Introduction to Inheritance & Processing

CSCI-UA 0101
Instructor: Randy Shepherd
Inheritance
What Is Inheritance?

- Different kinds of objects often have a certain amount in common with each other.

- Mountain bikes, road bikes, and tandem bikes all share the characteristics of bicycles (current speed, current gear, etc.)

- Yet each also defines additional features that make them different: tandem bicycles have two seats and two sets of handlebars;
What Is Inheritance?

- Object-oriented programming allows classes to inherit commonly used state and behavior from other classes.

- In this example, Bicycle now becomes the superclass of MountainBike, RoadBike, and TandemBike

- In Java, each class is allowed to have one direct superclass, and each superclass has the potential for an unlimited number of subclasses.

Adapted from The Java™ Tutorials
What Is Inheritance?

Adapted from The Java™ Tutorials
What Is Inheritance?

- The syntax for creating a subclass is simple.

- At the beginning of your class declaration, use the `extends` keyword, followed by the name of the class to inherit from

```java
public class MountainBike extends Bicycle {
    // new fields and methods defining
    // a mountain bike would go here
    // .. and MountainBike 'inherits' somes
    // methods and data from Bicycle
}
```
What Is Inheritance?

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

• In this way we get ‘code reuse’.

• It allows us to find a class which implements roughly what we need, and “extend” it to add and modify it to our specific purpose.

Adapted from The Java™ Tutorials
What Is Inheritance?

• We can even extend our MountainBike..

```java
public class CrossCountryMountainBike extends MountainBike {
    // inherits Bicycle methods and any methods
    // that MountainBike has added.
}
```

• So we can build ‘hierarchies’ of inheritance.

Adapted from The Java™ Tutorials
java.lang.Object

• In fact, if you do not specify an ‘extends’ declaration in your code the compiler will add one for you.

• By default, all objects extend java.lang.Object.

• All classes extend Object directly or indirectly.

• The java.lang.Object class is the ‘root of the class hierarchy.’
java.lang.Object

class A {
}

public class SimpleClass {

public SimpleClass() {
    A a = new A();
    a;
}

}
In an inheritance hierarchy, classes are known as "superclasses" and "subclasses".

For some class B that extends A:
  - A is a superclass of B
  - B is a subclass of A
  - A is a subclass of Object
  - B is also a subclass of Object (via A)

Sometimes also called ‘parent’ and ‘child’ classes.
With me so far?

• A subclass inherits certain properties (data) and methods from the superclass.

• But the subclass can also …
  
  • Add new properties (instance variables)
  
  • Add new methods
  
  • Override the methods of the superclass

• We do this because we want to reuse code and code reuse is a ‘good thing™’
Overriding

• But suppose we extend a class and its method does not do exactly what we need?

• For example, suppose we extended Bicycle and created MotorizedBicycle

• Conceptually, a ‘pedal’ method for Bicycle would not work properly for a MotorizedBicycle.
Overriding

• This is called method overriding.

• We declare a pedal method in MotorizedBicycled that knows how to function for a bike with a motor.

• When users call the pedal method on our MotorizedBicycled, it calls our special “overridden” implementation.

• We are effectively ‘hiding’ Bicycle’s pedal method with our own.
Overriding

• We’ve seen an example of a method override in the Snape exercise with toString

• Potion needed to have a toString method

• And since the one for Object is useful to exactly nobody ever in the history of programming we implemented our own.
Again.. with me?

- We can extend classes to reuse their powers.

- We can specialize our extensions to have behaviors specific to our new class by overriding.
Processing
What is Processing

• A ‘library’ of code that can be used to create graphical programs.

• A teaching tool for programming.

• Developed in part here at NYU’s ITP program.

  • https://processing.org/overview/

  • https://processing.org/tutorials/gettingstarted/

• The documentation is targeted at the beginner
How to Install Processing

- You set it up in an earlier homework.
- You may need to follow those instructions again.
- They are listed on the website.
How to Use Processing

• ‘import’ the PApplet

• Just like importing Scanner.

• Declares your intention to use something from the Processing library.
• ‘extend’ the PApplet

• Thus inheriting all of PApplet’s methods (including a main method)
How to Use Processing

```java
import processing.core.PApplet;

public class ProcessingExample00 extends PApplet {

    public void draw() {
        rect(random(0, 500), random(0, 500), 10, 10);
    }
}
```

- Then we ‘override’ its draw() method.
- We are going to ‘draw’ to the ‘canvas’ whatever we would like.
How to Use Processing

```java
import processing.core.PApplet;

class ProcessingExample00 extends PApplet {

    public void draw() {
        rect(random(0, 500), random(0, 500), 10, 10);
    }
}
```

- The in our draw() implementation we use other methods of PApplet to write things to the canvas.

- Note that draw gets invoked in a loop by the PApplet class.
Enough talk...