Due:
Activity (in-lab): Monday, November 5, 2012 by the end of lab

Goals:
By the end of this project you should be able to do the following:

- Have an understanding of polymorphism via interfaces and inheritance
- Create methods that use polymorphism to deal with multiple types of objects

Directions:
For this assignment, you will need your classes from Activity 9 except for the driver program.

ItemList: method stubs (15%)

- Create a class called ItemList. This class will hold a set of InventoryItem objects (including all instances of classes that inherit from InventoryItem).

- Create the following method stubs for your class:
  - `addItem`: Inventoryitem parameter (itemIn), no return
  - `getTotalPrice`: double parameter (shipRate), double return

ItemList: addItem and toString methods (20%)

- Declare an instance variable in your class of type ArrayList with the generic type InventoryItem called inventory.

```
private _____<InventoryItem> list;
```

- Create a constructor for ItemList that has no parameters. Add the following code to instantiate the ArrayList.

```
_____ = new ArrayList<InventoryItem>();
```

- The addItem method takes an InventoryItem as a parameter and adds it to list (use the add method in ArrayList).

- Override* the toString method in ItemList. The toString method should iterate through inventory and include the string value for each item:

```
public String toString() {
    String output = "All inventory:\r\n\r\n";

    for (_______ item : _______) {
        output += _____ + "\r\n";
    }

    return _____;
}
```

* The toString method is defined in the Object class, which is inherited by every class that is created. Therefore, when you define a toString method in your class, you are overriding the toString method in the Object class.

Main method (25%):

- Add a main method to ItemList. Because the main method will only be used to test the functionality of ItemList, you do not have to create a separate driver program. Instantiate a ItemList object called itemList.

- In the main method, invoke setTaxRate in InventoryItem and set the tax rate to 0.05.

- Instantiate the following 4 items in the main method and add them to itemList:
  - ElectronicsItem: name = Laptop, price = $1234.56, weight = 10 lbs

```
itemList._____(new ElectronicsItem(_____, 1234.56, 10));
```

  - InventoryItem: price = $9.8, name = Motor Oil
  - OnlineBook: price = $12.3, name = "All Things Java"
  - OnlineArticle: price = $3.4, name = Off-Color Acronyms

- Print the toString return value of the ItemList object to standard output:

```
----jGRASP exec: java ItemList

All inventory:

Laptop: $1311.288
Motor Oil: $10.290000000000001
"All Things Java" - Author Not Listed: 12.3
Off-Color Acronyms: $3.4


----jGRASP: operation complete.
```

ItemList: getTotalPrice (25%)

- The getTotalPrice method accepts shipping surcharge for ElectronicItems as a parameter and returns the price of all of the items. You'll have to iterate through each of the items in inventory and add the cost for each item to a running sum (price below). If the item is an ElectronicItem, invoke calculateCost method and add the shipping surcharge that was passed as a parameter to getTotalPrice.

```
double price = 0;
for (_____ item : _____) {

}
```

- **Use the *instanceof* reserved word to determine whether an object is an instance of particular class** (place this in the for loop). If the object is of type ElectronicsItem, you will not be able to access any methods in ElectronicsItem until it is cast as such. For this activity, the cast actually isn't necessary since ElectronicsItem has its own calculateCost method that already adds shipping. However, the following code shows how the instanceof operator can be used. It also adds a small additional shipping charge to offset the rising price of fuel:

```
if (item instanceof ElectronicsItem) {
   price =+ ((ElectronicsItem) item).__________() + shipSurcharge;
}
```

Add an else clause to invoke the calculateCost method using the parameters specified in InventoryItem:

```
else {
   price += item.__________();
}
```

- Return the price.

Main method (15%)

- In the main method of ItemList, add code to print the total price of all items with a shipping rate of 1.5:

```
System.out.println("Total Price: " +
     itemList.getTotalPrice(1.5) + "\r\n");
```

- Your output should appear as follows.

```
 ----jGRASP exec: java ItemList

Inventory:

Laptop: $1311.288
Motor Oil: $10.290000000000001
All Things Java - Author Not Listed: 12.3
Off-Color Acronyms: $3.4

Total Price: 1337.278

 ----jGRASP: operation complete.
```