July 7, 2010

Review of the terms from last class

- Class: The plan that defines an object.
- Field: An attribute of an object.
- Method: An action that can be performed on an object.
- Constructor: The very first method which is called when a class is instantiated. Returns a generic version of the object.
- Parameter: An input to a method
- Return type: The class or primitive data structure that a method will compute and return.
- private field: a field that cannot be seen outside of the class
- public field: a field that can be seen outside of the class. This is discouraged.
- static field: a field associated with the class. All objects can access this field.
- non-static field: a field associated with an individual instantiated object. It is owned by the object.
- static method: a method associated with a class. It can only access and mutate static fields.
- non-static method: a method associated with an object. It can access and mutate static and non-static fields.

Creating Subclasses

Yesterday we talked about code reuse. Java's inheritance features allow us to expand on this idea. Sometimes we need a class that has a few minor extra features to an already written class. By _extending_ an old class, we get all of the benefits of already written code. A class can be built on top of another class using the extends keyword.

We refer to inherited classes as “subclasses” or “child class”. We refer to older classes as “superclasses” or “parent class” or “base class”. By creating extended classes we are using _specialization_, which means that a child is a specialized version of a parent. We use the words is a to denote anything that is a specialized version of something else. If you can speak the sentence ”[objectA] is a [objectB]” and that sentence makes sense, then you can move forward and create an inherited class.

- Superman is a superhero.
- A horse is a mammal.
- A car is a vehicle.

Extend a class to created a specific version of a class which already exists, that way you take advantage of all of the methods that have been written using that class.

Public, Private, Protected, Package

This list is ordered from most restrictive permission to least restrictive permission.

- private means that only methods within this class can access this method or field.
- protected means that methods found in this class or classes that extend this class may access this field.
- Package (the default, has no keyword) means that any method in any class found in this directory may access this method or field.
- public means that any method in any other class can access this method or field.

Example of an Extended Class

Before we can begin, we need a base class. We'll create our own, but in truth we could have used in class already found in Java. For our purposes, let's begin with a vehicle.

Vehicle.java

```
class Vehicle {

        int wheels;

        int seats;

        int speed;
}
```

```
	Vehicle() {
		this.wheels = 0;
		this.seats  = 0;
		this.speed  = 0;
	}

	void setSpeed(int speed) { this.speed = speed; }
	void setWheels(int wheels) { this.wheels = wheels; }
	void setSeats(int seats) { this.seats = seats; }

	int getSpeed() {
		return speed;
	}
}
```

Bicycle.java

```
class Bicycle extends Vehicle {
	int gears;

	Bicycle (int gears) {
		setWheels(2);
		setSeats(1);
		this.gears = gears;
	}

	void setGears(int gears) { return this.gears = gears; }
	int getGears() { return gears; }
}
```

BikeTest.java

```
public class BikeTest {
	public static void main(String[] args) {
		Bicycle b = new Bicycle(1);
		b.setGears(2);
		b.setSpeed(10);
		System.out.println("Bike Speed: "+b.getSpeed());
	}
}
```

Example of an Extended Class, More Features

Book.java

```
public class Book {
         protected int pages; // Only methods found in this class or those that extend this class may access
this field.

        // The Default Constructor
        public Book() {
                this.pages = 1500;
        }

        public Book(int pages) {
                this.pages = 1500; // Default value

                // If the input value is legal, we'll use the input value.
                if (pages >= 1) {
                        this.pages = pages;
                }
        }

        public void setPages(int pages) {
                if (pages >= 1) {
                        this.pages = pages;
                }
        }

        public int getPages() {
                return pages;
        }
}
```

Dictionary.java

```
public class Dictionary extends Book {
        private int definitions;

        // Default Constructor
        public Dictionary() {
                super(1500);
                this.definitions = 52500;
        }
```

```
    // Constructor
    public Dictionary(int pages, int definitions) {
        super(pages);
        this.definitions = 52500;

        if (definitions > 0) {
            this.definitions = definitions;
        }
    }

    public double computeRatio() {
        return (double) definitions / pages;
    }

    public void setDefinitions(int definitions) {
        if (definitions > 0) {
            this.definitions = definitions;
        }
    }

    public int getDefinitions() {
        return definitions;
    }
}
```

Words.java

```
public class Words {
    public static void main(String[] args) {
        Dictionary basic = new Dictionary()
        Dictionary oxeng = new Dictionary(2000, 80000);

        System.out.println("Basic, pages: "+basic.getPages());
        System.out.println("Basic, words: "+basic.getWords());
        System.out.println("Basic, words per page: "+basic.computeRatio());

        System.out.println("Oxford English, pages: "+oxeng.getPages());
        System.out.println("Oxford English, words: "+oxeng.getWords());
        System.out.println("Oxford English, words per page: "+oxeng.computeRatio());
    }
}
```