Java in a Nutshell

Unil | HEC | dop lab | Benoît Garbinato
distributed object programming lab

Goals of this lesson

To remind you about:

- object-oriented programming (OOP)
- the Java language & platform

Object-Oriented Programming

- Any object-oriented Programming language should feature:
  - ☑ encapsulation
  - ☑ inheritance
  - ☑ polymorphism
- Java is such an object-oriented programming language

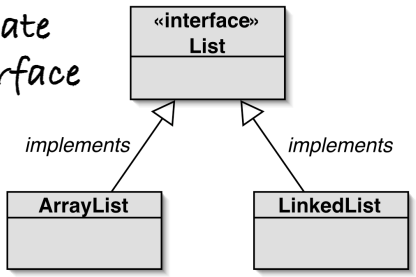
Encapsulation (1)

Encapsulation is about distinguishing specification from implementations

- The specification expresses what all objects of some type are expected to do
- An implementation expresses how some objects of that type are doing it

Encapsulation (2)

- In Java, a class defines both a specification (type) and an implementation of that specification
- In Java, an interface defines a "pure" specification
- It it thus impossible to instantiate (create an instance) of an interface
- One or more Java classes can implement a given interface



Inheritance (1)

- Types and subtypes express specification relationships, i.e., relevant design relationships
- Classes and subclasses express implementation relationships and are irrelevant at the design level
- In Java, a class inheritance relationship defines both a subtype and a subclass relationship
- In Java, an interface inheritance relationship is merely a synonym of subtype relationship

Inheritance (2)

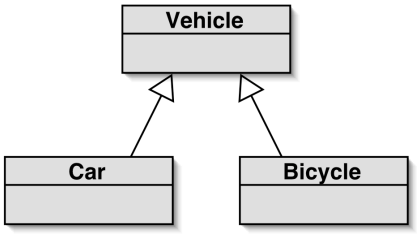
class inheritance ⇔ subtyping + subclassing

interface inheritance ⇔ subtyping

Polymorphism (1)

Substitution Principle:

An object of some subtype of type T can be used wherever an object of type T is required




```
Vehicle v1 = new Vehicle();
Vehicle v2 = new Car();
Vehicle v3 = new Bicycle();
...
myScreen.drawInColor(v1);
myScreen.drawInColor(v2);
myScreen.drawInColor(v3);
...
```

Polymorphism (2)

- Polymorphic variables can store objects of varying types
- The declared type of a variable is its static type
- The type of the object a variable refers is its dynamic type
- The Java compiler checks for static-type violations
- The Java runtime checks for dynamic-type violations

```
Car c = new Car();
Vehicle v = new Car();
```

what is the type of c and v?

Polymorphism (3)

- Methods calls are also said to be polymorphic, meaning that the dynamic type of the variable rather than its static type determines the method to be called
- The method of the subclass is said to override the method of the superclass

```
class Vehicle {
  void print(){ System.out.println("I am a vehicle");}
}
```

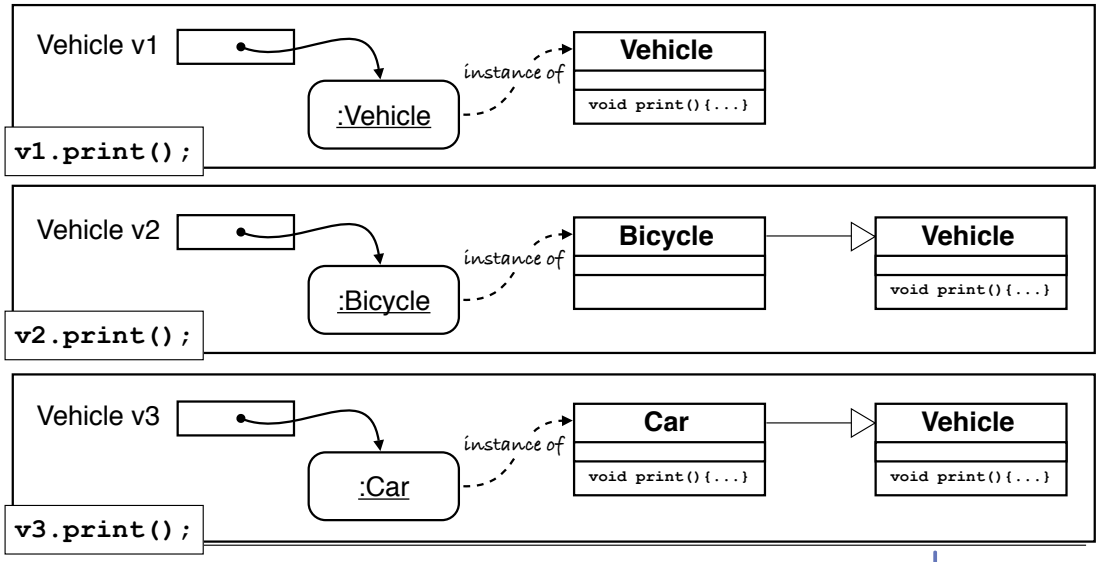
```
class Bicycle extends Vehicle {}
```

```
class Car extends Vehicle {
  void print(){ System.out.println("I am a car");}
}
```

```
Vehicle v1 = new Vehicle();
Vehicle v2 = new Bicycle();
Vehicle v3 = new Car();

v1.print(); v2.print(); v3.print();
```

what gets printed?

Polymorphism (4)



A simple example (1)

```
public interface MouseListener {
  public void mouseClicked(MouseEvent e);
  // Invoked when the mouse has been clicked on a component.

  public void mouseEntered(MouseEvent e);
  // Invoked when the mouse enters a component.

  public void mouseExited(MouseEvent e);
  // Invoked when the mouse exits a component.

  public void mousePressed(MouseEvent e);
  // Invoked when a mouse button has been pressed on a component.

  public void mouseReleased(MouseEvent e);
  // Invoked when a mouse button has been released on a component.
}
```

A simple example (2)

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

public class Spot extends Applet implements MouseListener {
  private java.awt.Point clickPoint = null;
  private static final int RADIUS = 7;

  public void init() {
    addMouseListener(this);
  }
  public void paint(Graphics g) {
    g.drawRect(0, 0, getSize().width - 1,
                     getSize().height - 1);
    if (clickPoint != null)
      g.fillOval(clickPoint.x - RADIUS,
                 clickPoint.y - RADIUS,
                 RADIUS * 2, RADIUS * 2);
  }
  public void mousePressed(MouseEvent event) {
    clickPoint = event.getPoint();
    repaint();
  }
     public void mouseClicked(MouseEvent event) {}
     public void mouseReleased(MouseEvent event) {}
     public void mouseEntered(MouseEvent event) {}
     public void mouseExited(MouseEvent event) {}
}
```

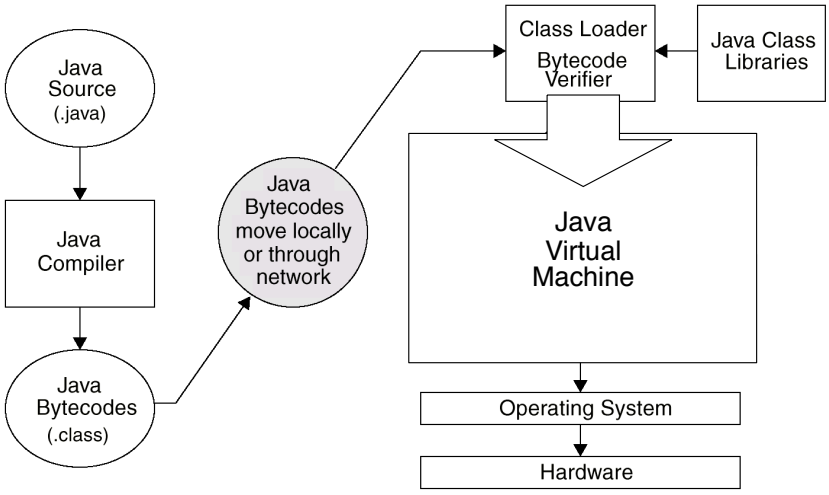
```
<HTML>
<TITLE>
Spot Applet
</TITLE>

<BODY>
<APPLET
CODE=Spot.class
        WIDTH=150
        HEIGHT=150>
</APPLET>
</BODY>
</HTML>
```


Java as programming platform

The Java platform consists of...

- The specification of a programming language
- The specification of a rich collection of standard Application Programming Interfaces (APIs)
- The specification of a virtual machine (bytecodes)
- Various implementations, e.g., one from a Sun Microsystems (JavaSoft), but also others (IBM, BEA Virtual Machines, etc.)

Development process



Java Development Kit (JDK)

- JDK is a compile-time environment that offers:
  - a reference implementation of the Java language;
  - a reference implementation of the core Java APIs;
  - various development tools, e.g., javadoc, javah, etc.
- JDK is also a runtime environment that offers:
  - a reference implementation of the Java virtual machine with:
    - incremental garbage collection,
    - green & native threads,
    - just-in-time compilation,
    - etc.

Questions?