Advanced Object Oriented Programming

Abstract Classes

- An **abstract class** models an abstract concept. For example, a musical instrument is an abstract concept. An instrument is something that can be played, but there is no such thing as "instrument" instrument. However, there are piccolo and clarinet or even vocal.
- Abstract classes cannot be instantiated because they do not represent objects. However variables can be declared as the type of an abstract class.
- Abstract classes are declared with the keyword `abstract` in the class declaration. They are intended to be inherited. The `public` members of the abstract class are visible to derived objects.
- An abstract class contains one or more abstract method. An abstract method can only be declared in the abstract class and must be defined in each subclass. The declaration is made in the superclass with the modifier `abstract` but no body for the method is provided in that class.
- e.g. The `Instrument` class is an abstract class with an abstract method. The `makeSound()` method must be implemented in an `Instrument` subclass.

```
abstract class Instrument {
     String musician;

     public Instrument (String name) {
          musician = name;
     }

     public String getMusician() {
          return (musician);
     }

     abstract String makeSound();
}
```

The `Vocal` class is a subclass of Instrument. It provides the body for the `makeSound()` method.

```
public class Vocal extends Instrument {
     public Vocal(String singerName) {
          super(singerName);
     }

     public String makeSound() {
          return ("LaLaLa");
     }
```

```
        public String toString() {
             return (getMusician() + “sings” +
makeSound() + “.”);
        }
  }
```

The `Clarinet` class is another subclass of Instrument. It also defines the body of the `makeSound()` method.

```
  public class Clarinet extends Instrument {
        public Clarinet (String clarinetist) {
             super(clarinetist);
        }

        public String makeSound() {
             return (“squawk”);
        }
  }
```

- Any class that does not define all the abstract methods of its abstract superclass must also be abstract.
- Abstract methods can help eliminate the need of casting
  e.g.

```
  Instrument i = new Clarinet();
  String s = i.makeSound();
```

e.g.

```
  Instrument[] instruments = new Instrument[MAX];

  instruments[0] = new Vocal("Peter");
  instruments[1] = new Clarinet("Zhang");
  instruments[2] = new Clarinet("Yo");

  for (int i = 0; i < MAX; i++) {
     System.out.println(instruments[i].makeSound());
  }
```

The above code would not produce an error because there is no possibility that the object to which `i` refers could be of type `Instrument`. This is because the `Instrument` class is abstract and, therefore, it cannot be instantiated.