4. **PROGRAMMING ASSIGNMENT 4**: UPDATED 7/12/17

**Read:**
Deitel: Ch 7-8  
Liang:Ch 9, App D

**Programming:** Name your **TWO SEPARATE** files: P4.java and P4StockFund.java

**DUE:** Tuesday, July 18, 2017 @ 6am

You will write an application program to compute the yearly interest for investment stock fund accounts, using classes, constructors, public methods, and private instance variables.

```java
/* Assignment 4  P4.java ... (file header comment) */
import java.util.Scanner; // Scan input from keyboard

/* public class P4 ... (class header comment) */
public class P4
{
    /* Instantiate 2 P4StockFund objects for beginning balances (method header comment) */
    public static void main(String[] args)
    {
        final int YEAR1 = 1; // 1st year to open account
        final double START_BUY = 10; // Start to buy 10 shares
        final double YTD_ROI = .1393; // Year to Date Rate of Return Invest: 13.93%
        boolean isFirstRun = true;
        boolean errRate = false; // Valid rate of return
        int year;
        double rate, ROI;
        double fundShares;
        Scanner scan = new Scanner(System.in); // Read input from keyboard
        String inputStr = new String(); // Repeat program
        P4StockFund investor1 = new P4StockFund(START_BUY); // Overloaded double ctor
        P4StockFund investor2 = new P4StockFund(); // No args ctor. Start 10 shares
        P4StockFund.setRateROI(YTD_ROI); // Current ROI
        do
        { 
            if (!isFirstRun) // Testing ctor, no input 1st run
            {
                do
                { 
                    investor1.buyShares(START_BUY); // Start buy 10 shares
                    while (errFundShares);
                    System.out.print(" Enter new % rate of return (.01): ");
                    rate = scan.nextDouble();
                    if (errRate = !investor2.setRateROI(rate))
                    {
                        System.out.print("Error! Enter valid Range of ");
                        System.out.println(P4StockFund.MIN_ROI + " to " + P4StockFund.MAX_ROI);
                    }
                } while (errRate);
            }
            isFirstRun = false; // Allow input after 1st run
            System.out.println(" \ Yearly earnings for 10 years at " + investor1.getRate());
            System.out.println(" % 7-s\%-{}\%-16s\n", "YEAR", "INVESTOR 1", "INVESTOR 2");
            System.out.println(" % 7-s\%-{}\%-16s\n", "-----", "-----", "-----");
            System.out.println(" % 7-s\%-{}\%-16s\n", "Base", investor1.toString(), investor2);
            for (year = MCM1; year <= YEAR10; year++)
            {
                investor1.calcEarnings();
                investor2.calcEarnings();
                System.out.print(" % 7-s\%-{}\%-16s\n", year, investor1.toString(), investor2);
                System.out.println(" % 7-s\%-{}\%-16s\n", "-----", "-----", "-----");
                System.out.print("Want to calculate more investments \(y/n\)? ");
            }
            System.exit(0); // Exit program
        } while (false);
    }
}
```
PA#4 consists of two public classes in **TWO SEPARATE** FILES (P4StockFund.java and P4.java):

**public class P4StockFund:** Holds value and rate of return data members/methods  
**public class P4:** Application driver class instantiates P4StockFund objects

---

**public class P4**

a) All Programming Assignments will have each method include a short header comment describing its purpose, parameters, and return value. Each class will have a class header comment to summarize class purpose. Each file should have a file header comment with your cs11u login account number, date, and description of the program. Refer to the CSE 11 Lab Guide.

b) The method signatures are shown and described below to produce the given sample output.

c) In `main()`, create 2 local P4StockFund objects.

```java
P4StockFund investor1 = new P4StockFund(START_BUY); // Instantiate overload ctor
P4StockFund investor2 = new P4StockFund(); // No argument constructor
```

d) The user is allowed to repeat these calculations until entering a word beginning with 'N' or 'n'. See program sample output below.

For program turnin, in the **same directory** containing the source code files, type: **"bundleP4"**. Remember to name your TWO files with a "java" extension.

e) Method Descriptions:

- **public static void main(String [] args)**
  - Refer to code given on page 12.

  - Grader input will include at least 2 real numbers. Note: Exceptions will be handled in PA#7.

---

**public class P4StockFund:** Holds data members and methods

---

P4StockFund.java

```java
/**
 * Assignment 4 P4StockFund.java ... (file header comment)
 */
/**
 * class P4StockFund ... (class header comment)
 */
public class P4StockFund {
    private final double INIT_BUY = 5; // Buy shares for no arg ctor
    private final double VFINX = 224.05; // Price/share CA S&P 500 index fund
    final static double MIN = 0; // Minimum for rate and balance
    final static double MIN_ROI = -1; // Minimum -100% rate of return
    final static double MAX_ROI = .5; // Maximum +50% rate of return
    private static double rateROI = 0; // Rate ROI for all accounts
    private double value; // Value of current account

    /**
     * P4StockFund() ... EACH method has a method header comment
     */
    public P4StockFund() { /* No argument constructor */
    }
    public P4StockFund(double val) { /* Ctor initialize with parameter */
    }
    public double getRate() {
        /* ... */
    }
    public static boolean getRateROI(double val) { /* ... */
    }
    public boolean buyShares (double val) { /* ... */
    }
    public void calcEarnings() { /* ... */
    }
    public String toString() { /* ... */
    }
}
```

---

**public class P4StockFund** (continued on next page)
public class P4StockFund (continued)

f) Write 2 constructors: a no argument constructor initializing value data member to initial buy of 5 shares of VFINX at $224.05 a share and a constructor taking a double parameter to initialize the data member value.

```java
public class P4StockFund
{
    // ...
    public P4StockFund() {/* No argument constructor */}
    public P4StockFund(double bal) {/* Ctor initialize with double parameter */ }
}
```

g) Use ONLY one double (instance variable): value and one class variable: rateROI. Both have private access. NO public instance variables in CSE 11.

h) Method Descriptions:
   - `public double getRate()`  
     - return rate of return
   - `public boolean buyShares()`  
     - Buy only positive non-zero number of shares and reduces value by $10 transaction fee  
     - returns true if positive and assigned into instance variable  
     - else false and not assigned into instance variable
   - `public static boolean setRateROI()`  
     - Do NOT assign invalid data (into instance variables  
     - returns true if in range (-1 to .5 inclusive), assigned into instance  
     - variable, else false and not assigned into instance variable
   - `public void calcEarnings()`  
     - Calculates cumulative value multiplied by rate of ROI
   - `public String toString()`  
     - returns a formatted "$%.2f" string, of value

HINTS: Solve this problem in small steps. Here's a suggestion.
1) Type in the above P4StockFund class definition. Write the code for step a). Create stub methods by inserting return statements to match the return types.
2) Complete `main()` as specified in step c) and as given on page 11 in step e).
3) Do step b) as you write each method and class.
4) Write the code for setRateROI(), and toString() in step d). Do input error checking later. Test.
5) Write calcEarnings() and setRateROI() in step h). Test.
6) Complete input error checking. Test. Test thoroughly. DONE!

WARNING: VERIFY you turned in *TWO* SEPARATE files (P4StockFund.java, P4.java)

PA#4 SAMPLE OUTPUT: (SAMPLE INPUT - typed in bold below)

<table>
<thead>
<tr>
<th>Year</th>
<th>INVESTOR 1</th>
<th>INVESTOR 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>$2240.50</td>
<td>$1120.25</td>
</tr>
<tr>
<td>1</td>
<td>$2552.60</td>
<td>$1276.30</td>
</tr>
<tr>
<td>2</td>
<td>$2908.18</td>
<td>$1454.09</td>
</tr>
<tr>
<td>3</td>
<td>$3313.29</td>
<td>$1656.64</td>
</tr>
<tr>
<td>4</td>
<td>$3774.83</td>
<td>$1887.41</td>
</tr>
<tr>
<td>5</td>
<td>$4300.66</td>
<td>$2150.33</td>
</tr>
<tr>
<td>6</td>
<td>$4899.75</td>
<td>$2449.87</td>
</tr>
<tr>
<td>7</td>
<td>$5582.28</td>
<td>$2791.14</td>
</tr>
<tr>
<td>8</td>
<td>$6359.89</td>
<td>$3179.95</td>
</tr>
<tr>
<td>9</td>
<td>$7245.82</td>
<td>$3622.91</td>
</tr>
<tr>
<td>10</td>
<td>$8255.17</td>
<td>$4127.58</td>
</tr>
</tbody>
</table>

Want to calculate more investments (y/n)? a

PA#4 SAMPLE OUTPUT (continued on next page)
PA#4  SAMPLE OUTPUT (continued) (SAMPLE INPUT - typed in **bold** below)

Enter buy new fund shares: **-10**
Error! Enter Positive non-zero Shares.

Enter buy new fund shares: **100**

Enter new % rate of return (.01): **5**
Error! Enter valid Range of -1.0 to 0.5

Enter new % rate of return (.01): **.5**

Yearly earnings for 10 years at 0.500 rate

<table>
<thead>
<tr>
<th>YEAR</th>
<th>INVESTOR 1</th>
<th>INVESTOR 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>$2230.50</td>
<td>$22395.00</td>
</tr>
<tr>
<td>1</td>
<td>$3345.75</td>
<td>$33592.50</td>
</tr>
<tr>
<td>2</td>
<td>$5018.63</td>
<td>$50388.75</td>
</tr>
<tr>
<td>3</td>
<td>$7527.94</td>
<td>$75583.13</td>
</tr>
<tr>
<td>4</td>
<td>$11291.91</td>
<td>$113374.69</td>
</tr>
<tr>
<td>5</td>
<td>$16937.86</td>
<td>$170062.03</td>
</tr>
<tr>
<td>6</td>
<td>$25406.79</td>
<td>$255093.05</td>
</tr>
<tr>
<td>7</td>
<td>$38110.18</td>
<td>$382639.57</td>
</tr>
<tr>
<td>8</td>
<td>$57165.28</td>
<td>$573959.36</td>
</tr>
<tr>
<td>9</td>
<td>$85747.91</td>
<td>$860939.03</td>
</tr>
<tr>
<td>10</td>
<td>$128621.87</td>
<td>$1291408.55</td>
</tr>
</tbody>
</table>

Want to calculate more investments (y/n)? **B**

Enter buy new fund shares: **0**
Error! Enter Positive non-zero Shares.

Enter buy new fund shares: **1**

Enter new % rate of return (.01): **-1**

Yearly earnings for 10 years at -1.000 rate

<table>
<thead>
<tr>
<th>YEAR</th>
<th>INVESTOR 1</th>
<th>INVESTOR 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>$2230.50</td>
<td>$214.05</td>
</tr>
<tr>
<td>1</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>2</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>3</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>4</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>5</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>6</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>7</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>8</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>9</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>10</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
</tbody>
</table>

Want to calculate more investments (y/n)? **noWay**

Note: This is a very simplistic accumulation of earnings that is highly complex in reality. i.e. market volatility!

**Verify you TURNED-IN your work by re-typing “bundleP4”**