Project 2: Interest (Calculating the Future Value of an Investment)

For the second project, you will complete a partial Java program that will be given to you. There are two main goals for this project:

- Learn how to extract individual data items from a line of input and convert them into equivalent numeric representations.
- Perform simple arithmetic computations in Java.

You will complete the writing of a program to calculate the *future value of an investment*, using the formula:

$$S = P\left(1+\frac{r}{n}\right)^{nt}$$

where:

`S`	is the value of the investment after `t` years
`P`	is the initial value of the investment
`r`	is the annual interest rate the investment earns
`n`	is the number of periods per year
`t`	is the number of years the investment is held

This assignment will help further prepare you for the third project when you have to author your first complete Java program.

Input file description and sample:

The program reads it's input from a file named `invest.txt` — (use of another input file name would result in massive deductions by the automatic grader). Each line of the input file will contain four values, separated by whitespace:

- the amount initially invested (positive real number between 1.00 and 100,000.00)
- the annual interest rate earned (positive real number between 0.001 and 0.20)
- number of periods per year (integer between 1 and 365)
- number of years investment is held (integer between 1 and 30)

It may be assumed that all input values will be logically correct (no negatives, for instance). The principal data values will be in columns 1-10, the rate data will be in columns 13-17, the period data will be in columns 21-23 and the years data will be in columns 31-32. A sample input file follows:

```
 Principal   Rate Periods   Years
   1000.00   0.05    12       10
   1000.00   0.05    12       20
   1000.00   0.05    12       30
   1000.00   0.05   365       10
   1000.00   0.05   365       20
   1000.00   0.05   365       30
```

Note that the given program code does not make **any** assumptions about the number of lines of data in the input file. The program has been written so that it will detect when it's out of input and terminate correctly.

What to Calculate:

The program reads in each line of input data, as described above, one line/record at a time. You must provide the code, (where indicated) to parse, (i.e separate), out the individual data values from a line into the provided correspondingly named reference variables. Where you need to do this is indicated in the code below. Use the

knowledge you gain from your work with strings in the lab to complete this section of the program. You must then provide code to calculate the corresponding future value of the initial investment using the above formula. In order to achieve the best possible accuracy, the program code uses floating point `double` storage for all real, (i.e., decimal), numbers needed in this program.

In order to raise the parenthesized quantity to a power, since Java does not provide a power operator, use the standard class `Math` library static function `Math.pow( x, y )`. This function may be used to calculate x^y, where x and y are of type `double` and the return value is of type `double`.

Output description and sample:

The program writes all output data to a file named `balance.txt` — (use of any other output file name would again result in a massive deduction of points by the auto-grader). The sample output file shown below corresponds to the sample input data given above:

```
Programmer: Dwight Barnette:
CS 1054 Project 2 Spring 2003

  Principal     Rate      Periods     Years            Balance
_____________________________________________________________
   $1000.00     0.05         12         10           $1647.01
   $1000.00     0.05         12         20           $2712.64
   $1000.00     0.05         12         30           $4467.74
   $1000.00     0.05        365         10           $1648.66
   $1000.00     0.05        365         20           $2718.10
   $1000.00     0.05        365         30           $4481.23
_____________________________________________________________
```

The program is not required to use this exact horizontal spacing, but the output must satisfy the following requirements:

- The first line of the output file must contain **your** name.
- The second line of the output file must contain the given project details.
- The third line of the output file must be blank.
- You **must** use the specified column labels, in the fourth line of the output file.
- The fifth line must be a line of underscore characters.
- All dollar amounts **must** be formatted to show two places after the decimal.
- The annual interest rates **must** be formatted to show **two** places after the decimal.
- The number of periods and number of years **must** be formatted as integer values.
- Allow sufficient space for each value your program prints; the future values **could** be on the order of 10 million.
- You **must** arrange your output in neatly aligned columns.
- You **must** use the same ordering of the columns as shown here.
- After the last line of interest data has been output, the final line output must be a line of underscore characters.
- **Do not** insert any additional lines of output; **do** have a newline at the end of each line.

Programming Standards:

You'll be expected to observe good programming and documentation standards. All the discussions in class about formatting, structure, and commenting your code will be enforced. A copy of Elements of Programming Style is available on the course web site. The given code satisfies the documentation standards below. You will be expected to follow these standards and others in all of your later programs. Some, but not necessarily all, specifics include:

- You must include header comments specifying your name, the compiler and operating system used and the date your source code and documentation were completed.
- The header comment must also include a brief description of the purpose of the program (sort of a user guide) — this should be in your own words, not copied from this specification.
- You must include a comment explaining the purpose of every variable you use in your program.
- You must use meaningful, suggestive (of function or purpose!) variable names.
- Precede every major block of your code with a comment explaining its purpose. You don't have to describe how it works unless you do something so sneaky it deserves special recognition.

Testing:

At minimum, you should be certain that your program produces the output given above when you use the given input file and the sample test files provide don the course web site. However, verifying that your program produces correct results on a few test cases does not constitute a satisfactory testing regimen. You should make up and try additional input files as well; of course, you'll have to determine what the correct output would be.

Evaluation:

Your program will be submitted to the Curator system for evaluation of correctness (details are given below). You will have five submissions to allow you an opportunity to correct mistakes that your own testing has not found. **You will not be given more submissions for any reason!** Your submitted program will be assigned a grade based on the correctness of your program and possibly evaluated by a TA. The grade you receive in the email from the Curator is your correctness grade. The grade for the project for future assignments will be computed by deducting any points for style and design from the correctness Curator grade. Therefore, the correctness grade is the *maximum* that your program will receive. **The TA will always grade the *first* submission that has the highest points.** No exceptions will be made, so be sure that you have done everything you need to before submitting. Note: Check your grade a few minutes after submitting. **If you get "N/A" as a grade, do not submit again until you receive a numeric grade.**

Submitting your Program

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*. You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted. The *Student Guide* can be found at: `http://www.cs.vt.edu/curator/`

Program Compilation

Your program must compile and run under JDK 1.2.

Pledge

Every program submission for this class must include an Honor Code pledge. Specifically, you **must always** include the following pledge statement in the header comment for your program:

```
// On my honor:
//
// - I have not discussed the Java language code in my program with
// anyone other than my instructor or the teaching assistants
// assigned to this course.
//
// - I have not used Java language code obtained from another student,
// or any other unauthorized source, either modified or unmodified.
//
// - If any Java language code or documentation used in my program
// was obtained from another source, such as a text book or course
// notes, that has been clearly noted with a proper citation in
// the comments of my program.
//
// - I have not designed this program in such a way as to defeat or
// interfere with the normal operation of the Curator System.
//
// On my honor, I have neither given nor received unauthorized
// aid on this assignment.
//
// <Student Name>
// <Student PID>
```

Failure to include this pledge in a submission is a violation of the Honor Code.

Source Code

The following source code is also available as a downloadable file from the course web site assignment page. Comments indicate the parts of the program that it is your responsibility to supply.

Program starts here

```
//  Program2.java:    Calculating Future Value of an Investment
//
//  You should substitute the correct information in the following
//  lines:
//
//  Programmer:   Dwight Barnette
//  Modified:     <STUDENT NAME>
//  Created:      Jan. 20, 2003
//  Mod date:     <Modification Date>
//  Compiler:     Metrowerks CodeWarrior ver. 4.2.5
//  Platform:     Pentium 4 / 1.5 GHz / 512MB / Windows 2000 5.00.2195 SP 3
//
//  Purpose of the program:
//
//    If P dollars are invested at an annual rate of r, compounded n
//    times per year for t years, the final (or future) value of the
//    investment is given by the formula:
//
//       S = P(1 + r/n)^(nt)
//
//    This program applies this formula to calculate the future value
//    of investments specified in the input file "invest.txt" and prints
//    a report summarizing each investment to the output file
```

```
//    "balance.txt".
//    The report includes the information above and the final value of
//    the investment.
//

// On my honor:
//
// - I have not discussed the Java language code in my program with
// anyone other than my instructor or the teaching assistants
// assigned to this course.
//
// - I have not used Java language code obtained from another student,
// or any other unauthorized source, either modified or unmodified.
//
// - If any Java language code or documentation used in my program
// was obtained from another source, such as a text book or course
// notes, that has been clearly noted with a proper citation in
// the comments of my program.
//
// - I have not designed this program in such a way as to defeat or
// interfere with the normal operation of the Curator System.
//
// On my honor, I have neither given nor received unauthorized
// aid on this assignment.
//
// <Your Name>
// <Your PID>


import java.io.*;
import java.text.*;


public class Program2 {

    // Method:        main
    // Programmer     Dwight Barnette
    // Email:         barnette@vt.edu
    // Modified:      <STUDENT NAME>
    // Email:         <STUDENT EMAIL>
    // Arguments:     String array; ignored
    // Returns:       void
    // Throws:        FileNotFound & IO Exceptions
    // Last Modified: Jan. 21, 2003
    //
    // Purpose: This method declares variables, reads and decodes
    //          input, and generates and writes appropriate output.

 public static void main(String args[]) throws Exception {

   //  Variable declarations:

   int    NumPeriods,         // number of periods per year
          NumYears;           // number of years investment is held

   double InitBalance,        // starting balance of investment
          AnnualRate,         // annual interest rate earned
```

```
        FinalBalance;       // final balance of investment
    int NumLines = 0;       // number of lines of data processed

    // String constants fors holding the input and output filenames
    final String MYINFILENAME  = "invest.txt";
    final String MYOUTFILENAME = "balance.txt";

    //String constants for output data labelling
    final String PROGRAMMER    = "Programmer: <STUDENT NAME>:";
    final String PROGRAM       = "CS 1054 Project 2 Spring 2003";
    final String OUTPUTHEADER  = "  Principal     Rate      Periods" +
                                 "     Years            Balance";

    //String constants for output data formatting
    final String SPACES        = "                                        " +
                                 "          ";
    final String LINE          = "_____________________________________" +
                                 "_____________________________________";

    // String objects representing the label line & date strings from the
    // input stream.
    String LabelLine,               // input columns labels
           Investment,              // investment data line
           Principal,               // principal of investment
           Rate,                    // annual interest rate
           Periods,                 // number of annual compounding periods
           Years;                   // number of years of investment

    // objects to read input and write output
    BufferedReader myInput;
    PrintStream    myOutput;

    //Decimal Format for precision output control
    DecimalFormat df;

    //*************************************************************************
    //*               SET UP THE BUFFERED INPUT STREAM READER                 *
    //*************************************************************************
    FileInputStream myFileIS = new FileInputStream(new File(MYINFILENAME) );
    myInput = new BufferedReader(new InputStreamReader(myFileIS) );
    //*************************************************************************

    //*************************************************************************
    //*              SET UP THE PRINT STREAM / OUTPUT STREAM                  *
    //*************************************************************************
    FileOutputStream myFileOS = new FileOutputStream(
                                new File(MYOUTFILENAME) );
    myOutput = new PrintStream(myFileOS);
    //*************************************************************************

//  Print the header info to output file:

    // write the headers out
    myOutput.println(PROGRAMMER);
    myOutput.println(PROGRAM);
    myOutput.println();
    myOutput.println(OUTPUTHEADER);
```

```
      myOutput.println(LINE);

      // ignore the label line
      LabelLine = myInput.readLine();


  //  Read first line of data from input file:

      //assume data is valid
      Investment  = myInput.readLine();   //Principal  Rate  Periods  Years


  //  Process investment records (lines of data) until none remain:

      while ((Investment != null)) {

            ////////////////////////////////////////////////////////////
            // STUDENTS COMPLETE THIS SECTION                         //
            ////////////////////////////////////////////////////////////
            //Extract Principal, Rate, Periods & Years strings from
            //Investment string
            Principal =                      ;    //pull out Principal
            Rate      =                      ;    //pull out Rate
            Periods   =                      ;    //pull out Periods
            Years     =                      ;    //pull out Years

            //erase leading/trailing whitespace
            Principal =                      ;
            Rate      =                      ;
            Periods   =                      ;
            Years     =                      ;
            ////////////////////////////////////////////////////////////

            //convert strings to doubles & ints as necessary
            InitBalance = Double.valueOf(Principal).doubleValue();
            AnnualRate  = Double.valueOf(Rate).doubleValue();
            NumPeriods  = Integer.parseInt(Periods);
            NumYears    = Integer.parseInt(Years);


            NumLines = NumLines + 1;                  // Count lines/records

            ////////////////////////////////////////////////////////////
            // STUDENTS COMPLETE THIS SECTION                         //
            ////////////////////////////////////////////////////////////
            // -------------------- Begin Processing Section
            // Perform intermediate computations




            //  Compute the future value using compound interest formula:

            FinalBalance =                                  ;
```

```
            // ------------------- End of Processing Section
            ////////////////////////////////////////////////////////////

            // ------------------- Print summary information for investment:

            myOutput.print(SPACES.substring(0, 10 - Principal.length()));
            df = new DecimalFormat("$######0.00");
            myOutput.print(df.format(InitBalance));

            myOutput.print(SPACES.substring(0, 9 - Rate.length()));
            df = new DecimalFormat("#0.00");
            myOutput.print(df.format(AnnualRate));

            myOutput.print(SPACES.substring(0, 11 - Periods.length()));
            myOutput.print(Periods);

            myOutput.print(SPACES.substring(0, 11 - Years.length()));
            myOutput.print(Years);

            myOutput.print(SPACES.substring(0, 16 - (String.valueOf(
                                        (long) FinalBalance)).length()));
            df = new DecimalFormat("$############0.00");
            myOutput.println(df.format(FinalBalance));


            // ------------------- End of Output Section

            //  Read the next line of input:

            Investment  = myInput.readLine(); //Principal Rate Periods  Years

        }// End of while loop.

        myOutput.println(LINE); //output table footer

        // close the streams

        myFileIS.close();
        myFileOS.close();

      }//main

    }//Program2
```