Goals:

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

➢ Implement interfaces
➢ Overload methods

Directions:

Don't forget to add your Javadoc comments for your classes, constructor, and methods in this activity.

Part 1: Customer.java

- Open a new Java file in jGRASP and create a class called Customer. Create three instance variables:
    - String object for the customer's name
    - String object for the customer's location (town)
    - double to store the customer's balance

- Create a constructor for the Customer class that takes the Customer's name as a parameter. **Set the location variable to an empty string and the balance to zero**. You do not have to include the comments.

```java
public Customer(String nameIn) {
    _____ = nameIn; // sets name to parameter input
    _____ = ""; // sets location to empty string
    _____ = 0; // sets balance to 0
}
```

- Create methods that set the customer's location, change the customer balance by an amount specified by a double parameter, get the location, and get the balance. That is, provide <u>method bodies</u> for the <u>method headers</u> below. Be sure to compile the completed methods before using them in interactions.

```java
public void setLocation(String locationIn) // sets location variable
public void changeBalance(double amount) // add amount to balance
public String getLocation() // returns variable for location
public double getBalance() // returns variable for balance
```

- Try the following example in the interactions pane (you can leave out the comments):

```
▶  Customer cstmr = new Customer("Lane, Jane");
▶  cstmr.changeBalance(30); // add $30 to balance
▶  cstmr.getBalance()
   30.0
▶  cstmr.changeBalance(-5); // take $5 off balance
▶  cstmr.getBalance()
   25.0
▶  cstmr.setLocation("Boston, MA");
▶  cstmr.getLocation()
   Boston, MA
```

- Suppose you want to be able to set the customer location with city and state in a single String or by entering city and state in two separate String parameters. **Overload** the setLocation method so that it takes two String parameters (do not delete the original setLocation method; rather create a second method).

```java
public void setLocation(String city, String state) {
    location = city + ", " + state;
}
```

- Now when the setLocation method is included in your code, the compiler will check to see whether you have one String parameter or two String parameters. It will then **bind the method call** with the appropriate **declaration** of the setLocation method. Try entering the following code into the interactions pane (recompile your program first):

```java
► Customer cstmr = new Customer("Lane, Jane");
► cstmr.setLocation("Boston, MA")
► cstmr.getLocation()
  Boston, MA
► cstmr.setLocation("Omaha", "NE")
► cstmr.getLocation()
  Omaha, NE
```

- Create a toString method that shows the customer's name, their location, and their balance (you do not have to format balance, but do include the dollar sign).

```java
► Customer cstmr = new Customer("Lane, Jane");
► cstmr.setLocation("Boston, MA")
► cstmr.changeBalance(5)
► cstmr
  Lane, Jane
  Boston, MA
  $5.0
```

- Suppose that you wanted to be able to compare Customer objects based on some attribute. You can **implement** the Comparable **interface** in your customer class by indicating this in the class header as shown below.

```java
public class Customer implements Comparable<Customer>
```

By adding this to the Customer header as shown above, your Customer class will now need to implement a compareTo mehod as described below.

- The Comparable interface has a method called compareTo which compares an object of this class to another object to another compatible object indicated by the generic parameter based on some value. Suppose you want to sort Customer objects based on their balance, which is a double, then the compareTo method can be defined as follows:

```java
public int compareTo(Customer obj) {

    if (Math.abs(this.balance - obj.getBalance()) < 0.000001) {
        return 0; // consider them equal and return 0
    }
    else if (this.balance < obj.getBalance()) {
        return -1; // should return a negative number
    }
    else {
        return 1; // should return a positive number
    }
}
```

Part 2: CustomerTest.java - Testing with JUnit the Comparable Interface

- Create a jGRASP project and add the Customer.java file. Now create the test file for the Customer by clicking on the Create/Edit test file button 📄 . Be sure to comment out the following statement: `import static org.junit.Assert.*;`

- Now begin adding test methods for to test each of the Customer methods created in Part 1. After each test method is added to CustomerTest.java, be sure to run the test file.

```java
@Test public void setLocationTest1() {

    Customer cstmr = new Customer("Lane, Jane");
    cstmr.setLocation("Boston, MA");
    Assert.assertEquals("Boston, MA", cstmr.getLocation());
}

@Test public void setLocationTest2() {
    Customer cstmr = new Customer("Lane, Jane");
    cstmr.setLocation("Atlanta", "GA");
    Assert.assertEquals("Atlanta, GA", cstmr.getLocation());
}

@Test public void changeBalanceTest() {
    Customer cstmr = new Customer("Lane, Jane");
    cstmr.changeBalance(100);
    Assert.assertEquals(100, cstmr.getBalance(), 0.000001);
}

@Test public void toStringTest() {
    Customer cstmr = new Customer("Lane, Jane");
    cstmr.setLocation("Auburn, AL");
    cstmr.changeBalance(999);
    Assert.assertEquals("Lane, Jane\nAuburn, AL\n$999.0", cstmr.toString());
}
```

- Now add the following test method to test the compareTo method. Note that three test methods are needed to ensure that the conditions in the compareTo method are covered.

```
@Test public void compareToTest1() {

    Customer cstmr1 = new Customer("Lane, Jane");
    cstmr1.changeBalance(500);

    Customer cstmr2 = new Customer("Lane, Lois");
    cstmr2.changeBalance(500);

    Assert.assertTrue(cstmr1.compareTo(cstmr2) == 0);
}
@Test public void compareToTest2() {

    Customer cstmr1 = new Customer("Lane, Jane");
    cstmr1.changeBalance(100);

    Customer cstmr2 = new Customer("Lane, Lois");
    cstmr2.changeBalance(500);

    Assert.assertTrue(cstmr1.compareTo(cstmr2) < 0);
}
@Test public void compareToTest3() {

    Customer cstmr1 = new Customer("Lane, Jane");
    cstmr1.changeBalance(1000);

    Customer cstmr2 = new Customer("Lane, Lois");
    cstmr2.changeBalance(500);

    Assert.assertTrue(cstmr1.compareTo(cstmr2) > 0);
}
```

- Finally, submit your Customer.java and CustomerTest.java files to Web-CAT.