

4. Object-Oriented Programming Concepts

Java

Summer 2008

Instructor: Dr. Masoud Yaghini

Outline

- What Is an Object?
- What Is a Class?
- What Is Inheritance?
- What Is an Interface?
- What Is a Package?
- References



What Is an Object?

Real-world objects

- Real-world objects:
 - your dog
 - your desk
 - your television set
 - your bicycle
 -
- Real-world objects have two characteristics:
 - ***State***
 - ***Behavior***

Real-world objects

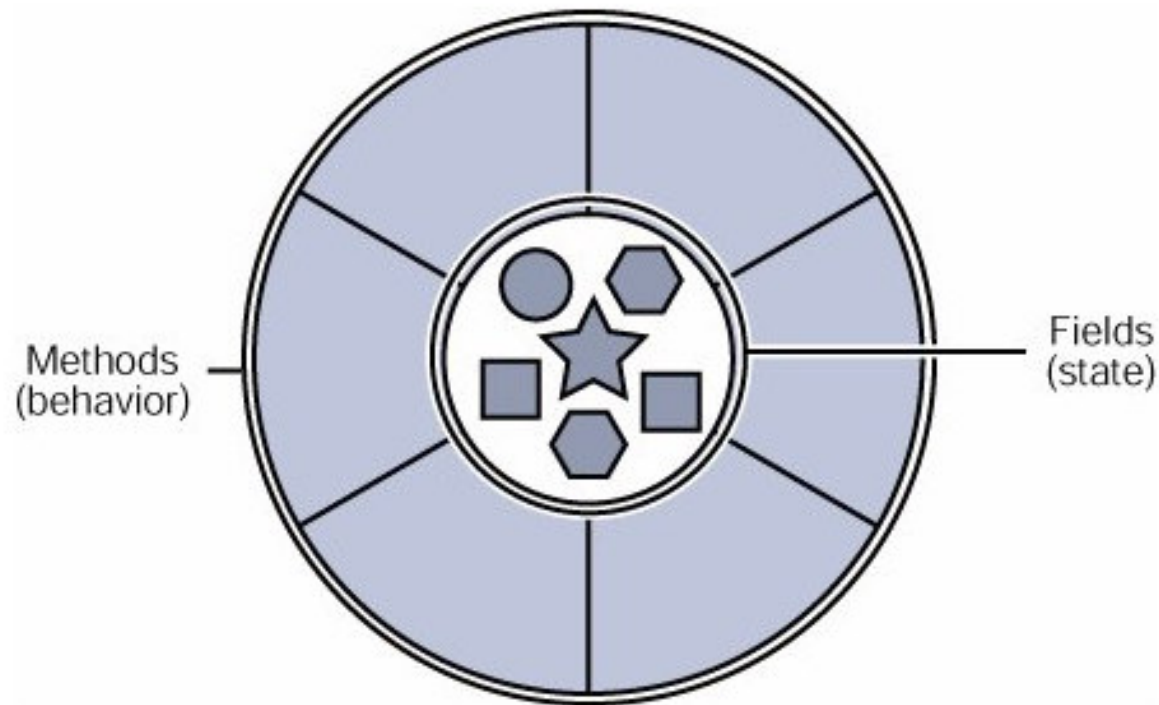
- Dogs:
 - States: name, color, breed, hungry
 - Behavior: barking, wagging tail
- Bicycles:
 - States: current gear, current pedal cadence, current speed
 - Behavior: changing gear, changing pedal cadence, applying brakes

Real-world objects

- Desktop lamp:
 - States: on, off
 - Behavior: turn on, turn off
- Desktop radio:
 - States: on, off, current volume, current station
 - Behavior: turn on, turn off, increase volume, decrease volume, scan

Software objects

- An object stores its state in ***fields*** (variables)
- An object represents its behavior through ***methods*** (functions)

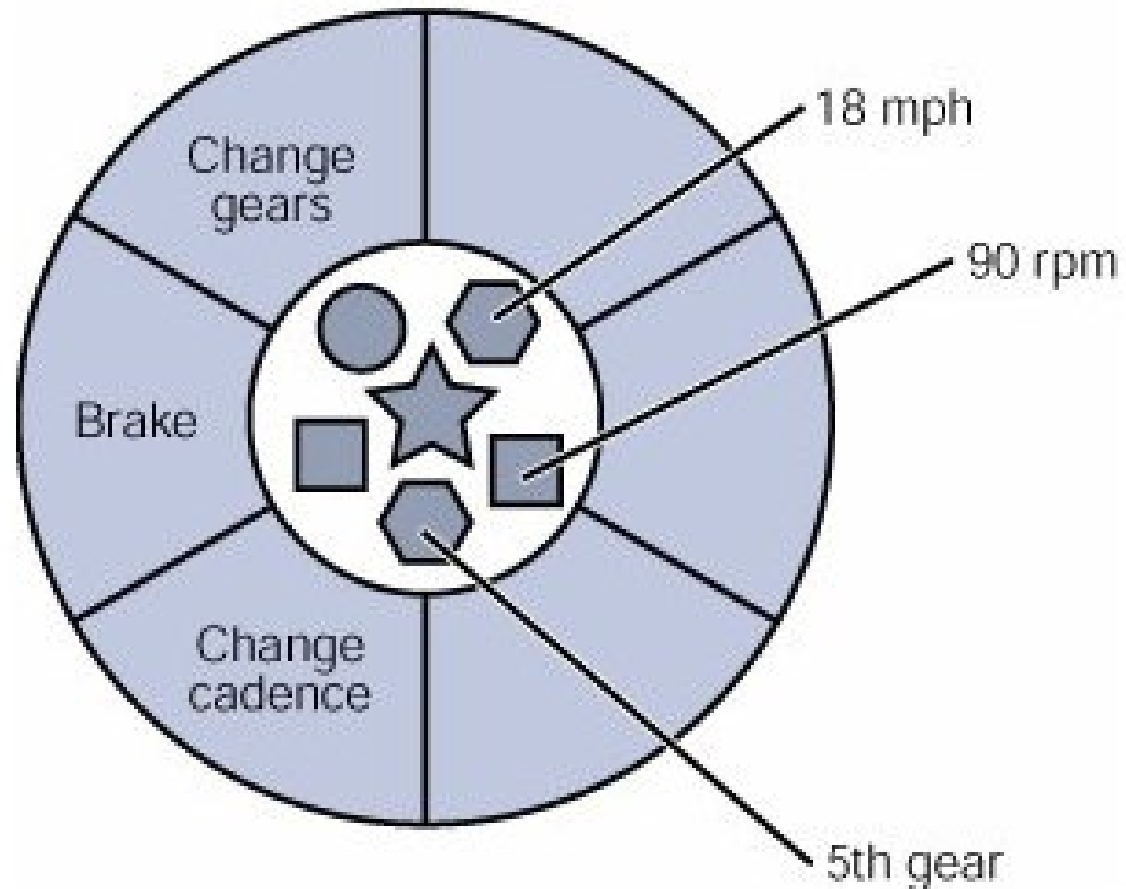
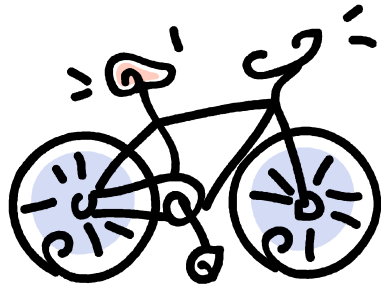


Data Encapsulation

- Methods operate on an object's internal state
- Hiding internal state and requiring all interaction to be performed through an object's methods is known as ***data encapsulation***.

Object-Oriented Programming Concepts

A bicycle modeled as a software object



Benefits of objects-oriented programming

- **Modularity:** The source code for an object can be written and maintained independently of the source code for other objects.
- **Information-hiding:** By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.
- **Code re-use:** If an object already exists (perhaps written by another software developer), you can use that object in your program.
- **Pluggability and debugging ease:** If a particular object turns out to be problematic, you can simply remove it from your application and plug in a different object as its replacement.



What Is a Class?

What Is a Class?

- A class is the blueprint from which individual objects are created.
- Each bicycle was built from the same set of blueprints and therefore contains the same components is an instance of the class of objects known as bicycles.

Bicycle class

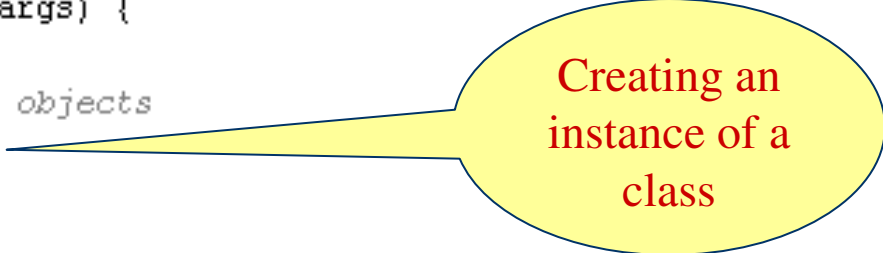
```
class Bicycle {  
  
    int cadence = 0;  
    int speed = 0;  
    int gear = 1;  
  
    void changeCadence(int newValue) {  
        cadence = newValue;  
    }  
  
    void changeGear(int newValue) {  
        gear = newValue;  
    }  
  
    void speedUp(int increment) {  
        speed = speed + increment;  
    }  
  
    void applyBrakes(int decrement) {  
        speed = speed - decrement;  
    }  
  
    void printStates() {  
        System.out.println("cadence:" + cadence + " speed:" + speed + " gear:" + gear);  
    }  
}
```

field

Method

BicycleDemo class

```
class BicycleDemo {  
    public static void main(String[] args) {  
  
        // Create two different Bicycle objects  
        Bicycle bikel = new Bicycle();  
        Bicycle bike2 = new Bicycle();  
  
        // Invoke methods on those objects  
        bikel.changeCadence(50);  
        bikel.speedUp(10);  
        bikel.changeGear(2);  
        bikel.printStates();  
  
        bike2.changeCadence(50);  
        bike2.speedUp(10);  
        bike2.changeGear(2);  
        bike2.changeCadence(40);  
        bike2.speedUp(10);  
        bike2.changeGear(3);  
        bike2.printStates();  
    }  
}
```



Creating an
instance of a
class

BicycleDemo class

- The output of this test prints the ending pedal cadence, speed, and gear for the two bicycles:

cadence:50 speed:10 gear:2

cadence:40 speed:20 gear:3



What Is Inheritance?

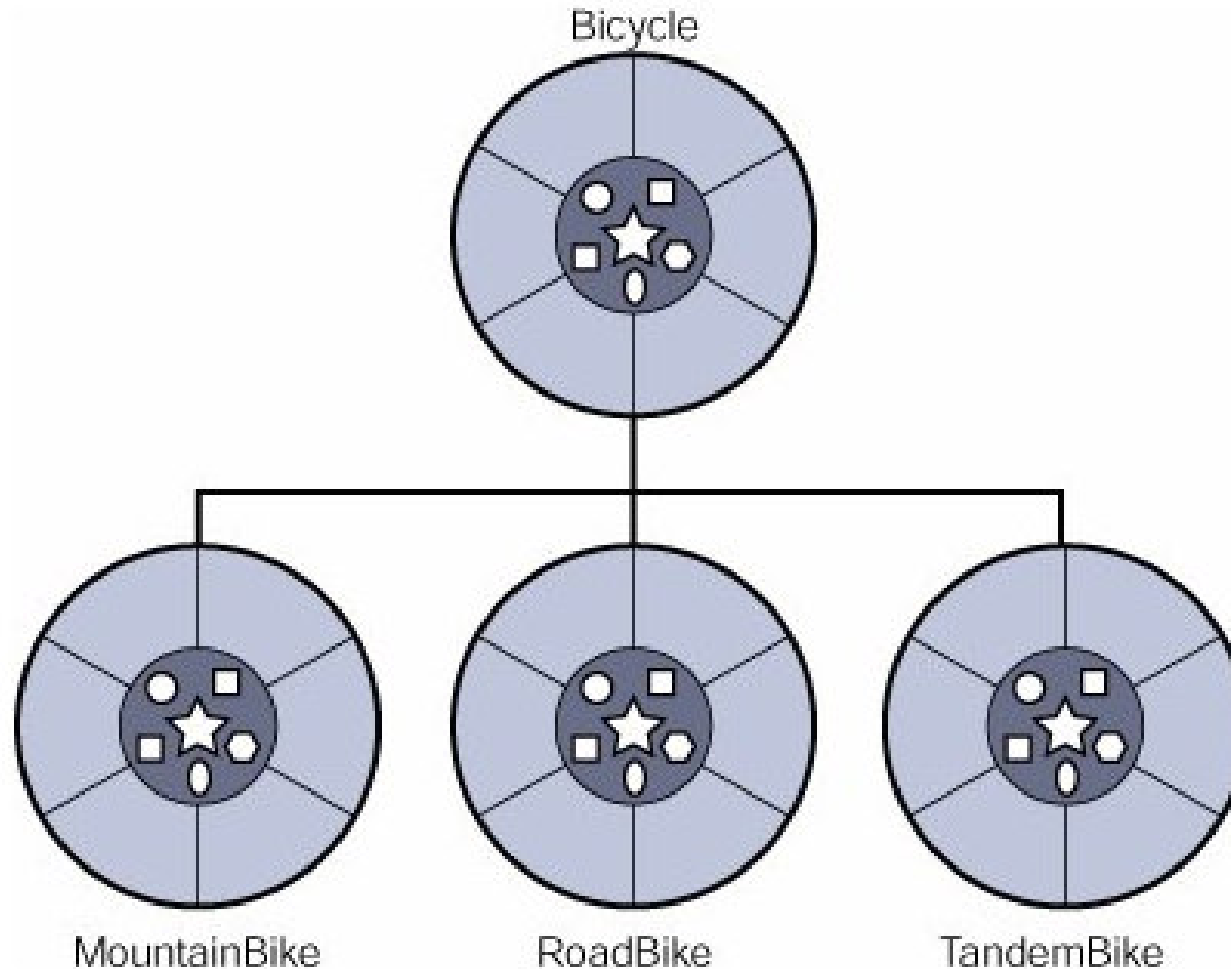
What Is Inheritance?

- Mountain bikes, road bikes, and tandem bikes, for example, all share the characteristics of bicycles (current speed, current pedal cadence, current gear).
- Yet each also defines additional features that make them different:
 - tandem bicycles have two seats and two sets of handlebars;
 - road bikes have drop handlebars;
 - mountain bikes have an additional chain ring, giving them a lower gear ratio.
- In this example, Bicycle now becomes the superclass of MountainBike, RoadBike, and TandemBike.

What Is Inheritance?

- Object-oriented programming allows classes to inherit commonly used state and behavior from other classes.
- In the Java programming language, each class is allowed:
 - to have one direct superclass, and
 - each superclass has the potential for an unlimited number of subclasses

A hierarchy of bicycle classes



Creating a subclass

- The syntax for creating a subclass:

```
class MountainBike extends Bicycle {  
  // new fields and methods defining a mountain bike  
  // would go here  
}
```

- This gives MountainBike all the same fields and methods as Bicycle, yet allows its code to focus exclusively on the features that make it unique.



What Is an Interface?

What Is an Interface?

- An **interface** is a group of related methods with empty bodies.
- A bicycle's behavior, if specified as an interface, might appear as follows:

```
interface Bicycle {  
    void changeCadence(int newValue);  
    void changeGear(int newValue);  
    void speedUp(int increment);  
    void applyBrakes(int decrement);  
    void printStates();  
}
```

What Is an Interface?

- To implement this interface, the name of your class would change (to `ACMEBicycle`, for example), and you'd use the `implements` keyword in the class declaration:

```
class ACMEBicycle implements Bicycle {  
// remainder of this class implemented as before  
}
```

What Is an Interface?

- If your class claims to implement an interface, all methods defined by that interface must appear in its source code before the class will successfully compile.



What Is a Package?



What Is a Package?

- A package is a namespace that organizes a set of related classes and interfaces.
- You can think of packages as being similar to different folders on your computer.
- The Java platform provides an enormous class library (a set of packages) suitable for use in your own applications.
- This library is known as the "Application Programming Interface," or "API" for short.

Application Programming Interface (API)

- API's packages represent the tasks most commonly associated with general purpose programming, for example:
 - A **String object** contains state and behavior for character strings;
 - A **File object** allows a programmer to easily create, delete, inspect, compare, or modify a file on the file system;
 - The **GUI objects** control buttons and checkboxes and anything else related to graphical user interfaces.
- This allows you, the programmer, to focus on the design of your particular application, rather than the infrastructure required to make it work.

Application Programming Interface (API)

- The Java Platform API Specification contains the complete listing for all packages, interfaces, classes, fields, and methods supplied by the Java Platform 6, Standard Edition:
 - <http://java.sun.com/javase/6/docs/api/>



References



References

- S. Zakhour, S. Hommel, J. Royal, I. Rabinovitch, T. Risser, M. Hoeber, **The Java Tutorial: A Short Course on the Basics**, 4th Edition, Prentice Hall, 2006. (Chapter 2)



The End