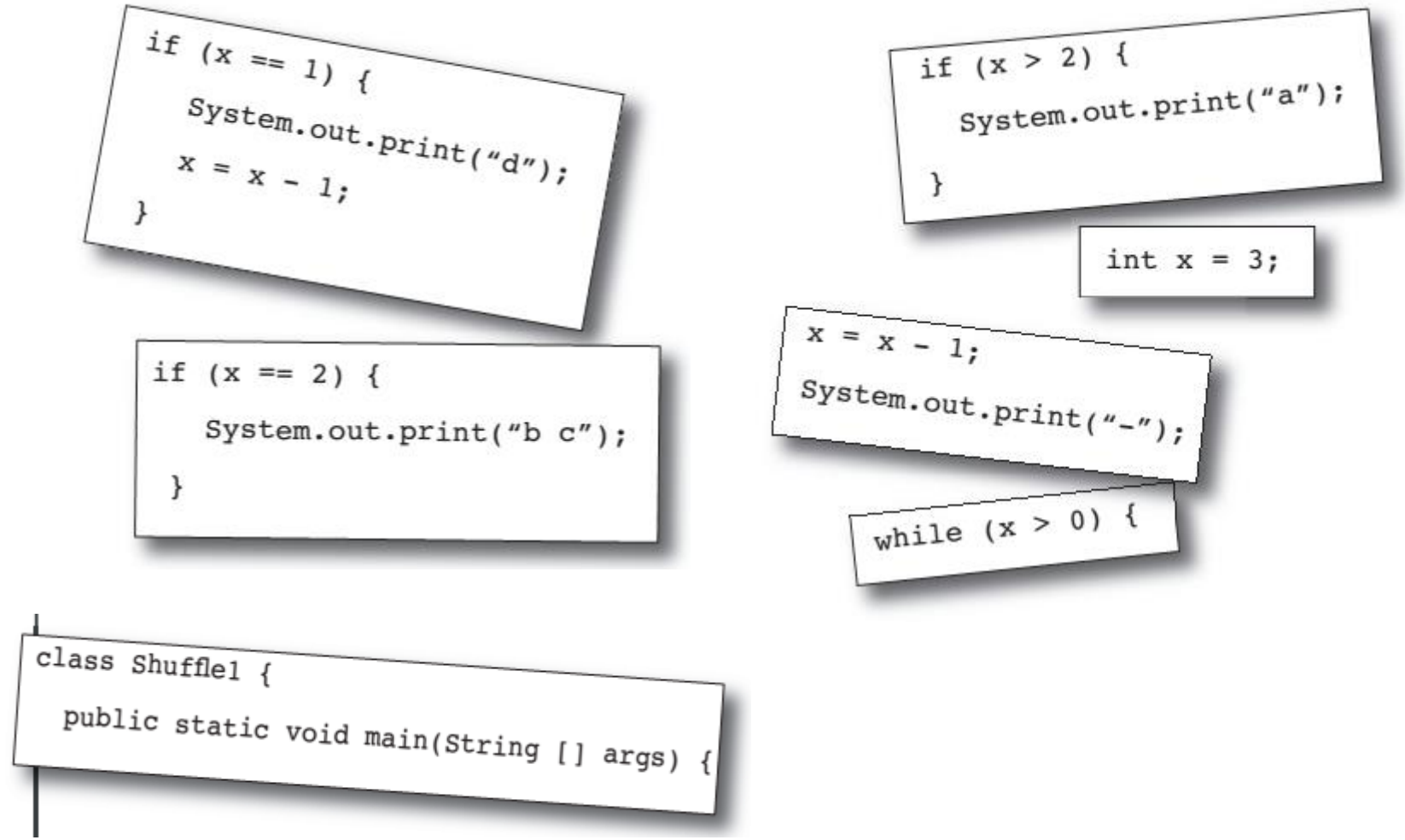
Fill in the missing code:

```
public class DooBee {
  public static void main (String[] args) {
   int x = 1;
   while (x < _____ ) {
    System.out._____("Doo");
    System.out._____("Bee");
    x = x + 1;
   }
   if (x == _____  ) {
     System.out.print("Do");
   }
  }
}
```

Class Exercise #4

Given the output:

```
% java Shuffle1
a-b c-d
```



Class Exercise #5

A

```
class Exercise1b {
  public static void main(String [] args) {
    int x = 1;
    while ( x < 10 ) {
      if ( x > 3) {
        System.out.println("big x");
      }
    }
  }
}
```

B

```
public static void main(String [] args) {
    int x = 5;
    while ( x > 1 ) {
      x = x - 1;
      if ( x < 3) {
        System.out.println("small x");
      }
    }
}
```

C

```
class Exercise1b {
    int x = 5;
    while ( x > 1 ) {
      x = x - 1;
      if ( x < 3) {
        System.out.println("small x");
      }
    }
}
```

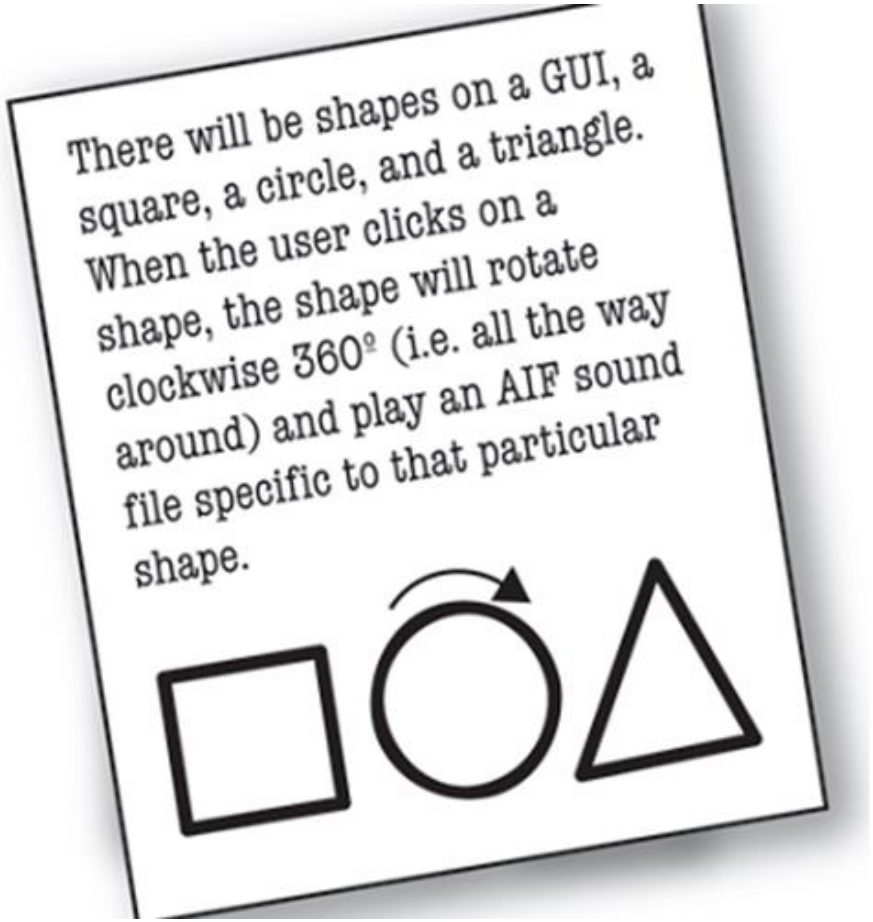
BE the compiler

Identify the correct program!!

Object-Oriented Programming Features in Java

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

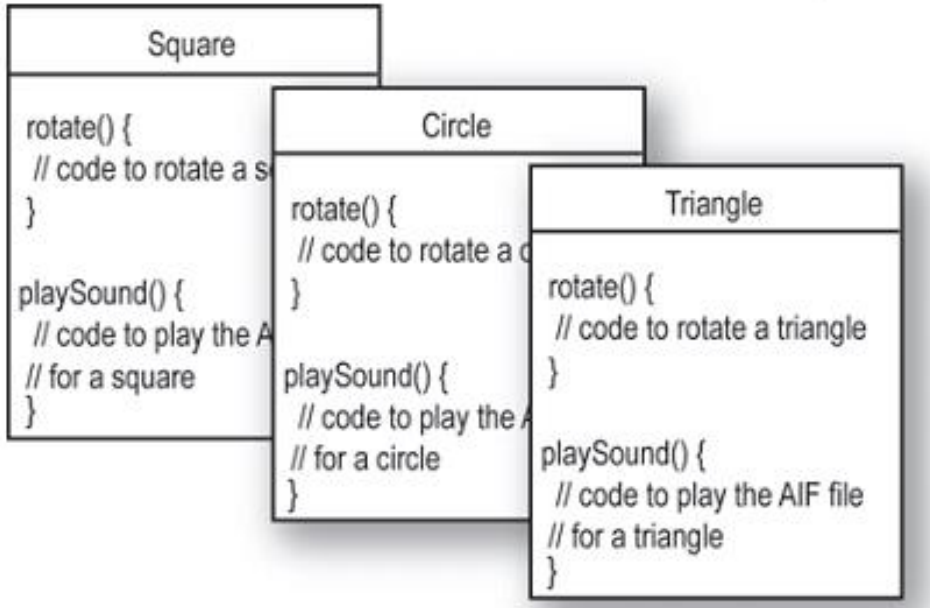
An Illustrative Example



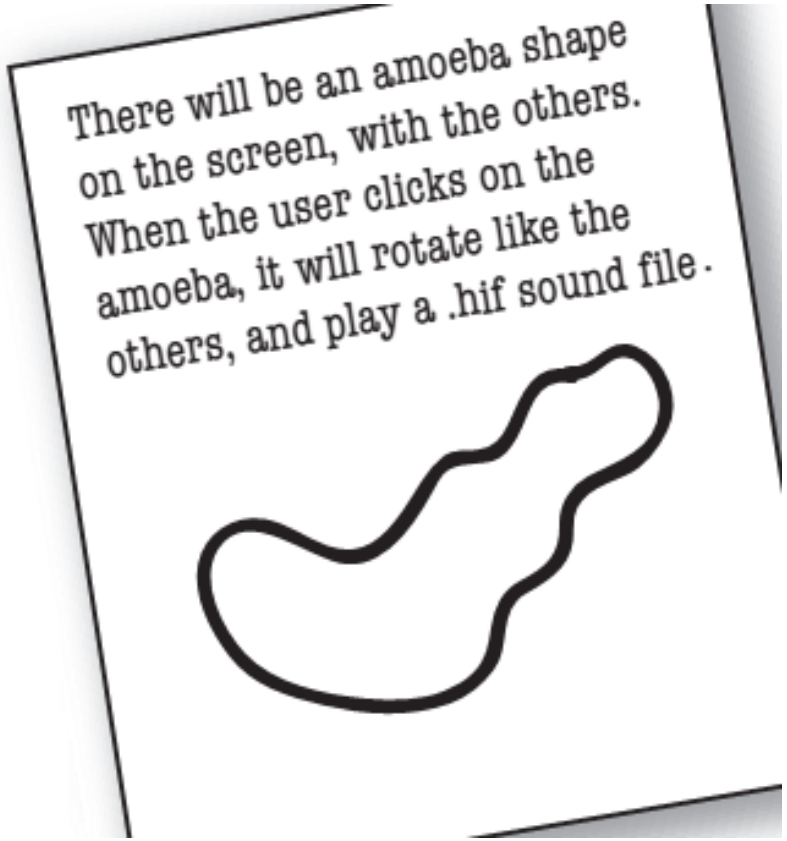
Programmer 1

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

Programmer 2



Specification change!!!



Programmer 1

```
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
}
```

Wrote New Class

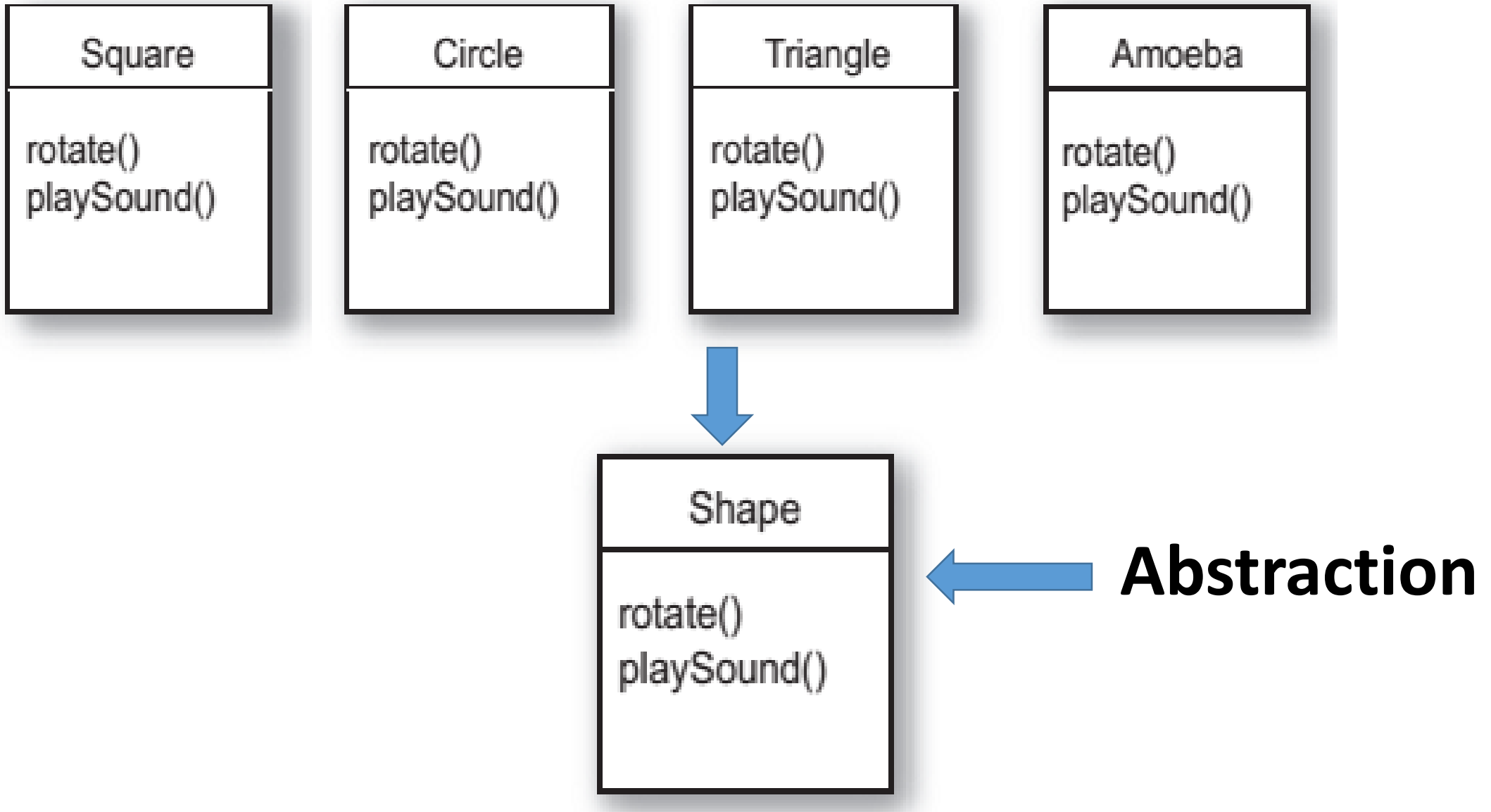
Programmer 2

Amoeba

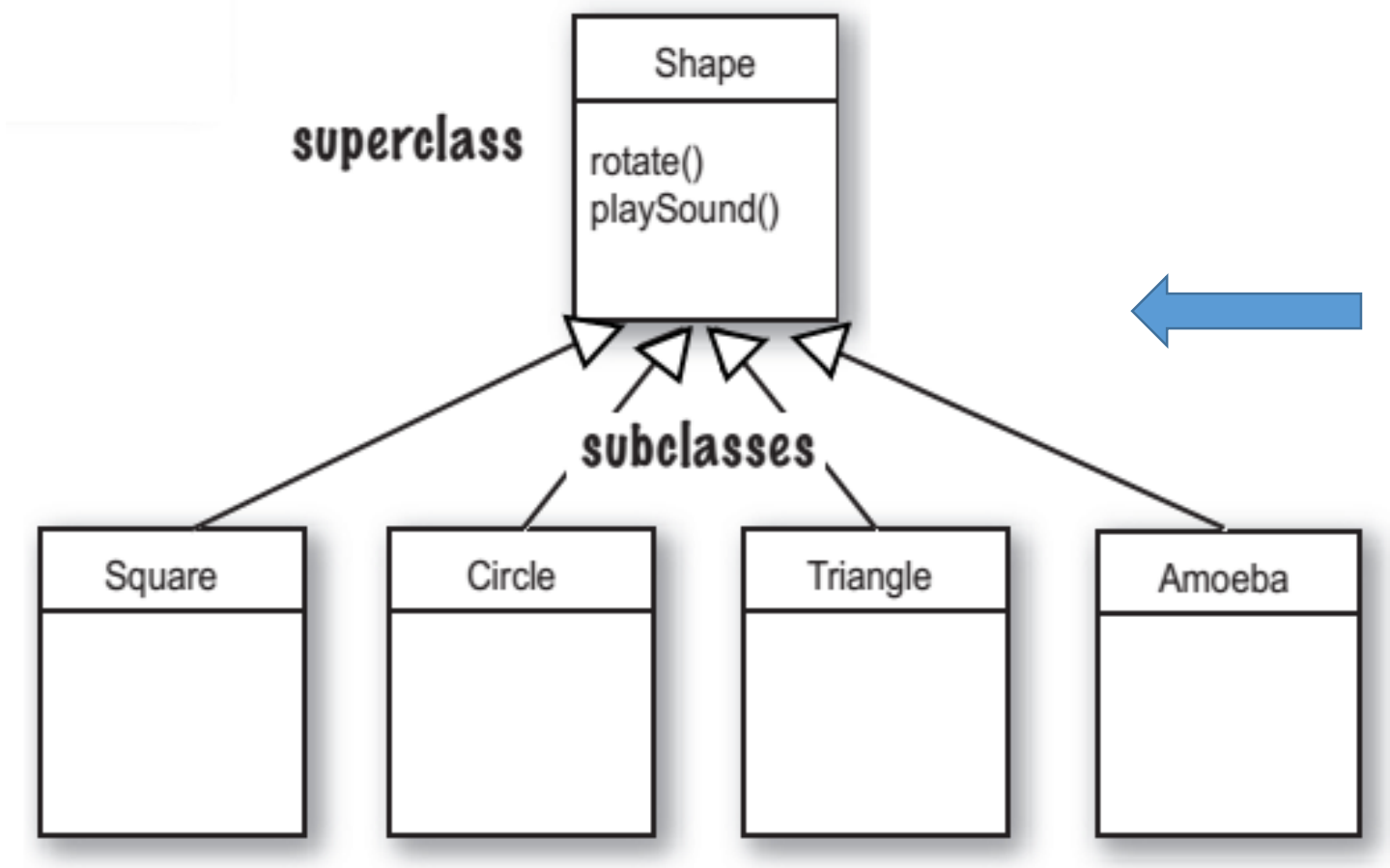
```
rotate() {
  // code to rotate an amoeba
}

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

OOB Concepts

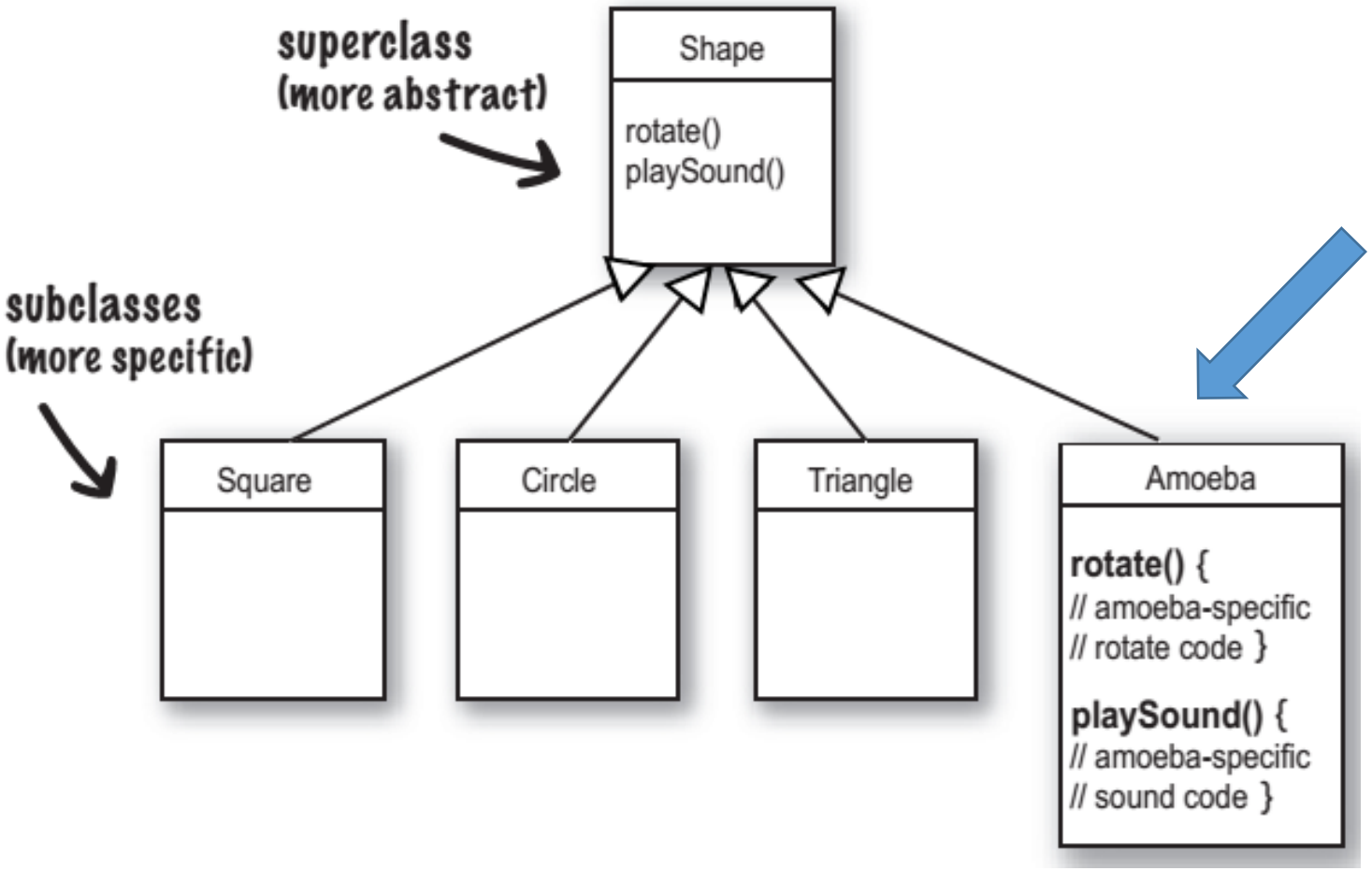


OOB Concepts



Inheritance

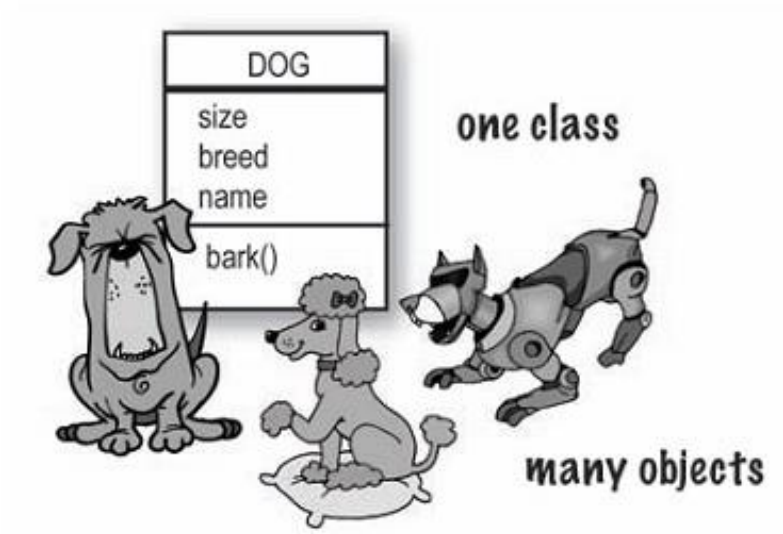
OOB Concepts



Over riding methods in Inheritance

Class and an Object?

- A class is not an object.
  - but it's used to construct them
- **A class is a *blueprint* for an object.**
- Each object made from that class can have its own values for the instance variables of that class.
- **An Illustrative Example**



An Illustrative Example

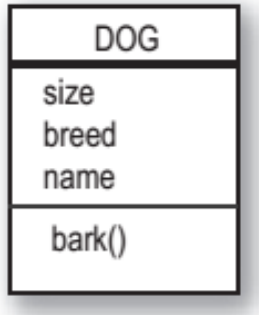
```
class Dog {

  int size;
  String breed;
  String name;

  void bark() {
    System.out.println("Ruff! Ruff!");
  }
}
```

instance variables

a method




```
class DogTestDrive {
   public static void main (String[] args) {
     Dog d = new Dog();
     d.size = 40;
     d.bark();
   }
}
```

make a Dog object

use the dot operator (.) to set the size of the Dog

and to call its bark() method

dot operator