Unit Eight Lecture Two:

Topic 1: Interface

An interface is similar to a class, but there are several important differences:

- All methods in an interface are abstract; that is, they have a name, parameters, and a return type, but they don't have an implementation.
- All methods in an interface are automatically public.
- An interface does not have instance variables.
- An interface does not have constructors (you can’t construct an interface object)

If two classes (DataSet1 & DataSet2) are programmed to do nearly identical tasks but with two different classes (BBPlayer & BankAccount), an interface would be appropriate. An interface would allow the two classes (DataSet1 & DataSet2) to be combined into one class and this one class could be used by either of the other classes (BBPlayer or BankAccount).

Topic 2: Polymorphism

The idea of polymorphism means that one method call can call different methods depending on the momentary contents of the object. You have already used something similar to polymorphism when you overloaded methods of a class. Overloading a method means that a single class has several methods with the same name but different parameter types. This occurs when a class has two or more constructors (i.e. a default constructor and a parametric constructor).

There is an important difference between polymorphism and overloading. In overloading, the compiler picks the appropriate method during the translation phase, while in polymorphism the virtual machine, not the compiler, selects the appropriate method during the execution phase. The type of method selection in overloading is called early binding.

Polymorphism means that a driver class could call the DataSet class and send it an object or either type BBPlayer or type BankAccount and the virtual machine would know which method to execute based on the type of object sent to DataSet. This type of method selection is called late (dynamic) binding.

Topic 3: Implementing Polymorphism with an Interface

Step 1: Develop the interface

```
public interface Measurable
{
      double getMeasure();
}
```

This interface is saying that any class that implements Measurable must have a getMeasure() method. This interface requires a single method to be implemented, but in general, an interface can require multiple methods to be implemented.

Step 2: All classes that want to use the interface must implement it in the class header.

```
public class BBPlayer implements Measurable
{
      .
      .
      .

public class BankAccount implements Measurable
{
      .
      .
      .
```

Step 3: All classes that implement an interface must make sure that they have all the methods defined in the interface. This may mean that a method in the class must be renamed.

Change

```
public double getPPG()
{
      return ppg;
}
```

```
To

public double getMeasure()
{
      return ppg;
}


and Change

public double getBalance()
{
      return balance;
}

To

public double getMeasure()
{
      return balance;
}
```

Step 4: Make sure that the one class (DataSet), which is a combination of the classes (DataSet1 & DataSet2) is accepting an object of type Measurable instead of objects of type BBPlayer or BankAccount.

```
public DataSet(Measurable x)   // parametric constructor
{
      double num = x.getMeasure();
      .
      .
      .
```

Step 5: It is optional that the driver class send the DataSet class an object of type Measurable instead of objects of type BBPlayer or BankAccount.

```
.
.
.
input = in.readLine();
BBPlayer aPlayer = new BBPlayer(input);
Measurable m1 = aPlayer;
DataSet z = new DataSet(m1);
.
.
.
BankAccount anAccount = new BankAccount(num, bal);
Measurable m2 = anAccount;
DataSet z = new DataSet(m2);
.
.
.
```

Notice an object of type Measurable is never declared using the word new. New is used only with classes and Measurable is not a class it is an interface.

With these changes a DataSet object can now be used to analyze collections of BBPlayers or BankAccounts and any other class that is willing to implement the Measurable interface.

Unit Eight Assignment One: Combining the BBPlayer & BankAccount classes with an interface