Review of Java

CSCI-C 322 Object-Oriented Software Methods

Indiana University Bloomington - Spring 2023

Outline

- How Java works
- Classes and objects
- Inheritance
- Polymorphism

The way Java works

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

class Party {
 public void buildInvite() {
  Frame f = new Frame();
  Label l = new Label("Party at Tim's");
  Button b = new Button("You bet");
  Button c = new Button("Shoot me");
  Panel p = new Panel();
  p.add(l);
 }  // more code here...
}
```

Source

1

Type your source code.

Save as: ***Party.java***

```
File  Edit  Window  Help  Plead
%javac Party.java
```

Compiler

2

Compile the ***Party.java*** file by running `javac` (the compiler application). If you don't have errors, you'll get a second document named ***Party.class***.

The compiler-generated Party.class file is made up of *bytecodes*.

```
Method Party()
  0 aload_0
  1 invokespecial #1 <Method java.lang.Object()>
  4 return
Method void buildInvite()
  0 new #2 <Class java.awt.Frame>
  3 dup
  4 invokespecial #3 <Method java.awt.Frame()>
```

Output (code)

3

Compiled code: ***Party.class***

```
File  Edit  Window  Help  Swear
%java Party
```

Virtual Machines

4

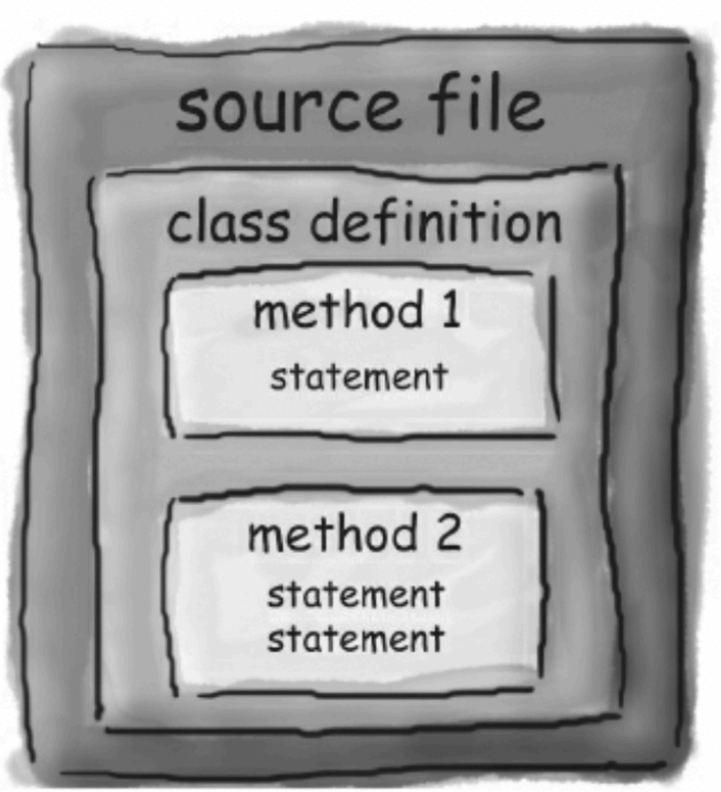
Run the program by starting the Java Virtual Machine (JVM) with the ***Party.class*** file. The JVM translates the *bytecode* into something the underlying platform understands, and runs your program.

Speed and memory usage

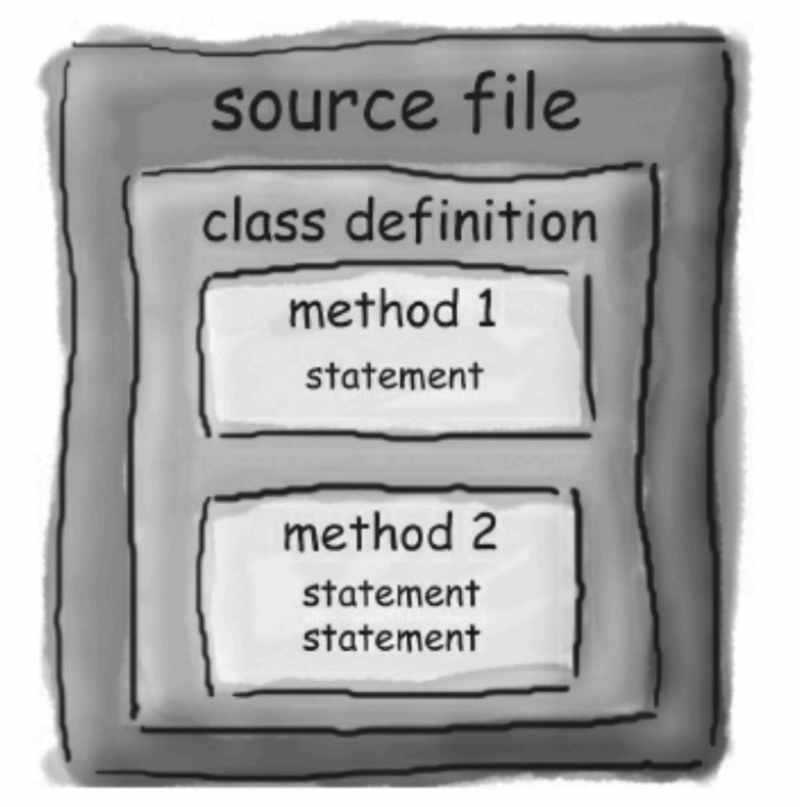
- Java is as fast as C and Rust, well, almost!
- Compared to C and Rust, Java uses a lot of memory.

Code structure in Java

- A source code file (.java extension) typically holds one class definition.
- In a class there are methods.
- In a method there are statements.



Code structure in Java




```
public class Dog {



}
```

class

```
public class Dog {

   void bark() {



   }

}
```

method

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

statements

Anatomy of a class

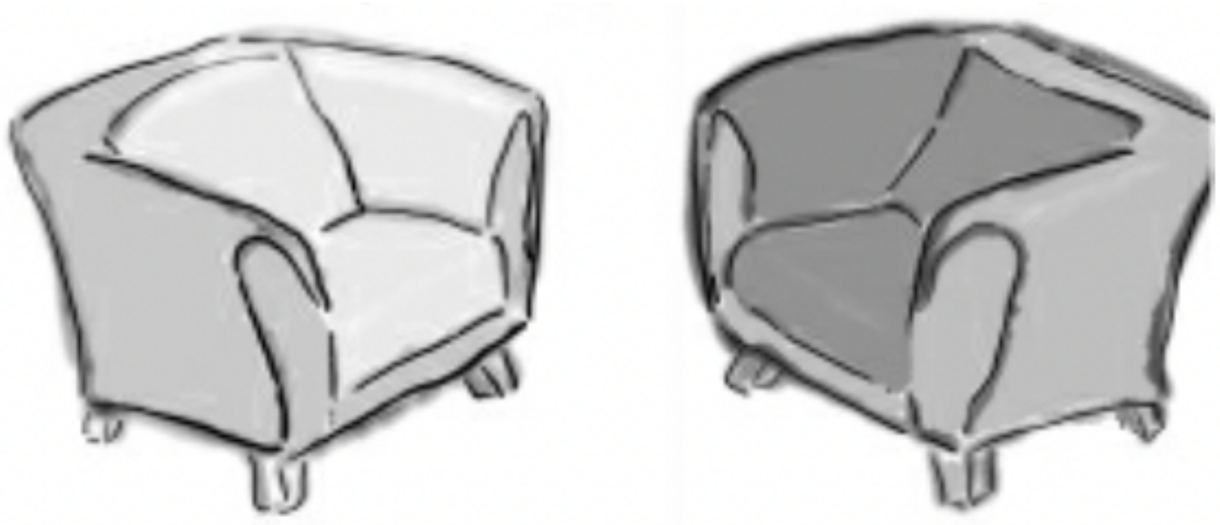
- When the JVM starts running, it looks for a method written exactly like:

```
public static void main (String[] args) {
    // your code goes here
}
```

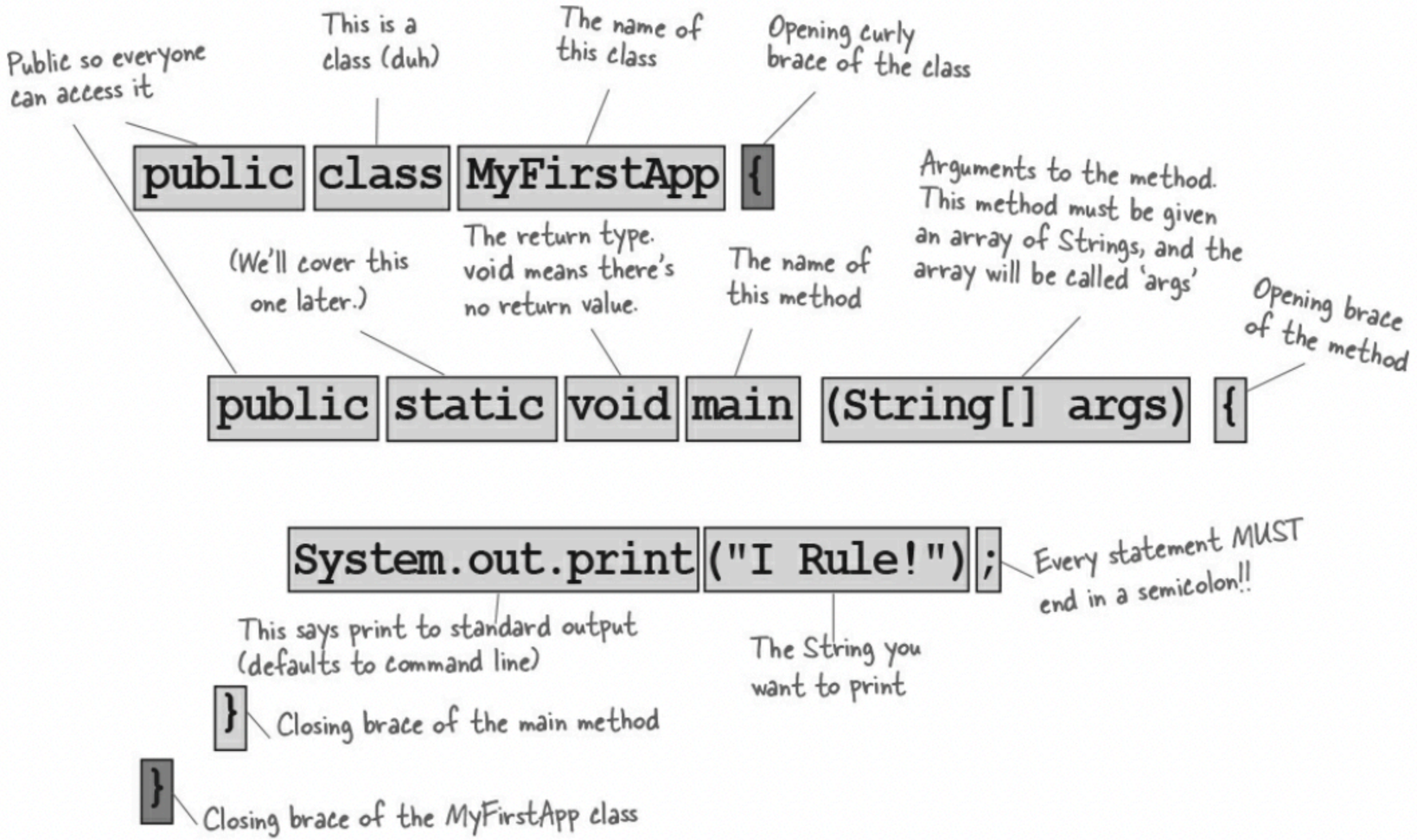
- Next the JVM runs the statements in the method
- Every Java application has to have at least one class and at least one *main* method.
- Not one *main* per class; just one *main* per application

Fireside chats

- Tonight's talk: **The compiler and the JVM battle over the question, "Who is more important?"**



Anatomy of a class



Java Syntax

❶ do something

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
```

❷ do something again and again

Loops: *for* and *while*

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

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

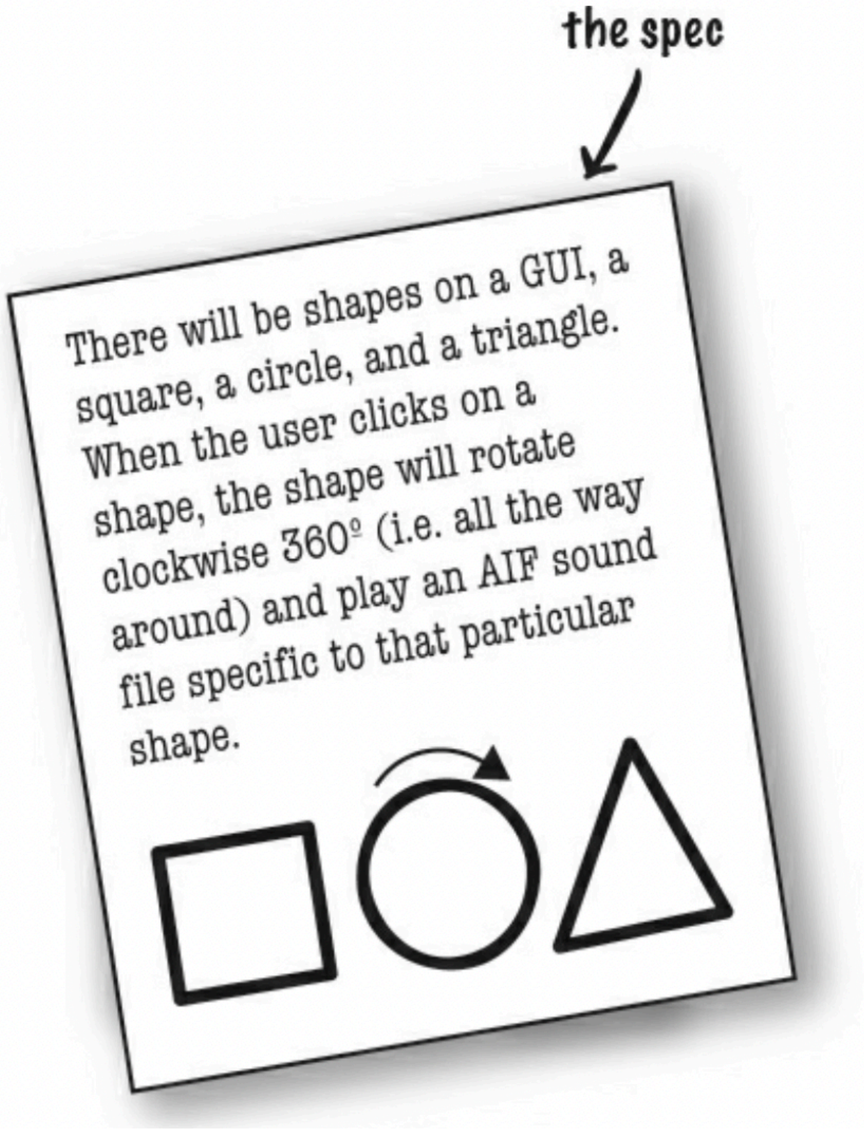
❸ do something under this condition

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");
```

Classes and Objects

- Is a program a pile of procedures?
- Or a number of things and key players?

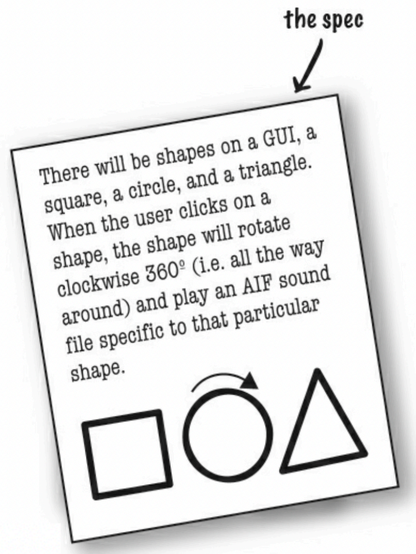


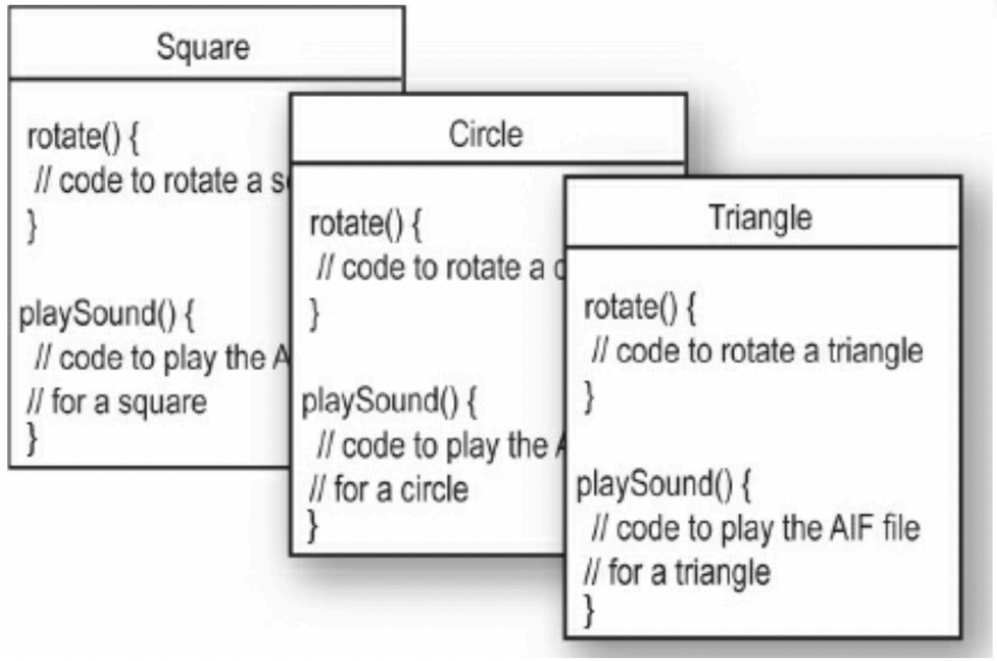
Classes and Objects

- Is a program a pile of procedures?
- Or a number of things and key players?

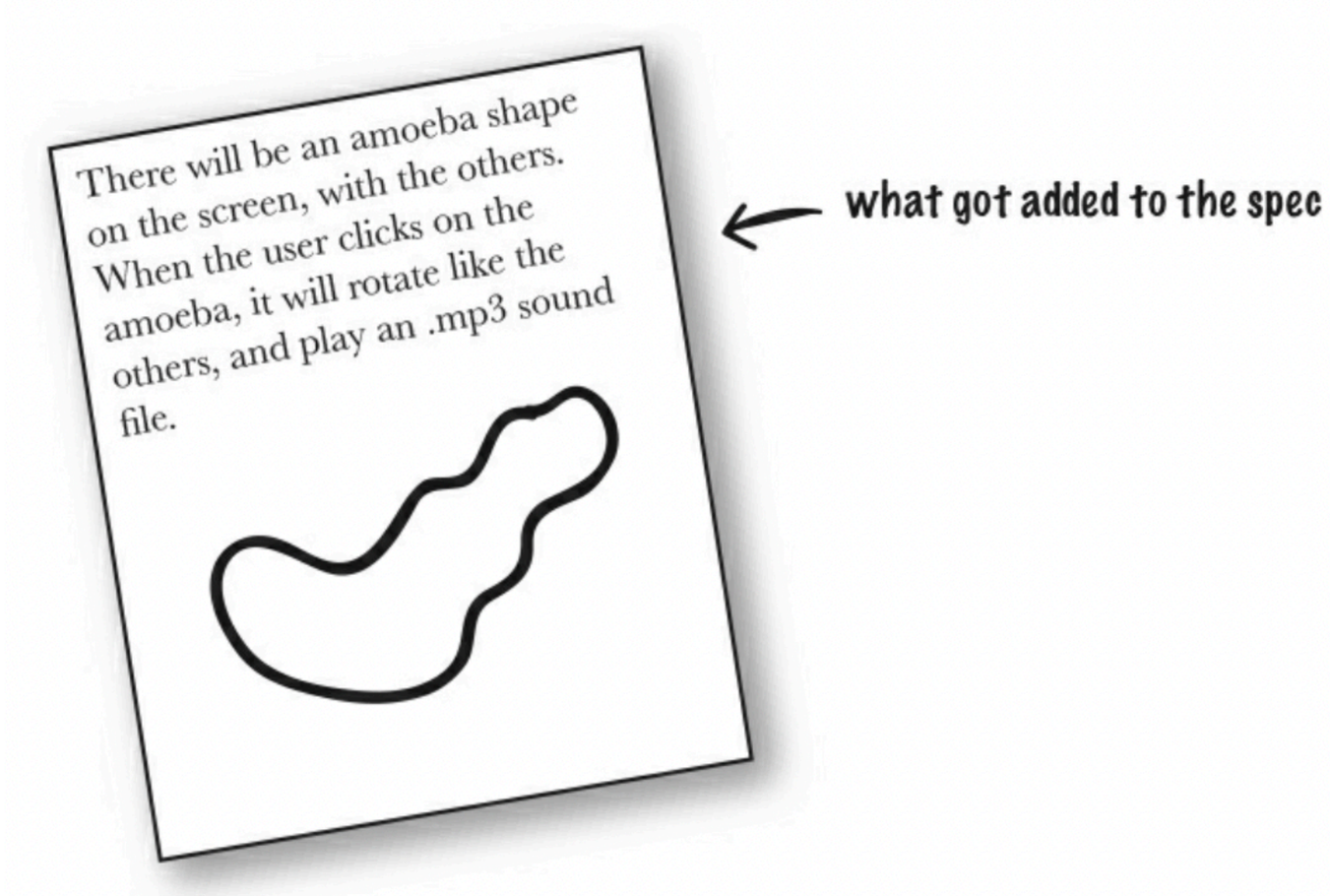
```
rotate(shapeNum) {
   // make the shape rotate 360°
}
playSound(shapeNum) {

  // use shapeNum to lookup which
  // AIF sound to play, and play it
}
```



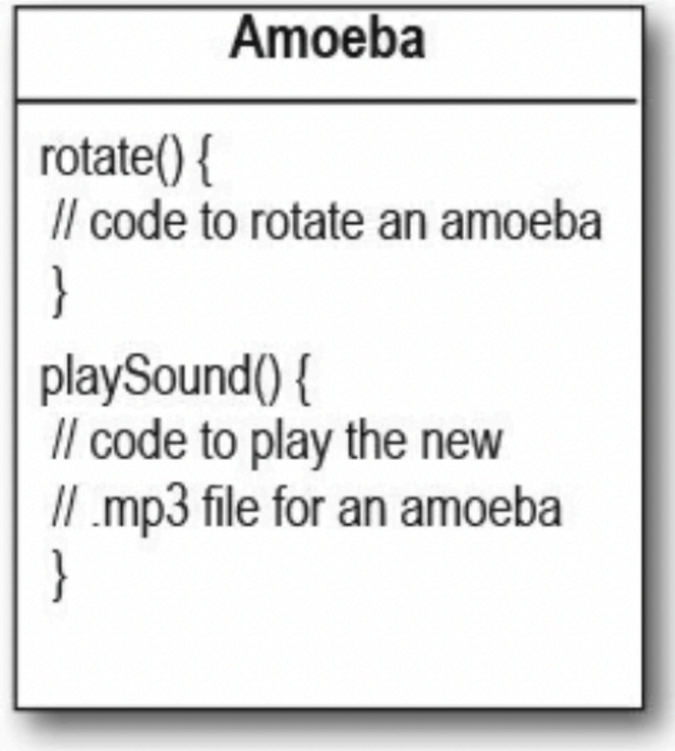


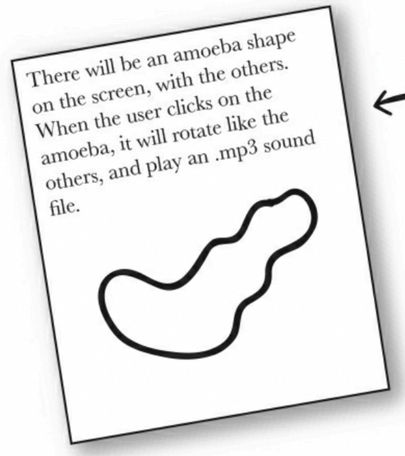
What if there is a spec change?



What if there is a spec change?

```
playSound(shapeNum) {
   // if the shape is not an amoeba,
      // use shapeNum to lookup which
      // AIF sound to play, and play it
   // else
      // play amoeba .mp3 sound
}
```





And yet another change?

- Both programmers had written their rotate code like this:
  - Determine the rectangle that surrounds the shape
  - Calculate the center of that rectangle and rotate around that point

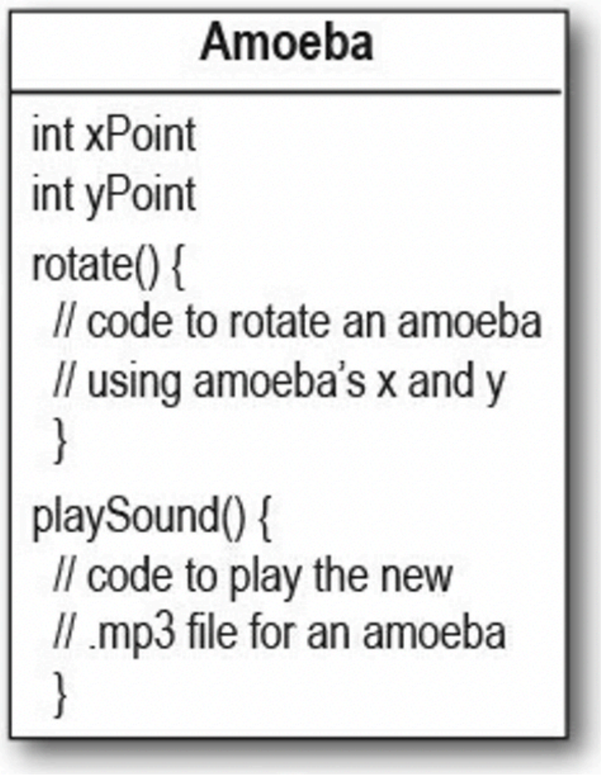
And yet another change?

- But amoeba is supposed to rotate around a point on one end!

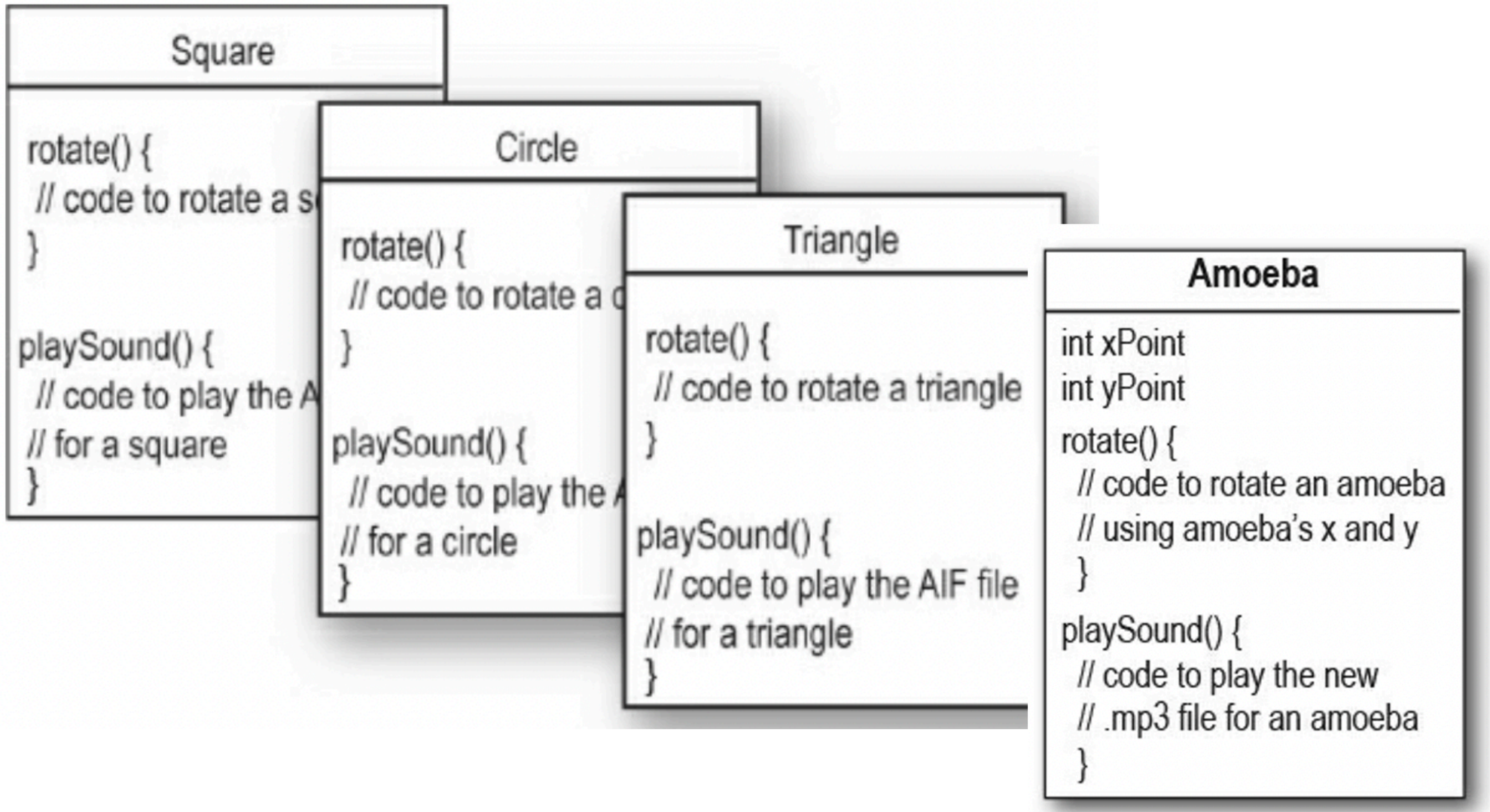
And yet another change?

- But amoeba is supposed to rotate around a point on one end!

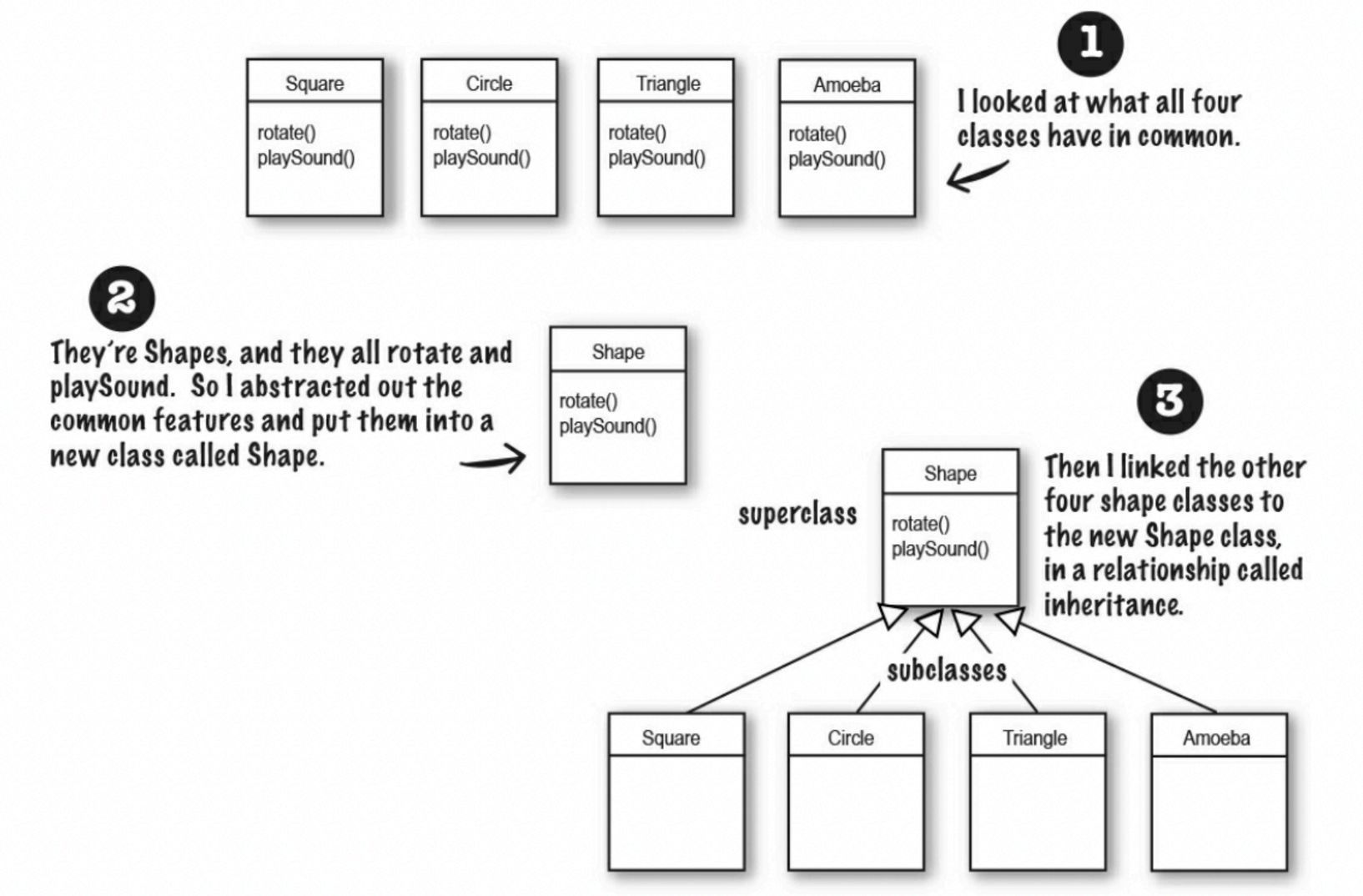
```
rotate(shapeNum, xPt, yPt) {
  // if the shape is not an amoeba,
    // calculate the center point
    // based on a rectangle,
    // then rotate
  // else
    // use the xPt and yPt as
    // the rotation point offset
    // and then rotate
}
```



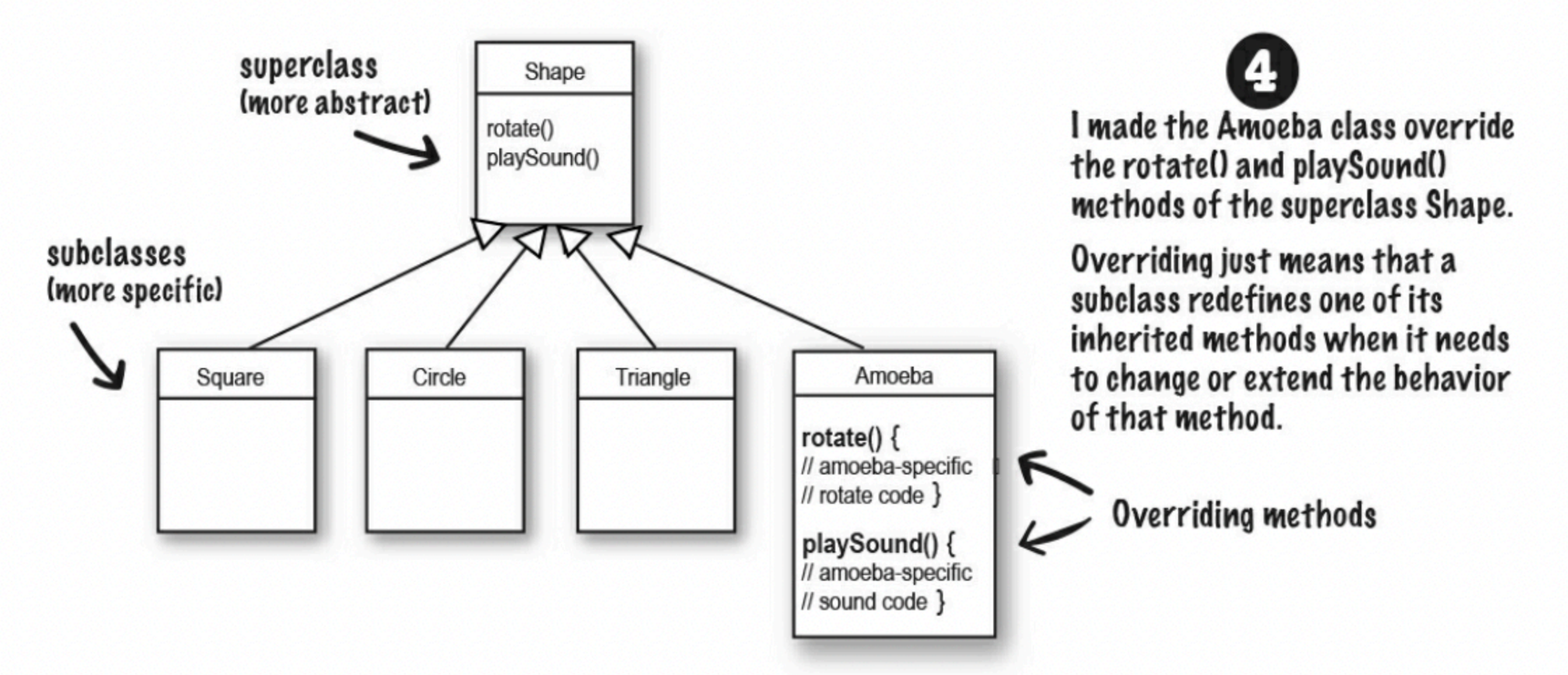
But we have got duplicated code!



Abstraction and Inheritance



Inheritance

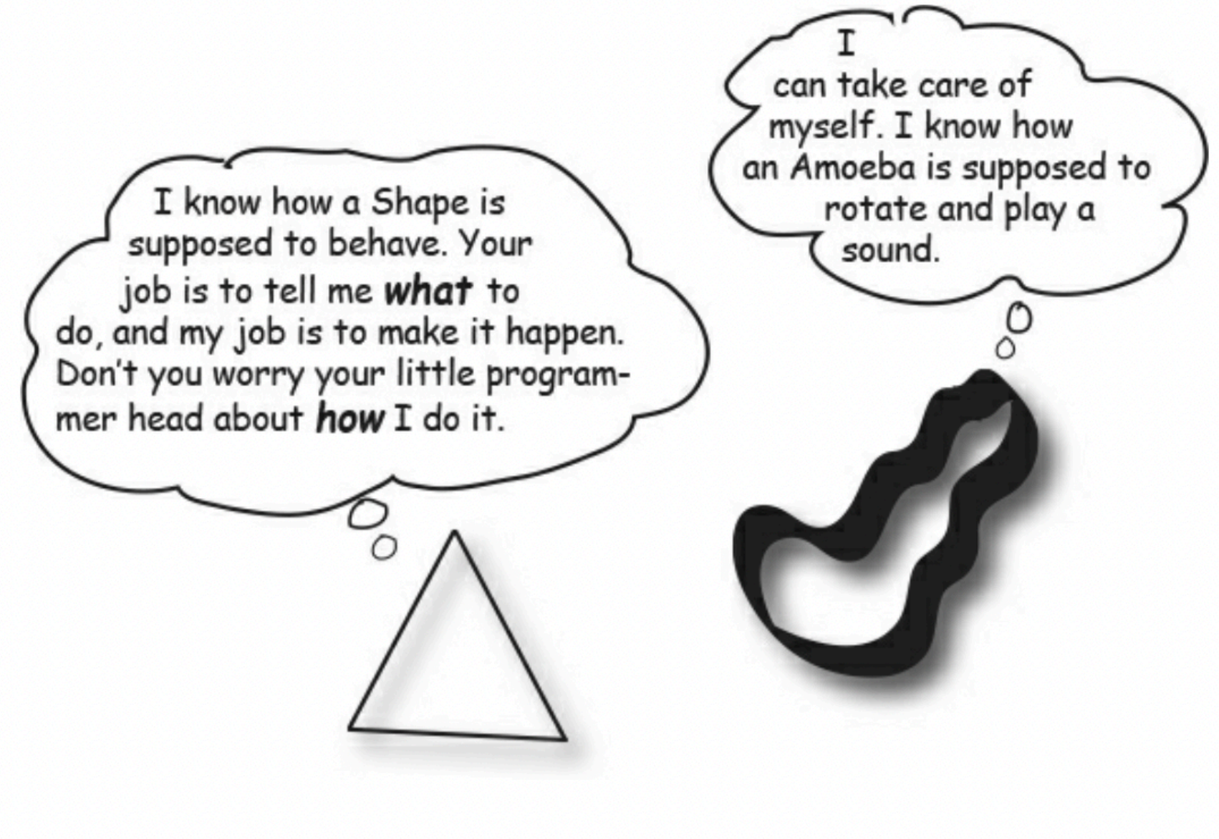


What do you like about Object Oriented design?

- Design in a more natural way.
- Not messing around with the code that has been already tested; just add a new feature.
- The data and the methods operating on the data are together in one class.
- Reusing code in other applications.

Objects

How do you tell a triangle to do something?



How to design a Java class?

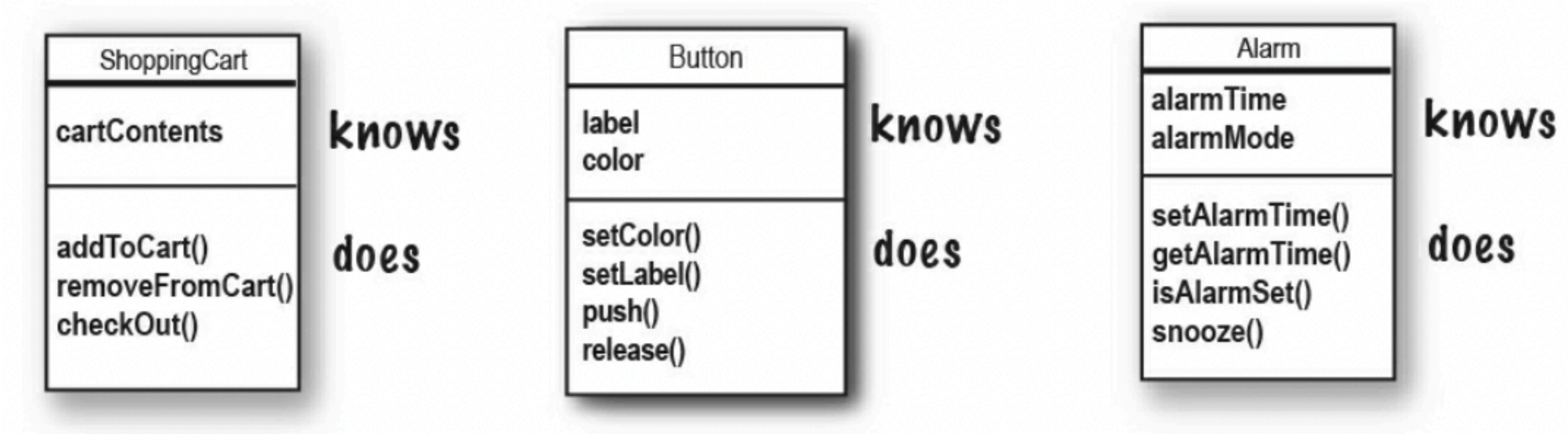
- What are the fundamental things you need to think about when you design a Java class?
- What are the questions you need to ask yourself?

How to design a Java class?

- Think about the objects that will be created from the class type.
  - things the object knows
  - things the object does

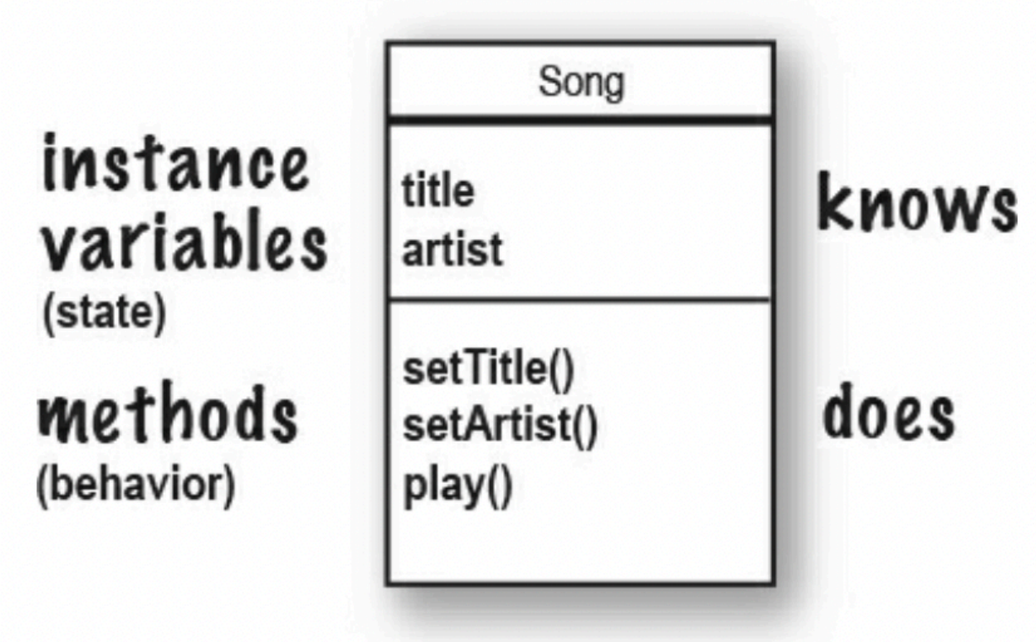
How to design a Java class?

- Think about the objects that will be created from the class type.
  - things the object knows
  - things the object does



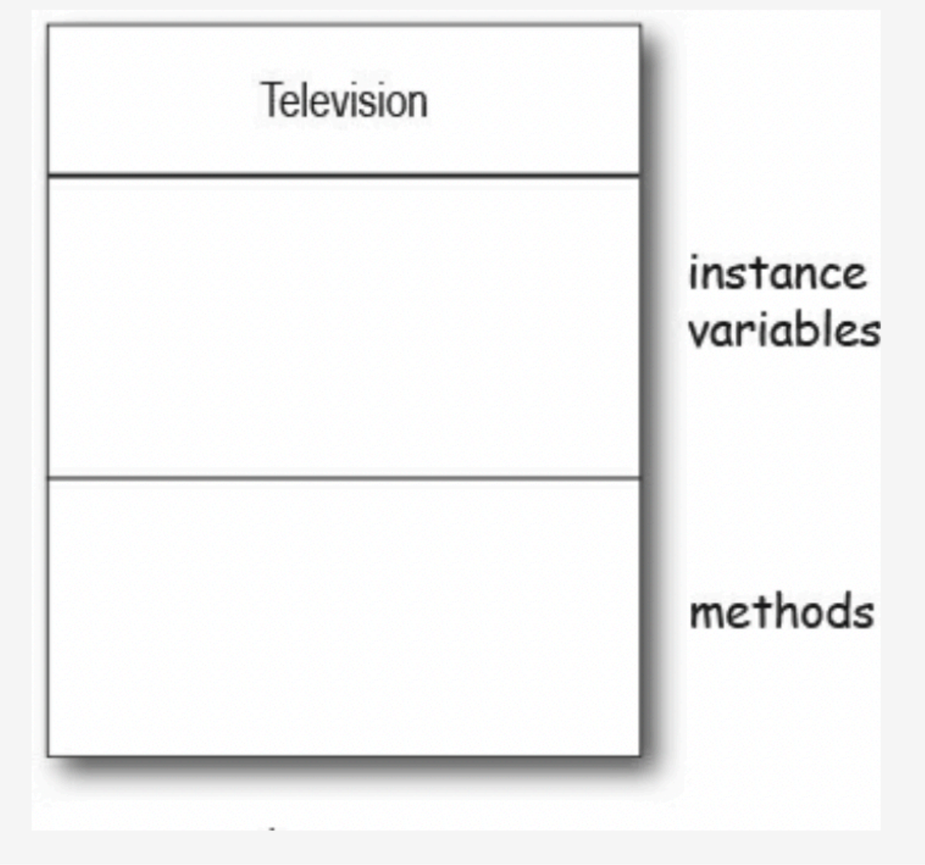
How to design a Java class?

- Things an object knows about itself are called *instance variables*.
- Things an object can do are called *methods*.

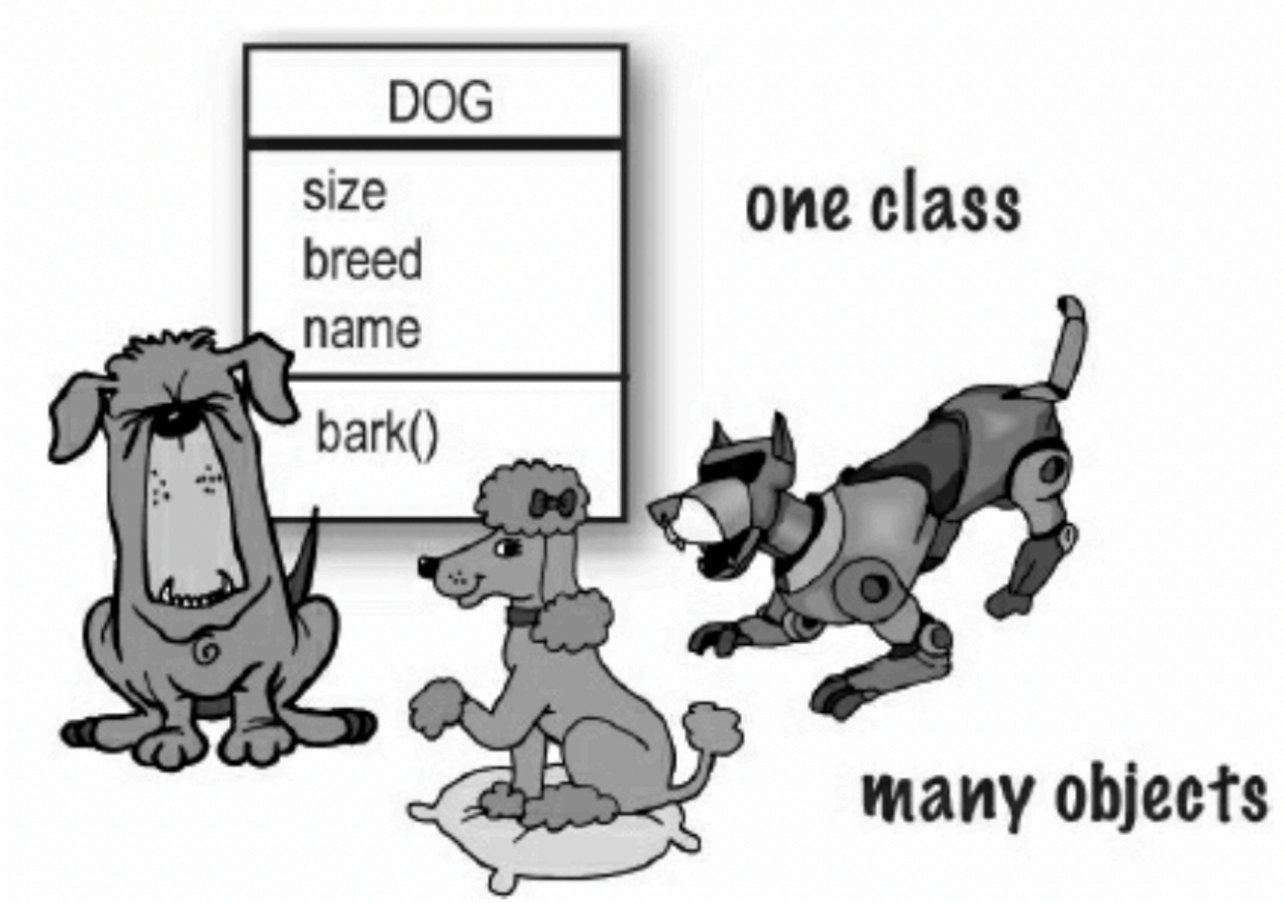


Exercise

A television object

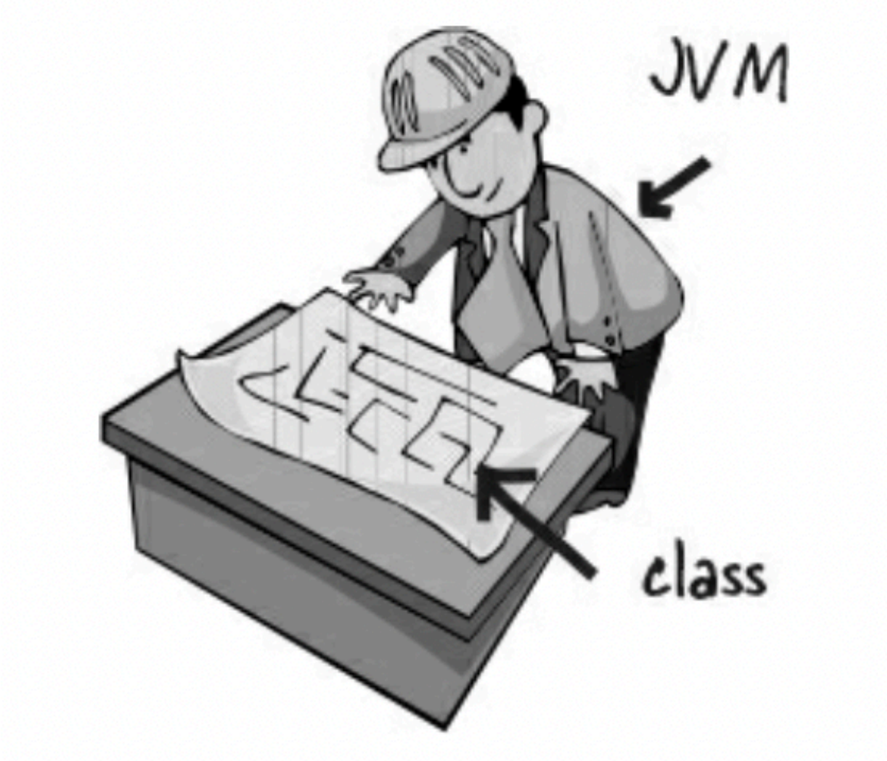


Class versus object



Class versus object

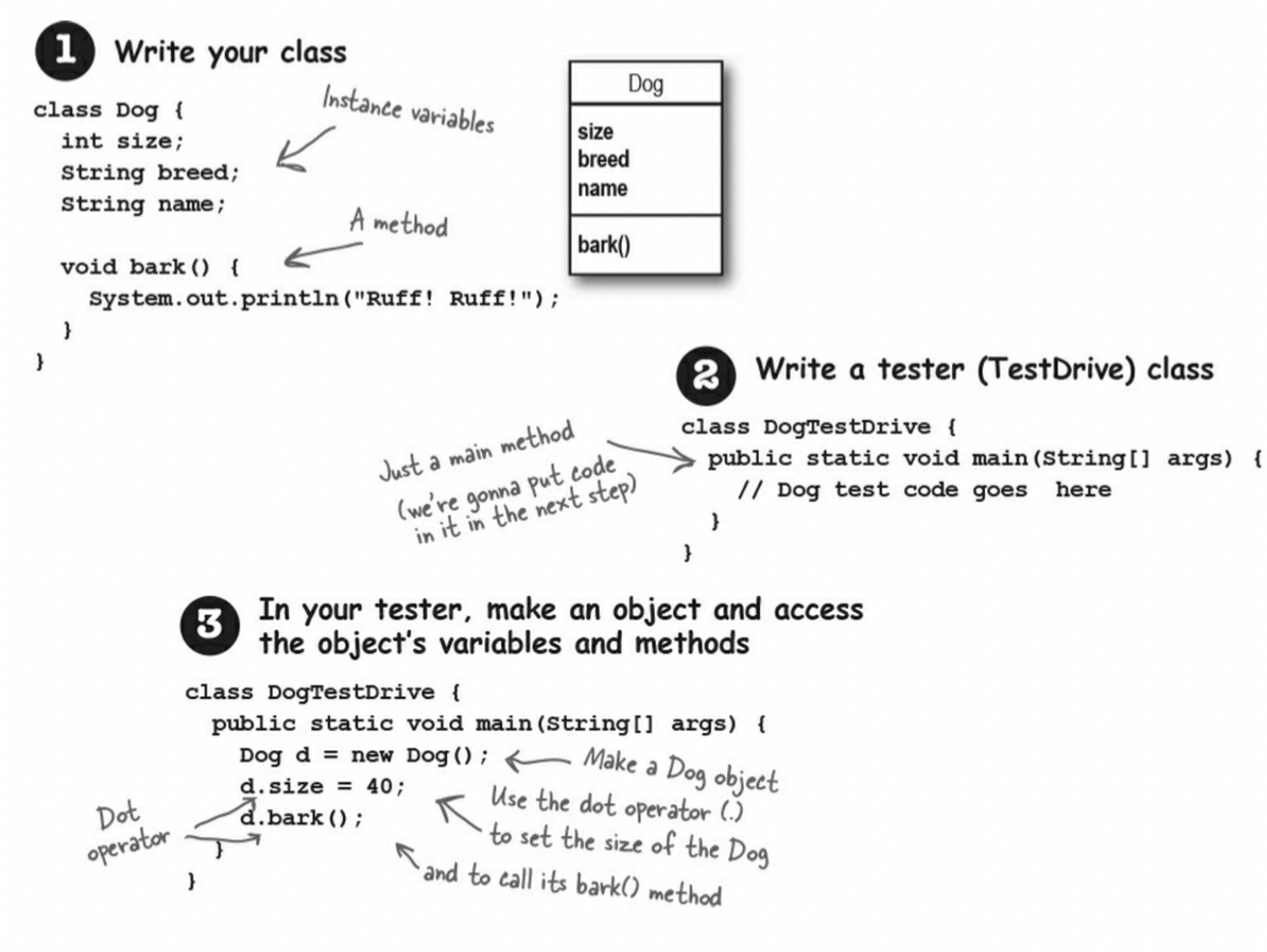
- A class is a blueprint for an object



How to create and use an object?

- We need *two* classes:
  - One class for the type of object you want to create and use
  - Another class to *test* your new class
- The *tester* class where we put the main method
- In the main method we create and access objects of our new class type

How to create and use an object?



Java Heap

- Each time an object is created, it goes into an area of memory known as the heap.
- All objects live on the heap.
- Java heap is a garbage-collectible heap: When JVM can see that an object never be used again, that object becomes eligible for garbage collection.

Questions

- What if we need global variables and methods? How do we do that if everything has to go in a class?
- What is a Java program? What do you actually deliver?

Questions

- What if we need global variables and methods? How do we do that if everything has to go in a class?
  - Mark the variable as *public*, *static* and *final.*
- What is a Java program? What do we actually deliver?
  - We deliver the classes.
  - Put all the classes into a **J**ava **ar**chive file, a **.jar file**. Include a text file, called manifest, that defines which class in the jar holds the *main()* method that should be run.

How to design an inheritance structure?

- Which would be the subclass, and which would be the superclass?
- Or are they both subclasses to some other class?

An inheritance Example

An inheritance example

```
public class Doctor {

  boolean worksAtHospital;

  void treatPatient() {

    // perform a checkup
  }
}
```

```
public class FamilyDoctor extends Doctor {

  boolean makesHouseCalls;

  void giveAdvice() {
    // give homespun advice
  }
}
```

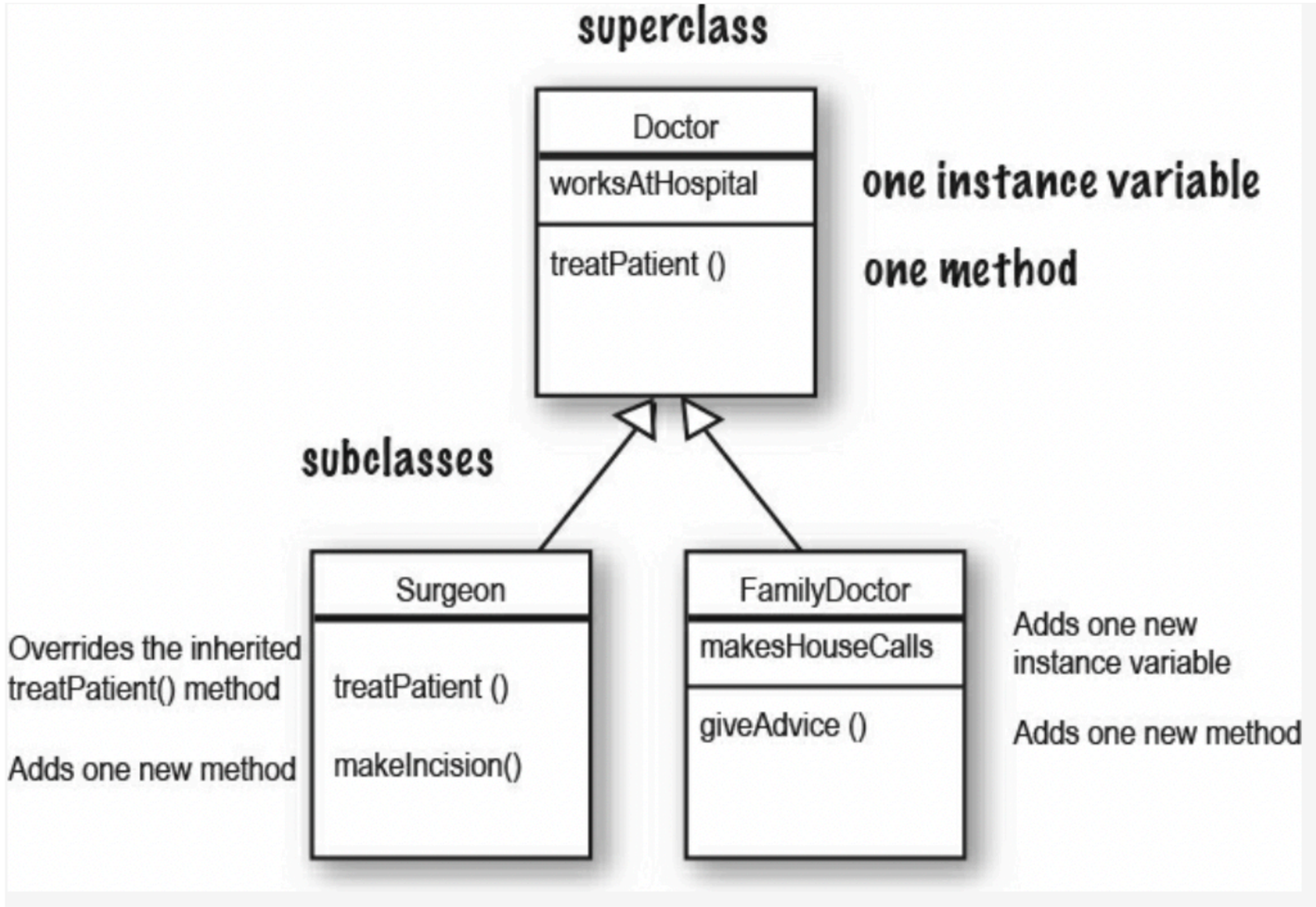
```
public class Surgeon extends Doctor {

  void treatPatient() {
    // perform surgery
  }

  void makeIncision() {
    // make incision (yikes!)
  }
}
```

An inheritance example

- How many instance variables does Surgeon have?
- How many instance variables does FamilyDoctor have?
- How many methods does Surgeon have?
- How many methods does FamilyDoctor have?
- Can a FamilyDoctor do makeIncision?
- Can a FamilyDoctor do treatPatient?



How to design an inheritance structure?

- Which would be the subclass, and which would be the superclass?
- Or are they both subclasses to some other class?

How to design an inheritance structure?

1. Look for objects that have common attributes and behaviors.
2. Design a class that represents the common state and behavior.
3. Decide if a subclass needs behaviors that are specific to that particular subclass type.
4. Look for more opportunities to use abstraction, by finding two or more subclasses that might need common behavior.

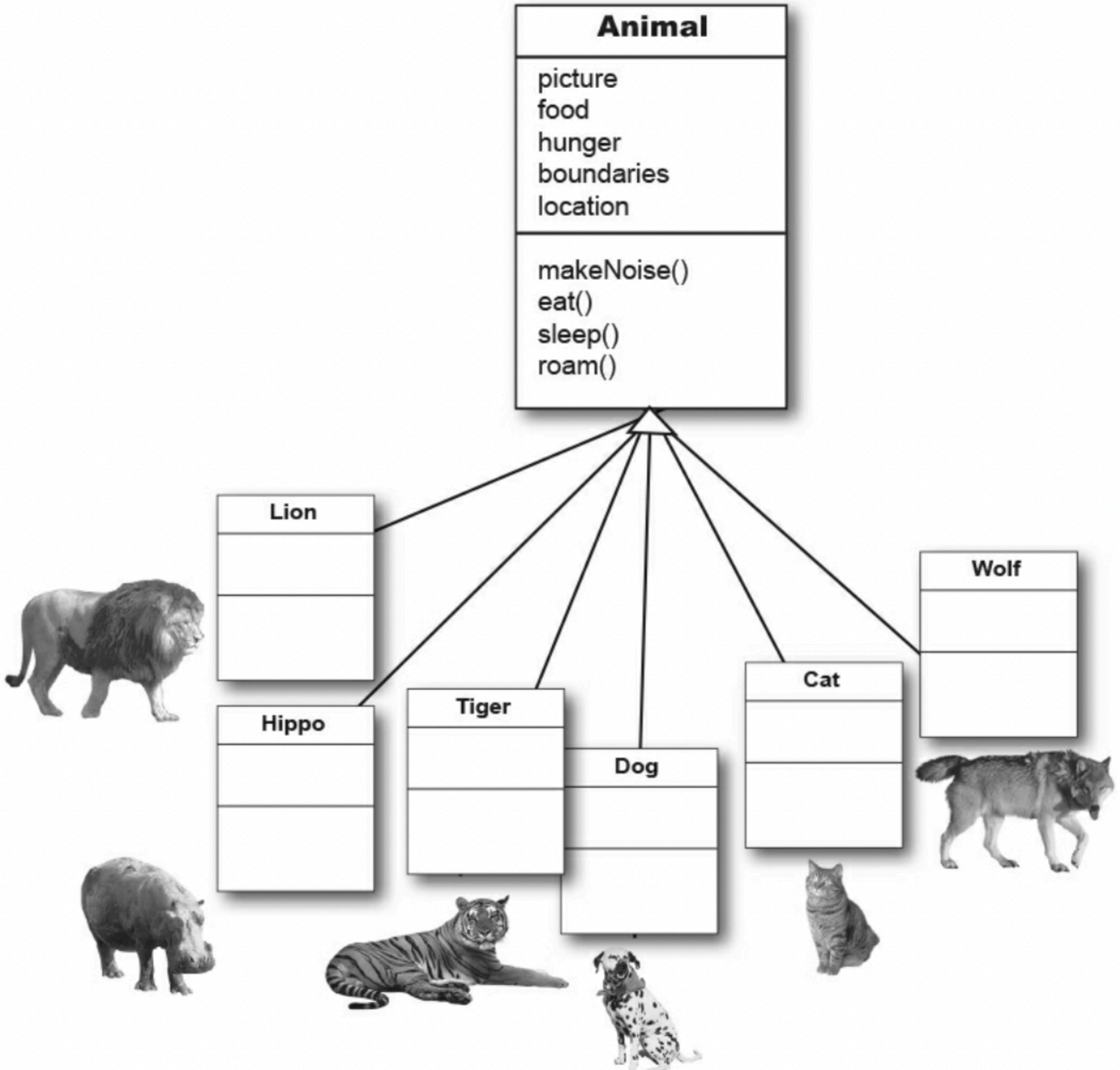
Example

- Imagine you are asked to design a simulation program that lets the user put a number of different animals into an environment to see what happens.
- We have been given a list of some of the animals, but not all.
- We want other programmers to be able to add new kinds of animals to the program at any time.

Example

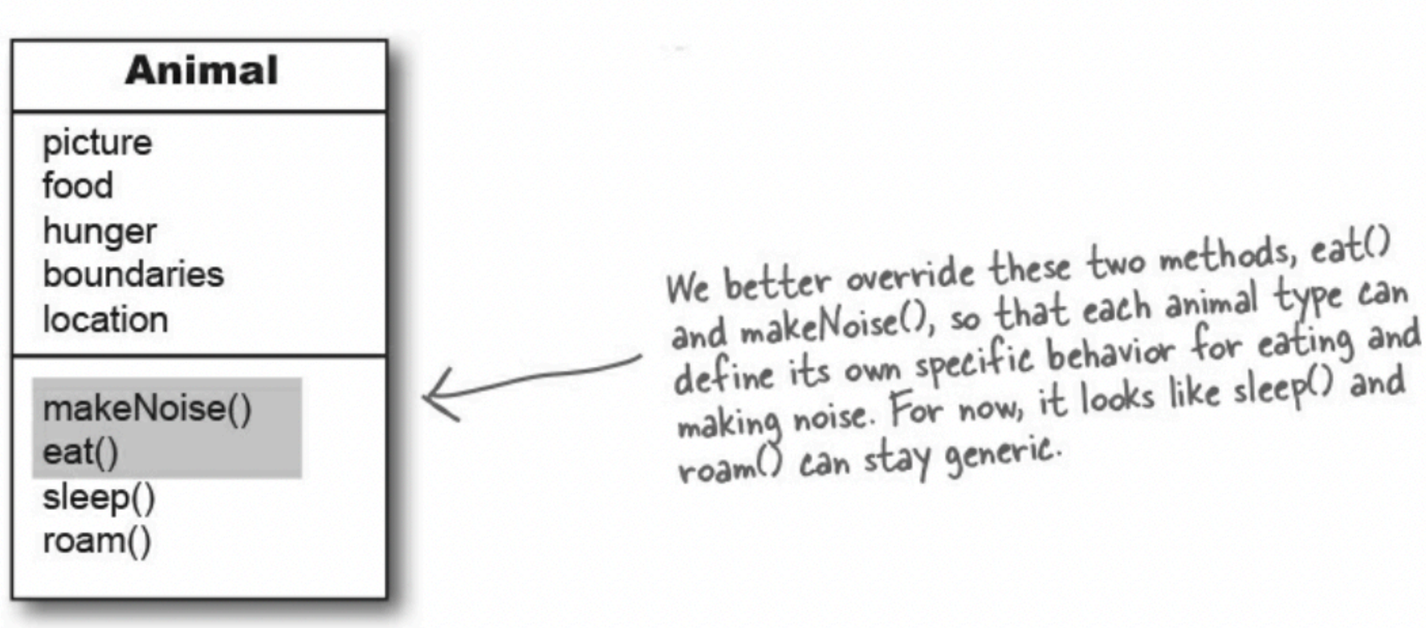
Example

1. Look for objects that have common attributes and behaviors.
2. Design a class that represents the common state and behavior.
3. Decide if a subclass needs behaviors that are specific to that particular subclass type.
4. Look for more opportunities to use abstraction, by finding two or more subclasses that might need common behavior.



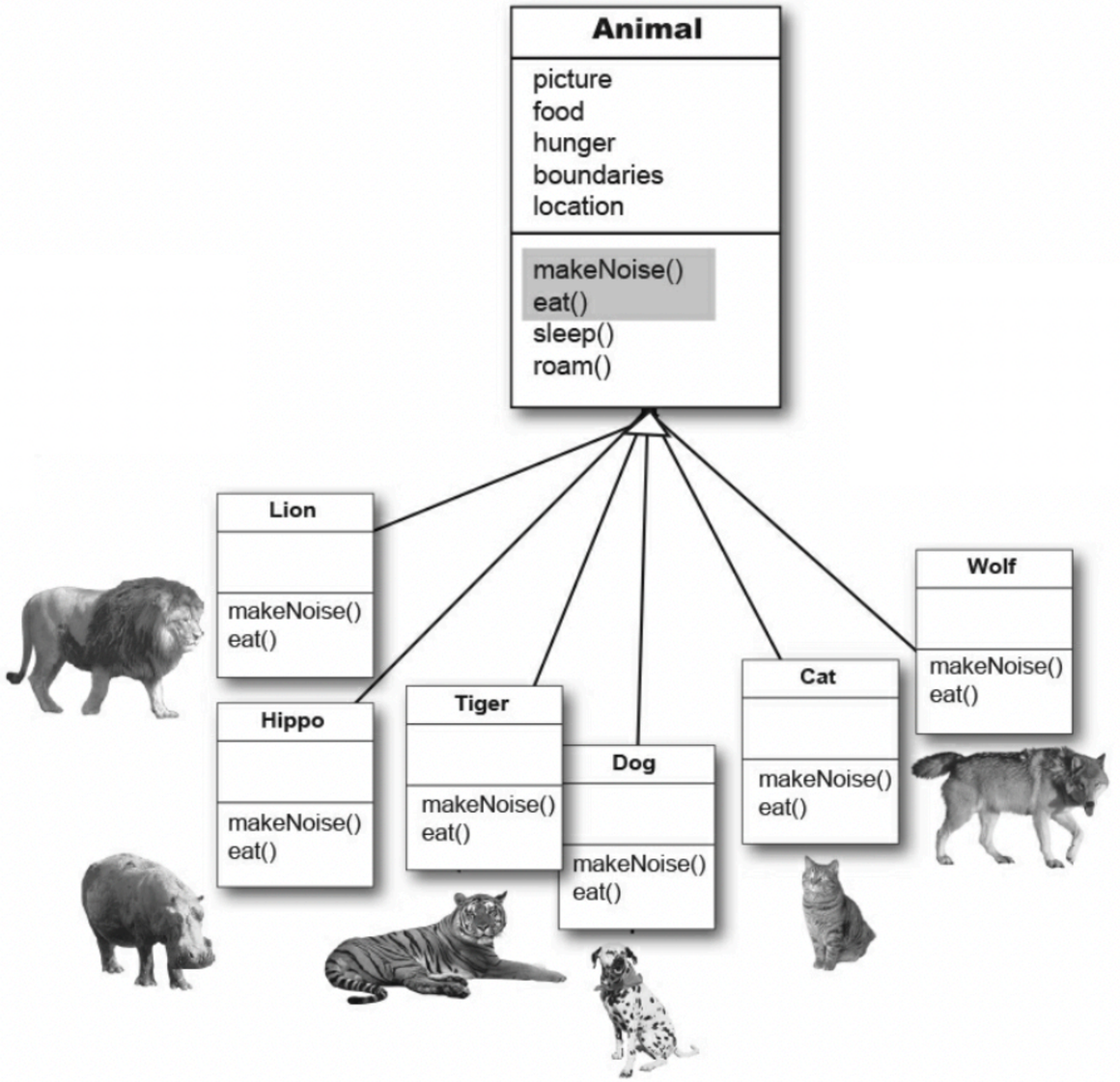
Example

1. Look for objects that have common attributes and behaviors.
2. Design a class that represents the common state and behavior.
3. Decide if a subclass needs behaviors that are specific to that particular subclass type.
4. Look for more opportunities to use abstraction, by finding two or more subclasses that might need common behavior.



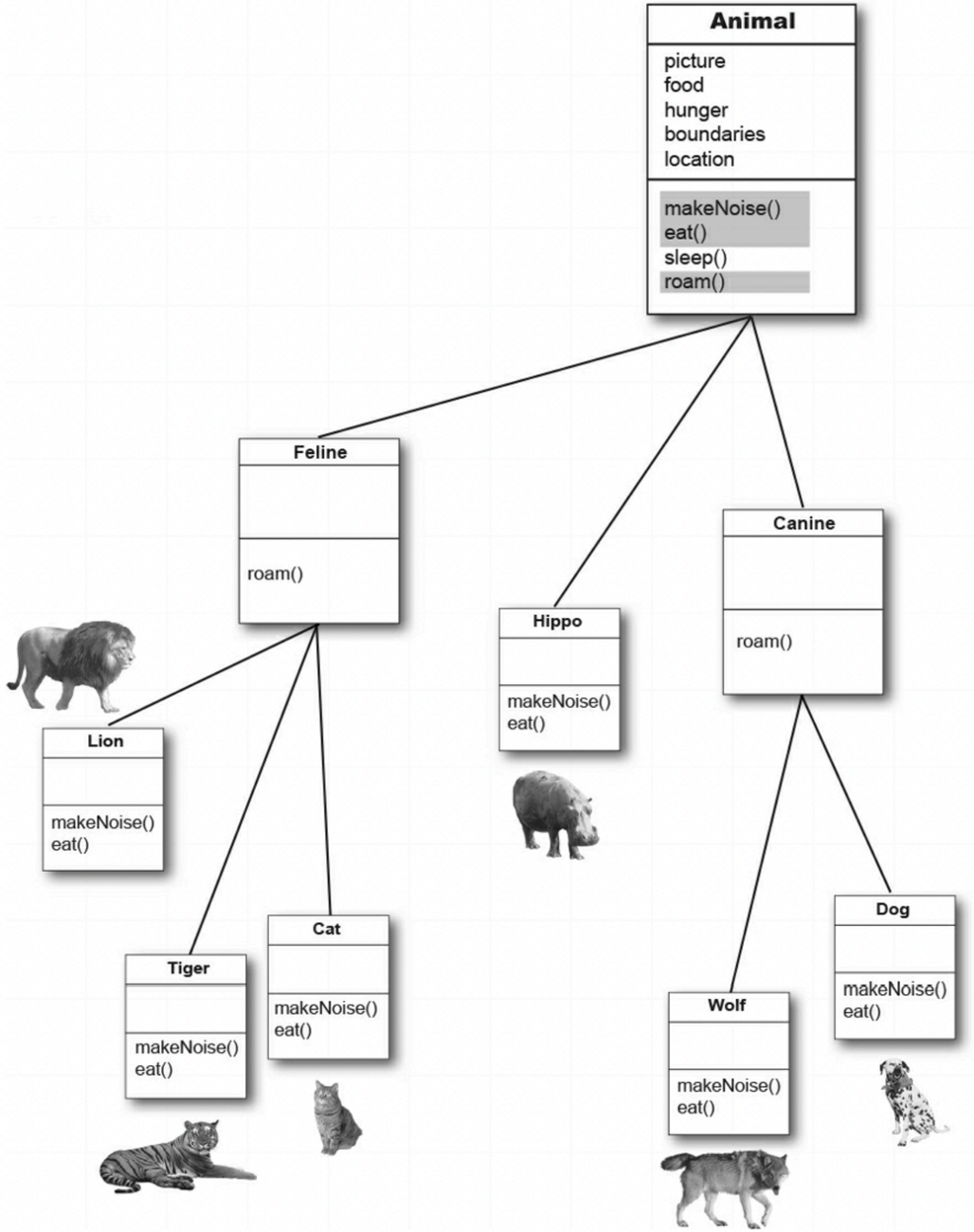
Example

1. Look for objects that have common attributes and behaviors.
2. Design a class that represents the common state and behavior.
3. Decide if a subclass needs behaviors that are specific to that particular subclass type.
4. Look for more opportunities to use abstraction, by finding two or more subclasses that might need common behavior.



Example

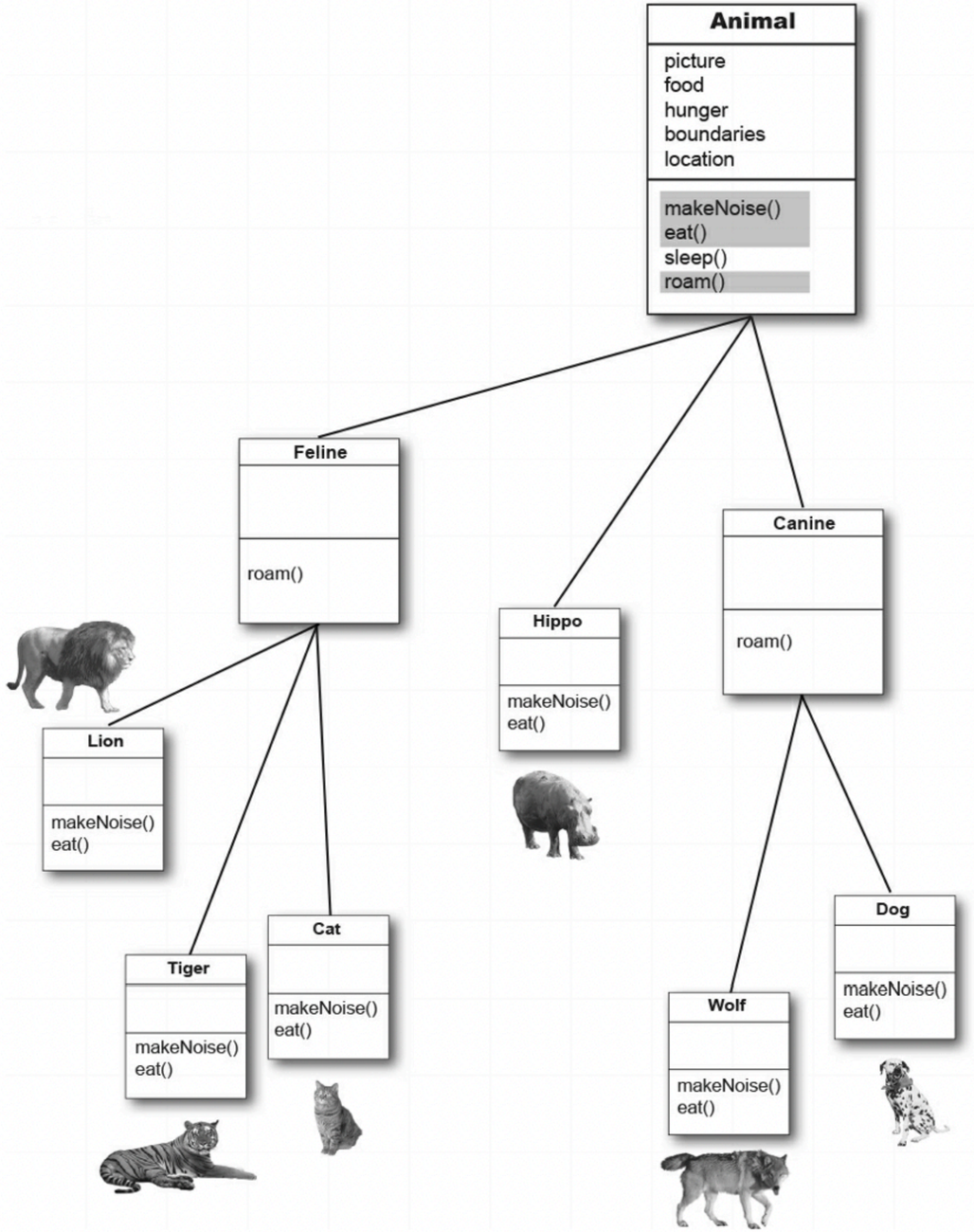
1. Look for objects that have common attributes and behaviors.
2. Design a class that represents the common state and behavior.
3. Decide if a subclass needs behaviors that are specific to that particular subclass type.
4. Look for more opportunities to use abstraction, by finding two or more subclasses that might need common behavior.



Question

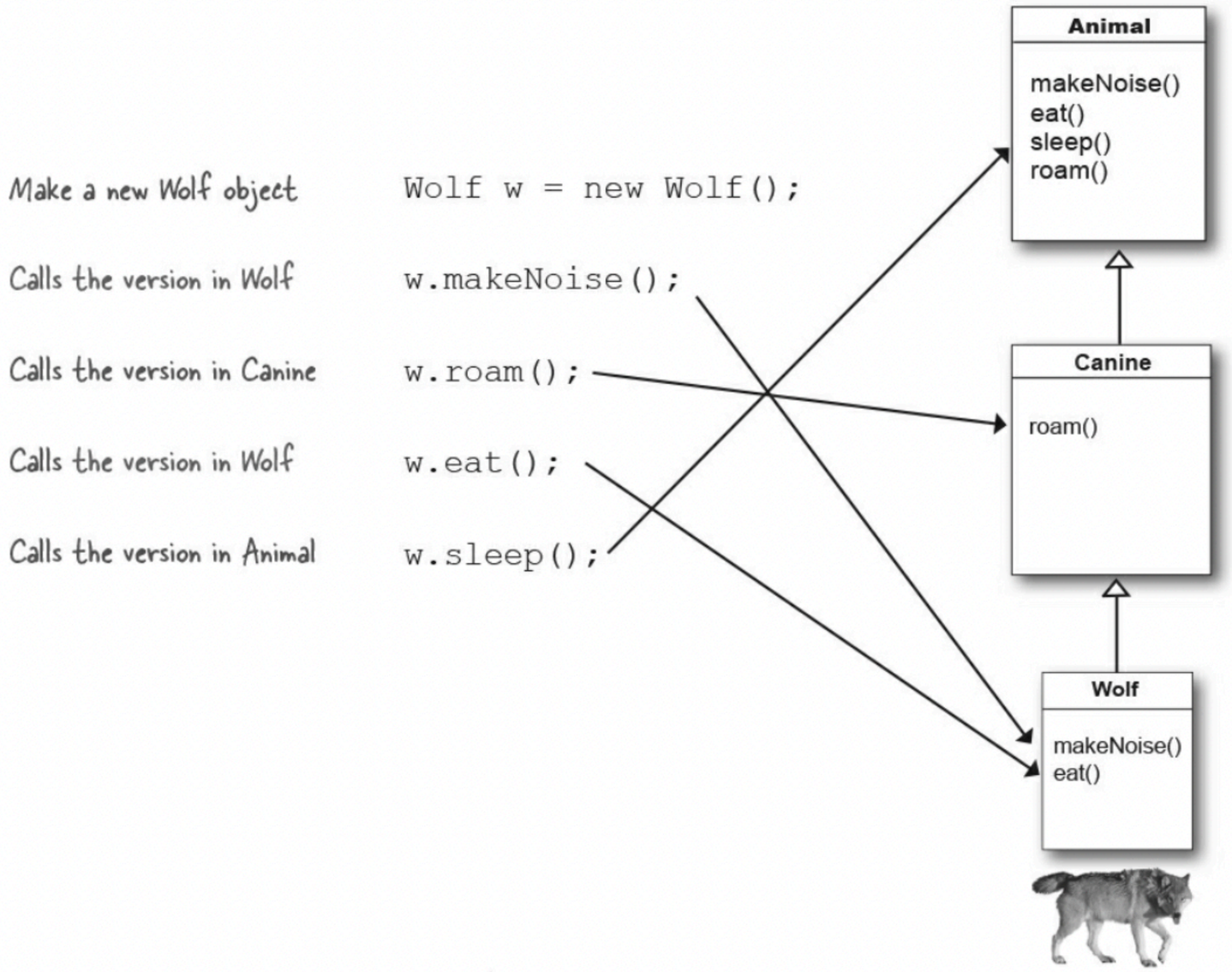
- Which method is called?

```
Wolf w = new Wolf();
w.makeNoise();
w.roam();
w.eat();
w.sleap();
```



Question

- Which method is called?
- When we call a method on an object reference we are calling the most specific version of the method for that object type.
- The JVM starts walking up the inheritance tree, starting at the class type we invoked the method on.

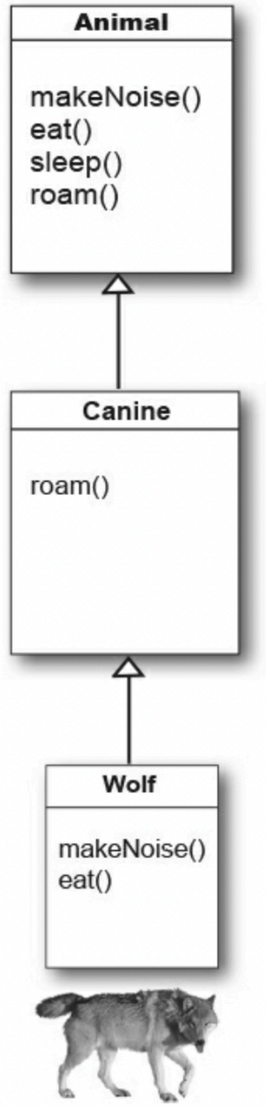


The IS-A test

- To know if you have designed your types correctly, ask “does it make sense to say X IS-A Y”.
  - Triangle is a shape
  - Cat is a feline
  - Surgeon is a doctor
  - Chair is a classroom

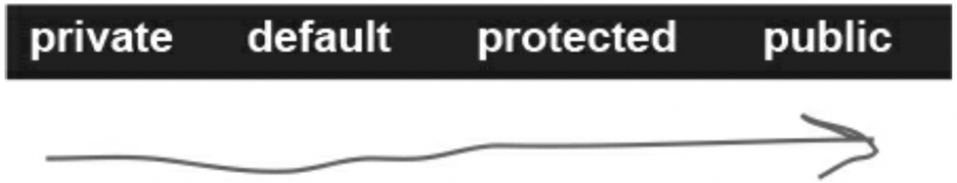
The IS-A test

- To know if you have designed your types correctly, ask “does it make sense to say X IS-A Y”.
  - Wolf is a canine
  - Wolf is an animal
  - Canine is an animal



Access levels

- Controls who sees what.
- Public members are inherited.
- Private members are **not** inherited.



Inheritance

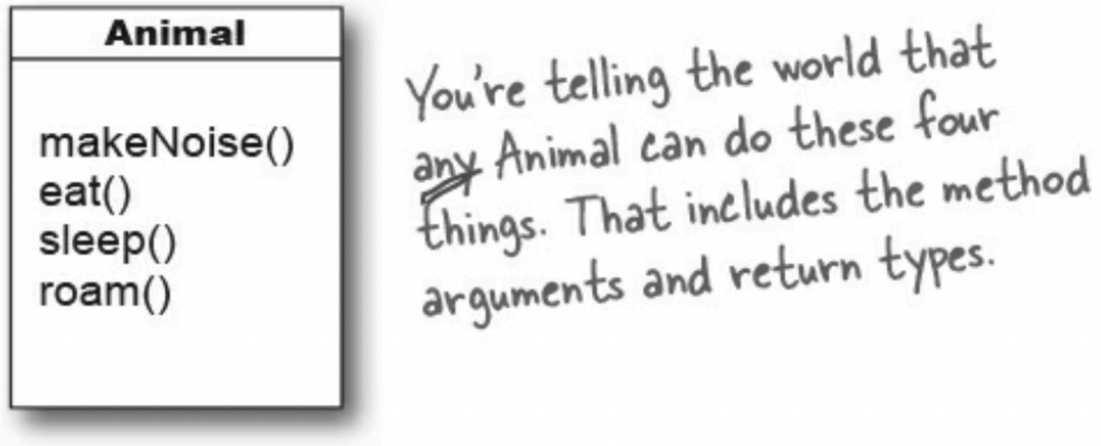
- Use it when one class is a more specific type of a superclass.
- Consider it when have behavior that should be shared among multiple classes of the same general type.
- DO **NOT** use inheritance just so that you can reuse code from another class, if the relationship between the superclass and the subclass is not an IS-A relationship.

Question

- Imagine you wrote special printing code in the Animal class and now you need printing code in the Potato class. You want to reuse the printing code. What would you do?

What does inheritance buy you?

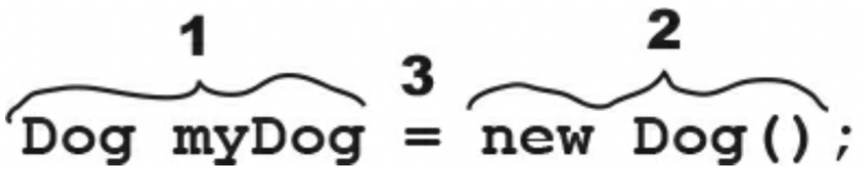
- You avoid duplicate code.
- You define a common protocol for a group of classes. In other words you establish a *contract*.



What does inheritance buy you?

- You avoid duplicate code.
- You define a common protocol for a group of classes. In other words you establish a *contract*.
- When you define a supertype for a group of classes, any subclass of that supertype can be substituted where the supertype is expected.

Polymorphism

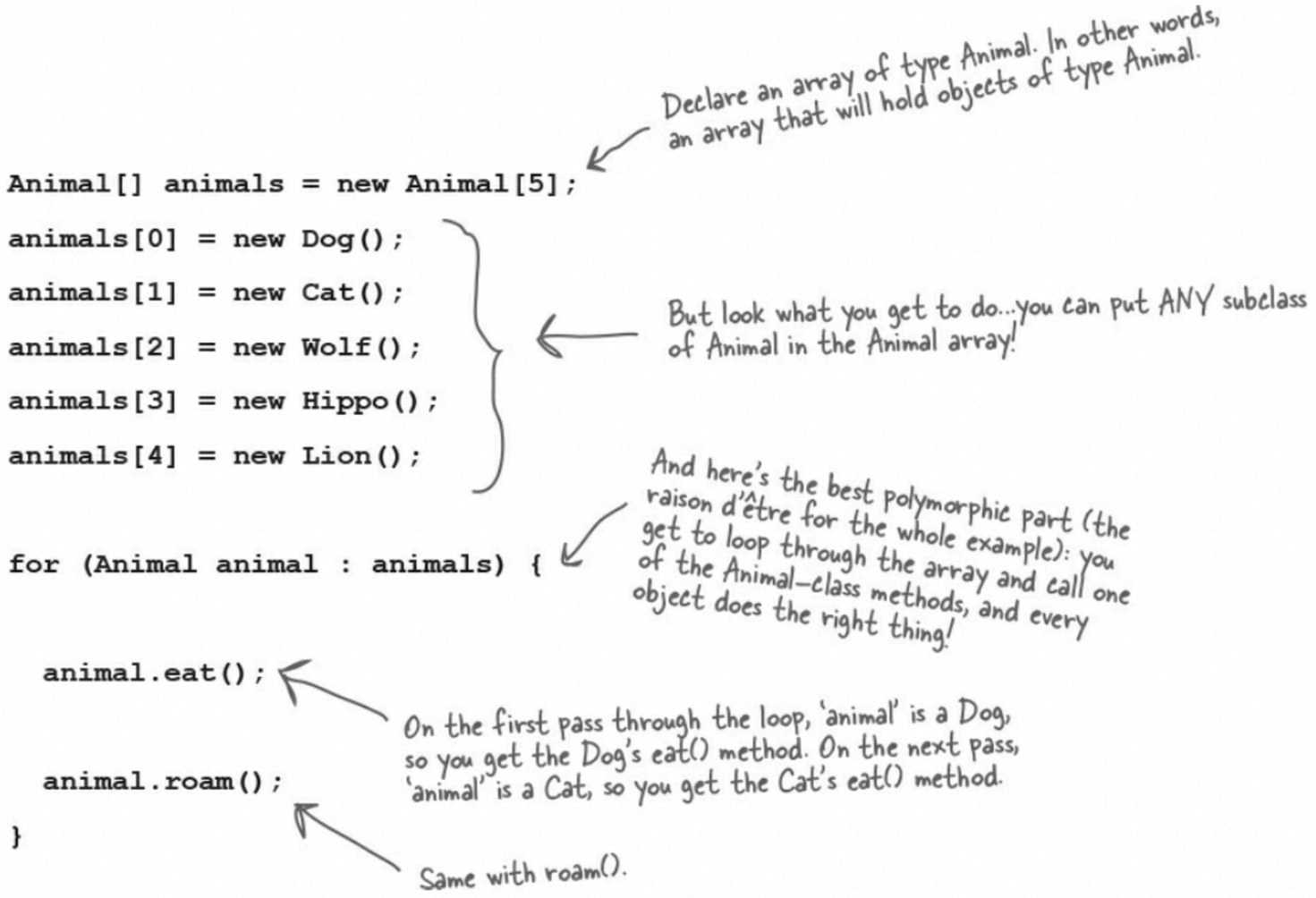


- 1 Declare a reference variable
- 2 Create an object
- 3 Link the object and the reference

Polymorphism

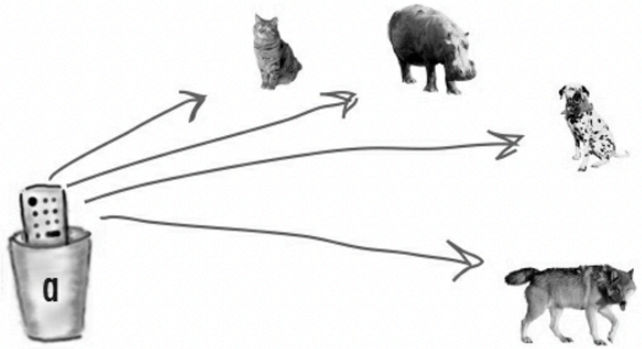
```
Animal myDog = new Dog();
```

Polymorphism



Polymorphism

- We can also have polymorphic arguments and return types.
- With polymorphism, you can write code that does not have to change when you introduce new subclass types into the program.




```
class Vet {
  public void giveShot(Animal a) {
    // do horrible things to the Animal at
    // the other end of the 'a' parameter
    a.makeNoise();
  }
}
```

The 'a' parameter can take ANY Animal type as the argument. And when the Vet is done giving the shot, it tells the Animal to makeNoise(), and whatever Animal is really out there on the heap, that's whose makeNoise() method will run.

```
class PetOwner {
  public void start() {
    Vet vet = new Vet();
    Dog dog = new Dog();
    Hippo hippo = new Hippo();
    vet.giveShot(dog);
    vet.giveShot(hippo);
  }
}
```

The Vet's giveShot() method can take any Animal you give it. As long as the object you pass in as the argument is a subclass of Animal, it will work.

Dog's makeNoise() runs

Hippo's makeNoise() runs