Inheritance, Polymorphism, and Interfaces

New classes from existing classes

```
public class NewClass extends ExistingClass {
    /*  . . .  */
}
```

- `NewClass` is the **subclass**; `ExistingClass` is the **superclass**
- `NewClass` **inherits** everything from `ExistingClass`
- `NewClass` can add new features

Other terminology

- **Superclass** is to **subclass** as
  - **Base class** is to **derived class**
  - **General class** is to **specific class**
- **Inheritance** is also called
  - **subclassing**
  - **specialization**
  - **extension**

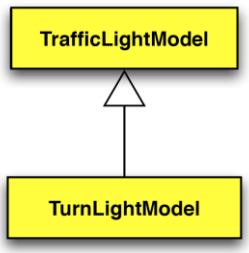
Example

- Extend the traffic light class to add a left turn arrow

```
[( )  ( )  ( )  (<)]
```


Visualizing inheritance

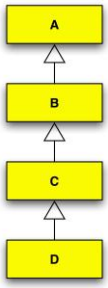
- The Unified Modeling Language (UML)



The *is a* relationship

- An instance of a subclass *is a* instance of a superclass as well
  - A `TurnLightModel` object *is a* `TrafficLightModel` object, since `TurnLightModel` is a subclass of `TrafficLightModel`
- An instance of a subclass can be used in any context that requires an instance of a superclass of that object

Is a transitivity



8

Polymorphism

- When a method is called on an object reference, the code that executes depends on the type of the object
- Because of inheritance and the *is a* relation, we cannot always know at compile-time what code will be executed when a method is called
- Polymorphism is the runtime determination of which method to execute based on the exact type of the object on whose behalf the the method is called

9

Single Inheritance

- In Java, every class except `java.lang.Object` has exactly one superclass
  - `Object` has no superclass
- Some languages such as C++ and Python support multiple inheritance
- Java supports something similar to multiple inheritance via interfaces

10

Interface

- An interface is specified with the `interface` keyword:

```
interface IShape {
    double perimeter();
    double area();
    void draw(Graphics g);
}
```

11

An Interface is like an Abstract Class

- Any interface is similar to an abstract class:
  - A client may not instantiate an interface
- **All** methods in an interface are implicitly `abstract` and implicitly `public`
- An interface may not specify any data (fields)
  - Although **constant** data is okay

```
interface IShape {
    double perimeter();
    double area();
    void draw(Graphics g);
}
```

12

Implementing an Interface

- An interface specifies a *protocol* and provides no *implementation*
- In order to make use of an interface a client must define a class that *implements* the interface

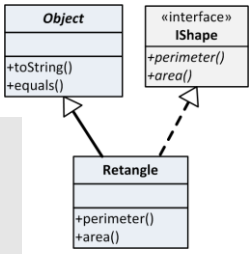
```
interface IShape {
    double perimeter();
    double area();
}
```

```
class Rectangle implements IShape {
   private double width;
   private double height;
   public Rectangle(double w, double h) {
       width = w; height = h;
   }
   public double perimeter() {
       return 2*width + 2*height;
   }
   public double area() {
       return width*height;
   }
}
```

Implementing an Interface UML

```
interface IShape {
    double perimeter();
    double area();
}
```

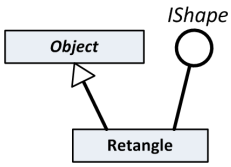
```
class Rectangle implements IShape {
   private double width;
   private double height;
   public Rectangle(double w, double h) {
       width = w; height = h;
   }
   public double perimeter() {
       return 2*width + 2*height;
   }
   public double area() {
       return width*height;
   }
}
```



14

Simpler Diagram

- Useful when the interface details are known by the viewer
- Can make complex diagrams easier to read



15

Interface specifies *can-do*

- An implementing class exhibits an is-a relationship with its interface
- Since the interface provides no implementation (no data), there is nothing to inherit except method signatures
- These method signatures represent what instances of the class *can do*
- A better description is to say an interface specifies what instances of implementing classes *can do*

16

Multiple Implementation of Interfaces

- Any class we define must inherit from exactly one superclass (the single inheritance rule)
- A class may implement as many interfaces as we like
- Java supports multiple implementation of interfaces

17