Intermediate Java II
MISM/MSIT 95-713
Homework 1

There are 4 exercises in this homework, each 25 points.

For this homework, we will be grading your programs based on the correct execution only. We will not deduct points based on programming style, output format, comments and coding convention. You may assume the user will always enter valid input.

In this homework, you practice basic operators and control structures of the Java programming language. The objectives of this homework include:

- Getting familiar with the integrated development environment in Eclipse
- Practicing Java operators
- Practicing control structures such as loops and if-else statements
- Reading input from user and printing out messages back to user
- Practicing random number generation

Name your files/classes as Homework1_X where X is the exercise number below. Put all your java and the compiled class files into a zip file named Homework1.zip and submit it to the blackboard before the due date.

1. Write a Java program to calculate the letter grades of students and report them. Your program should first ask the user to enter the number of students in the class and read it. For each student, generate a random integer number ranging from 50 to 100 to be the average score of the student. Calculate the letter grade of the student as follows[1]:
   For average scores from 90 to 100, the letter grade should be “A”.
   For average scores from 80 to 89, the letter grade should be “B”.
   For average scores from 70 to 79, the letter grade should be “C”.
   For average scores from 60 to 69, the letter grade should be “D”.
   For average scores from 0 to 59, the letter grade should be “F”.

   Report the average scores by printing the generated scores in increasing order for each letter grade on a separate line of output. You program should print all the scores that are being assigned an “A” in increasing order in one line of output, then all scores of “B” in increasing order in next line of output, and similarly for the letter grades “C”, “D” and “F”.

2. Write a program to calculate the number of years required for a person to retire. Your program should generate an integer number in the range of 150,000.00 and 200,000.00 to be the retirement goal of the person. You should print out the retirement goal to the console.

[1] This calculation is just for homework exercise purpose and is not related to assigning your letter grade for this course.

The initial balance for the person should be 0. For each year, the person may contribute an amount less than or equal to $10,000.00. Your program should generate a random number for the contribution value and add it to the balance. Your program should then generate a positive double number less than 5.0 representing the percentage of the rate of return for that year, and another positive double number less than 4.0 representing the percentage of inflation. The actual gain (or the loss) should be calculated for this year as the multiplication of the balance and the percentage value of (rate of return – inflation), and should be added onto the balance. The contribution, the actual gain (or the loss), and the new balance should be printed for each year. Your program should continue until the balance meets or exceeds the retirement goal of the person, after which, it should print the number of years needed to retire.

3. Write a Java program to calculate the monthly payment for a loan as follows. Read the principal, interest rate and number of years from the user of your program. Using the formula below, calculate the monthly payment.

   $$\text{monthlyPayment} = (\text{interestRate} * (\text{principal} / \text{paymentPerYear})) / (1 - ((\text{interestRate} / \text{paymentPerYear}) + 1)^{-\text{paymentPerYear}* \text{ numberOfYears}})$$

   Define paymentPerYear as a constant which is equal to 12. Your program should present the following information to the user:

   The monthly payment
   The total payment after each year
   The total interest paid during the course of the loan.

4. Write a Java program to experiment String and StringBuffer classes. See their documentation at http://download.oracle.com/javase/7/docs/api/. Your program should expect exactly two command-line arguments. If the number of command-line arguments is not two, the main function should return immediately. Otherwise, it should print out the command-line arguments and continue as follows:
   - Create two String objects, say s1 and s2, initializing them with the first and the second command-line arguments. Experiment and print out the results of the following: s1.length(), s1.charAt(i) for all i for String s1, s1.equals(s2), s1.equalsIgnoreCase(s2), s1.compareTo(s2), s1.regionMatches(int toffset, s2, int offset, int len) for some offset and len, s1.regionMatches(boolean ignoreCase, int toffset, s2, int offset, int len) for some offset and len, s1.indexOf(c, i) for some c and i, s1.concat(s2), s1.replace(c1, c2), s1.upperCase(), s1.lowerCase().
   - Create two StringBuffer objects, say sbuf1 and sbuf2, initializing them with the first and the second command-line arguments. Experiment and print out the results of the following: sbuf1.length(), sbuf1.delete(int start, int end), sbuf1.deleteCharAt(int index), sbuf1.reverse() methods. Invoke sbuf1.replace() with proper arguments. Call the sbuf1.append().append() methods in a chain of method calls by passing primitive data types and sbuf2 as the parameters. Additionally, introduce a new class named MyClass in the same file. MyClass should define nothing but the toString method that returns

a string of “This is my object”. Invoke sbuf1.append() by passing an object of MyClass as the parameter. Invoke sbuf1.insert() method just like you invoked the append() method.