Assignment #2 (50 points)

Write a class with methods to help you balance your checking account (an object class-main method is not in this class). The CheckingAccount Class should have at least two instance variables: the balance and the total service charges, along with methods to get and set each instance variable. You may add other variables and methods if you like. The program should read the initial balance for the month, followed by a series of transactions. For each transaction entered, the program should display the transaction data, the current balance for the account, and the total service charges. Service charges are $0.10 for a deposit and $0.15 for a check. If a check forces the balance to drop below $500.00 at any point during the month, a service charge of $5.00 is assessed but only for the first time this happens. Anytime the balance drops below $50.00, the program should print a warning message. If a check results in a negative balance, an additional service charge of $10.00 should be assessed. A transaction takes the form of an int number (the transaction code), followed by a double number (the transaction amount). If the int number is a 1, then the double number is the amount of a check. If the int number is 2, then the double number is the amount of a deposit. The last transaction is 0 with no number to follow it. Use the JOptionPane to produce the dialog. A sample Input/Output dialogue is provided via the Sample Run Link below. Use proper style and indentation, efficient programming techniques and meaningful variable and method names according to Java conventions.

Suggested Reading in Textbook: Chapter 3.1 – 3.8, 4.1 – 4.5, 5.1 – 5.5, 6.6

Sample Run

Download Assignment2SampleRun.pdf from the class website.

Code Template

The following code template will help with this assignment:

```
----------  Main.java ----------
public class Main
{
  // global variables
  // define a CheckingAccount object to keep trach of the
  //    account information

  public static void main (String[] args)
```

```
  {
    // defines local variables
    // get initial balance from the user
    // perform in a loop until the transaction code = 0
    // get the transaction code from the user
    //    and process it with appropriate helper method
    // When loop ends show final balance to user.
  }

  public static _____ getTransCode()
  {
  }

  public static _____ getTransAmt()
  {
  }

  public static _____ processCheck(_____)
  {
  }

  public static _____ processDeposit(_____)
  {
  }
}

--------------------CheckingAccount.java ------------------------
public class CheckingAccount
{
  private double balance;
  private double totalServiceCharge;
  public CheckingAccount(double initialBalance)
  {
    balance = _____;
    totalServiceCharge = _____;
  }

  public _____ getBalance()
```

```
  {
    return _________________;
  }

  public void setBalance(double transAmt, int tCode)
  {
    if(tCode == 1)
      balance = ____________________;
    else //if(tCode == 2)
      balance = ______________________;
  }

  public _____________ getServiceCharge()
  {
    return totalServiceCharge;
  }

  public void setServiceCharge(double currentServiceCharge)
  {
    totalServiceCharge = ___________________________;
  }
}
```

Sample Code

```
public class ChemicalElement
{
    private String nameOfElement;
    private String chemicalSymbol;
    private int  atomicNumber;
    private static int numOfElements=0;

    /**
    The ChemicalElement class enables an object that represents a
    chemical element from the periodic table of elements.
    @param name is the name of the element.
    @param symbol is the chemical symbol of the element.
    @param number is the atomic number of the element.
```

```
*/
public ChemicalElement(String name, String symbol, int number)
{
    nameOfElement = name;
    chemicalSymbol = symbol;
    atomicNumber = number;
    numOfElements++;
}

/**
@return The name of the element.
*/
public String getName()
{
    return nameOfElement;
}

/**
@return The chemical symbol of the element.
*/
public String getSymbol()
{
    return chemicalSymbol;
}

/**
@return the atomic number of the element.
*/
public int getNumber()
{
    return atomicNumber;
}

/**
@return the number of elements constructed.
*/
public static int getNumOfElements()
{
```

```
        return numOfElements;
    }
}

//********************************************************************
//  MyElements       Author:  E. Ambrosio
//  Demonstrates the use of the JOptionPane class with the
//  Chemical Elements class
//********************************************************************
import javax.swing.JOptionPane;

public class MyElements
{
  //-----------------------------------------------------------------
  //  Input data for a ChemicalElement object and display the object.
  //  Uses multiple dialog boxes for user interaction.
  //-----------------------------------------------------------------
  public static void main (String[] args)
  {
    String name, symbol, numStr, message;
    int number, again;

    do
    {
      name = JOptionPane.showInputDialog ("Enter the Element name: ");
      symbol = JOptionPane.showInputDialog ("Enter the Element symbol: ");
      numStr = JOptionPane.showInputDialog ("Enter the Element number: ");
      number = Integer.parseInt(numStr);
      ChemicalElement e = new ChemicalElement(name,symbol,number);
      message = "The element name is "+e.getName()+"\n" +
                "Its atomic symbol is "+e.getSymbol()+"\n" +
                "Its atomic number is "+e.getNumber();
      JOptionPane.showMessageDialog (null, message);
      again = JOptionPane.showConfirmDialog (null, "Do Another?");
    }
    while (again == JOptionPane.YES_OPTION);

    message = "The number of elements processed was "+
```

```
            ChemicalElement.getNumOfElements();

      JOptionPane.showMessageDialog (null, message);
   }
}
```