July 8, 2010

Review of the Last Class

- A new class can use all of the features of an already made class by extending that class using theextends keyword.
- Similar terms: base class, parent class, super class
- Similar terms: inherited class, child class, derived class
- A child class must hold an "is-a" relationship with a parent class.
- You can have multiple constructors in a class as long as they have different parameters.
- A constructor with no parameter inputs is called a "Default constructor".
- If you do not use a default constructor, Java will make one for you.
- To make a field public to inherited classes but still private to outside classes, mark the field as "protected".
- To the user of a class, all methods appear to be defined in the class being used. This is not always the case. Methods inherited by a parent class appear in the child class.
- When working inside of a child class, to call the constructor of a parent class, the keyword method to use is "super".
- Java is a single inheritance language, meaning that a class can only be derived from at most one parent class. C++ is a multiple inheritance language, meaning that a class can be derived from multiple classes.

Overriding Methods

When creating a subclass, all of the public and protected data members are inherited. Sometimes we aren't completely satisfied with all of the members brought over from the old class. For the fields we wish to change, we simply give them new values. For the members we wish to change, we create new methods using the same name. This is called "overriding the method". When a method is called, first Java will check all of the methods in the current class and attempt to execute the best match. If none are found, it moves up to the parent class and checks those methods. This process continues until it reaches the topmost class. If none are found in that class, the program errors out.

If you instantiate a class based on "Dull" and print that object, this just says "I'm an object."

Let's create a more formal example.

TestQuote.java

```
class Quote {
	public void message() { System.out.println("Whatever you are, be a good one. -Lincoln"); }
}


class NewQuote extends Quote {
	public void message() { System.out.println("A nickel ain't worth a dime anymore. -Yogi Berra"); }
}


class TestQuote {
	public static void main(String[] args) {
		Quote q = new Quote();
		NewQuote p = new NewQuote();
		q.message();
		p.message();
```

```
    }
}
```

You can prevent a method from being overridden by a child class by prefixing the method in the parent class with the keyword final. If Java sees that the final flag is set, it will cause a compile-time error if that method is ever overridden. Use this feature if you are ever certain that the implementation that you create will never need to specialized in a child class.

Class Hierarchies

You are not limited in how many times a class can be derived. In fact, you can derive classes from already derived classes. This chain of classes forms a class hierarchy. Two children of the same parent classes are known as siblings. Each class shares the characteristics of the parent, but they are not related by inheritance because one is not inherited by the other class.
In class hierarchies, common features should be kept high in the chain as possible, so that it the common properties can trickle down to as many child classes as possible. It also helps in maintaining these classes: if the common property needs to be changed, then it can be changed high up in the chain once. Always remember to maintain the "is-a" relationship between child and parent classes.

Properties inherited by a child class become part of the child class. If the child class is extended, thus making it also a parent class, those properties taken from its parent are passed down to its child.

The Object Class

All classes in Java descend from class called "Object". All class that are not extended actually extend "Object" by default (this rule is applied to every class at compile time). Semantically, writing "class X" and "class X extends Object" are identical. Inside of Object is a method called "toString" and if this method is ever called, it throws an Exception. Overriding the method "toString" prevents this from happening.

```
class Dull {
    public string toString() { return "I an object!"; }
}
```

Abstrct Classes

An abstract class represents a generic concept in a class hierarchy. By itself, the class isn't useful, but it has properties which can be passed on to other classes. An abstract class should never be instantiated. Here's the general concept:

- An abstract class is used as a building block for other classes. You aren't suppose to use the abstract class directly.
- There is usually a mixture of typical methods (methods which are called by child classes) and abstract methods.
- Abstract methods must be overridden in the child class. Abstract methods are typically defined using just the return type and signature.
- It is possible to override the non-abstract methods, but this depends on how the abstract class is to be used.

An example of an abstract class.

Work.java

```
abstract public class Work {
    public void result() { System.out.println("Get paid."); }
    abstract public void action();
}
```

Teaching.java

```
public class Teaching extends Work {
    public void action() { System.out.println("I'm teaching."); }
```

```
}
```

AbstractTest.java

```
public class AbstractTest {

	public static void main(String[] args) {

		Teaching t = new Teaching();

		t.action();

		t.result();

	}

}
```

Abstract classes are ideal for creating templates of work.

Designing for Inheritance

These are some tips for designing with inheritance in mind:

- Every child class should hold an “is-a” relationship with the parent class.
- Child classes should always be more specific than their parents.
- Design a class hierarchy around reuse. Put common functionality as high in the in the hierarchy as possible.
- Override methods to tailor the functionality of currently written methods.
- Avoid reusing old variable names from class to class.
- Each class should be responsible for managing its own data.
- It's a good idea to overread “toString”, even when you don't think you will need it.
- Abstract classes are concept classes. They aren't actually created, but they contain functionality that will be shared among one or more child classes.
- Use access modifiers to prevent breaking encapsulation.