MIST.3050 Business Applications Development

Homework: Investment Simulator (Arrays)

You are encouraged to form a team of 2 or 3 to work on this homework. Each team submits only one copy of the homework. Submit your homework on Blackboard before the deadline. No late homework is accepted.

Introduction

You probably have heard of the saying, "A penny saved is a penny earned". Well, if you invest your saved money wisely, a penny saved can be more than a penny earned. How much can it be? In this homework, you will figure this out through simulation.

Suppose you are a disciplined investor. In the beginning of each month, you invest the same amount of your saved money in selected funds. After 10 ten years, how much will be your investment? This depends on two factors: the amount of your monthly investment, the monthly return of your investment portfolio. For the monthly investment amount, you may consider scenarios of different amounts, e.g., $50, $100, $500, and $1,000. For monthly return, well, nobody would know because the market fluctuates all the time. But we can use simulated values for the returns.

According to the article at `https://www.ifa.com/articles/with_stock_returns_normally_distributed/`, historic monthly returns roughly have a normal distribution. For the simulation, we assume that monthly return is drawn probabilistically from two normal distributions:

- 95% probability from $N(1\%, 2.5\%)$
- 5% probability from $N(-4.5\%, 2.5\%)$

You will be provided with a partially finished program. After adding your code to complete the program, it will output the monthly rate of return and account balance for each saving scenario. Below is a sample output.

```
 Month  Return          50         100         500        1000
  1    1.1696%       50.58      101.17      505.85    1,011.70
  2    0.2533%      100.84      201.68    1,008.40    2,016.79
  3    2.0974%      154.00      308.01    1,540.03    3,080.07
  ...
119    1.8881%    9,481.88   18,963.76   94,818.81  189,637.61
120    1.5656%    9,681.11   19,362.23   96,811.13  193,622.25
Average Monthly Return: 0.6426%
```

Partially Completed Program

The program includes two helper methods to generate simulated monthly returns. You only need to add code to the `main` method as instructed by the embedded comments.

```java
import java.util.Random;
public class InvestSimulation {
  private static Random random = new Random();
  private static double nMean=-.045, nStd=.025, pMean=.01, pStd=.025;

  public static void main(String... args ) {
    int N=120; // for ten years (120 months)
    double[] r = new double[N];
    generateReturns(r);

    //add your code here:
    //use a 1-D array to hold monthly investment amounts (four scenarios)
    //create a 2-D array for storing monthly balance (120 months, 4 scenarios)
    //define a variable to store running total of monthly return
    //use a nested loop to populate the 2-D array and output data.

  }

  public static double getGaussian(double mean, double stddev) {
    return mean + stddev*random.nextGaussian();
  }

  public static void generateReturns(double[] r) {
    for (int i=0; i<r.length; i++) {
      if (Math.random()<.05) { // negative return
        r[i] = getGaussian(nMean, nStd);
      } else {
        r[i] = getGaussian(pMean, pStd);
      }
    }
  }

}
```

Implementation Requirement

This program must be implemented by following the instructions included as comments in the code provided above.

Here is a bit more information about how to calculate monthly balance. The balance at the end of a given month depends on the balance at the end of the previous month, the investment amount each month, and the rate of return of the given month. Let $a_{i,j}$ be the account balance at the end of month i of the j–th investment scenario (monthly investment amount), r_i be the rate of return for month i, and m_j be the amount of new investment in the beginning of each month (e.g., 50, 100, etc) of the j–th investment scenario. Then we have

$$a_{i,j} = (a_{i-1,j} + m_j)(1 + r_i)$$

The values of the three variables are from three arrays. Pay special attention to the first month, for which the previous month's balance must be 0, i.e., the $a_{i-1,j}$ must be 0. If you let $i = 0$ be the the first month, then the first month must be handled differently from the rest of the months; if you let $i = 1$ be the first month, then you can calculate all months the same way (but the array must have one extra row). Example code for choosing $i = 0$ to te the first month is shown below:

```
if (i==0) {
  data[i][j] = monthlyInvest[j]*(1+r[i]);
} else {
  data[i][j] = (data[i-1][j] + monthlyInvest[j] )*(1+r[i]);
}
```

Submission

Create your write-up using a word processor (e.g., Word). Make sure you include the following parts:

- Title page. The title page should include the assignment information, your name (or team member names), and a sentence summarizing what the program is about.
- Table of contents. List of all sections included in the submission. You do not need to have page number for each section.
- The required sections of this assignment include:
  - Overview. A brief description of the program: its function, input (the input is tuition information hard-coded in the program), and output.
  - Design and Implementation. A description of the design choices and structure of the program: class, methods, variables, and other notable functions used (e.g., `printf`). **Include a flowchart for the `generateReturns` method. Also include a UML class diagram of the class.** You can create the class diagram using a table of the word processor to separate the three sections.

- Testing. Include screenshots to show the function (e.g., output) of the program.
- Discussion. Include the following discussions as subsections:
  - Difficulties encountered and how you solved them.
  - Sources of help you used.
  - What did you learn in terms of programming?
  - What did you learn in terms of personal finance? Try to related to the annual expense exercise.
  - Any other lessons learned from the assignment.
- Source code. Include source code as text, not images. Include comments in the source code to document your understanding of the code.

Note: It is important for you to include all required parts. A working program counts for ~70% of the points; documentation counts for the rest of the points.