P8A. If “abstract classes” allow you “factor out” common instance variables and methods from subclasses, Java “interfaces” take this concept one step further by allowing you to factor out methods that may be defined by classes that belong to different inheritance hierarchies. Factoring out methods into an interface allows you use polymorphism to process objects that don’t necessarily inherit from the same superclass.

As shown below, the “Connectable” interface creates this “commonality” between “MyPhone”, “MyPod”, and “MyCar” objects although they belong to different superclasses, namely MyDevice and Vehicle. Allowing “MyCar” to inherit from both “MyDevice” and “Vehicle” (a.k.a. “multiple inheritance”) would somewhat solve this problem, but it also introduces other bigger problems, thus, multiple inheritance is not allowed in Java.
In the example below, the interface “Connectable” contains one abstract method “connectToBlueTooth()” that all types of MyDevices (MyPhones and MyPods) AND MyCar will define.

```java
public interface Connectable
{
    // an interface is a collection of one or more
    // abstract methods
    public void connectToBluetooth();
} // end interface connectable
```

Once this is done, we will be able to create an ArrayList of “Connectables” and process them polymorphically.

In this program, we will do the following:

a. Write a “MyCar” class that inherits from the Vehicle superclass and will implement the “Connectable” interface. Specifically, you will write the code for the method “connectToBlueTooth()” that will return the String “Connecting to bluetooth for a MyCar”

b. Edit the “MyDevicesDriver” so that the ArrayList contains “Connectables” instead of “MyDevice” objects.

c. Add a “MyCar” object to the ArrayList of Connectables

d. Polymorphically call the method “connectToBlueTooth()” using for-each loop.

e. Edit the “MyCar” class so that it also implements the Discountable interface (this demonstrating that a class can implement one or more interfaces)
P8B. In this program, you will write an abstract class “TV” and two interfaces “Analog” and “Digital” that you will extend and implement with concrete classes. The diagram below outlines the class hierarchy.

The Analog and Digital interfaces are shown below:

```java
public interface Analog {
    public abstract String rotateRabbitEars();
} // end interface Analog

public interface Digital {
    public abstract String connectHDMI();
} // end interface Digital
```

Part 1:
Write the class definition for the abstract class TV. It should include a private boolean instance variable “tvON” along with a setter and getter method for this variable. It should also include an ABSTRACT method “tvType()” that takes no arguments and returns a String.
Part 2:
Write the BWTV and HDTV classes. The BWTV should extend the abstract class TV and implement the Analog interface. When implementing the Analog interface method “rotateRabbitEars()”, the method should simply return the statement:

"Rabbit ears rotated 45 degrees"

The HDTV class should extend the abstract class TV and implement the Digital interface. When implementing the Digital interface method “connectHDMI()”, the method should simply return the String:

"HDMI cable connected"

Each class should also implement the abstract method “tvType()” in the abstract class TV. A “BWTV” should return the String “I'm a black and white TV” and an HDTV should return the String "I'm a High Def 24 million color TV".

Part 3:
Write a class AnalogDigitalDriver that creates an instance of a BWTV and a HDTV. These objects should then be added to an ArrayList of “TV”s.
   a. Use a for-each loop to print the type of TV and whether or not the TV is on.
   b. Use a regular for loop to turn on each TV.
   c. Use a separate for-each loop to print the type of TV and whether or not it is on.
   d. Print the result of calling the “rotateRabbitEars()” method for a BWTV.
   e. Print the result of calling the “connectHDMI()” method for a HDTV.

Sample output is shown below:

```
I'm a black and white TV...Am I on? false
I'm a High Def 24 million color TV...Am I on? false

I'm a black and white TV...Am I on? true
I'm a High Def 24 million color TV...Am I on? true
Rabbit ears rotated 45 degrees
HDMI cable connected
```

P8C. AP 2012 #3 Horse
By the end of the lesson students should be able to:

a. State the difference between a concrete class and abstract class.

b. Write the Java code to create an abstract class.

c. Write the Java code for concrete and abstract methods in an abstract class.

d. Use an abstract class as an object reference.

e. Write the Java code to use the equals(), getClass(), hashCode(), and toString() methods of the Object class.

f. State the problem caused by multiple inheritance (aka “Deadly Diamond of Death”.)

g. State the difference between an abstract class and a Java interface.

h. Write the Java code to create an interface.

i. Explain why a Java class may more than one Java interface.

j. Write the Java code to invoke a superclass version of a method using the keyword “super”.