Lab09
More on Java Branching Statements

Submission Details

In this lab, you are required to write **TWO** Java programs to solve the given problems shown in the section “Lab Exercises”. To make the program implementation easier, you are suggested to write your program using **eclipse** instead of doing it directly using paper & pencil. After you have completed the implementation using eclipse, submit your program files (i.e. CheckHoroscope.java and AdvancedQuadraticCalculator.java) to the assignment page of Canvas. The deadline would be **ONE** week after your lab section is conducted. You are reminded to double-check your solution to verify everything in correct before submission.

Note: You are required to put your name, student ID and lab section at the beginning of your source file. Also, please DON'T ZIP your files for submission. Instead, you should upload all .java files to the Canvas.

Note: Only answer of selected question will be graded.

Objective

The objective of this lab is to give you further practice to reinforce your programming concepts (e.g. basic Java input / output, variable declarations, use of operators and ***branching statements***) taught in the last few lectures.

Software Required

The following software is required for this lab.
1. Java Development Kit Version 1.8
2. An Integrated Development Environment, e.g. eclipse

Introduction

In this lab, you will again practice:
1. Forming a program skeleton by declaring a class with a main method / operation
2. Declaring variables to store values in Java programs
3. Obtaining inputs from keyboard using methods / operations in the Scanner class
4. Solving problems by performing arithmetic operations (+, -, *, /, %) and executing code using ***branching statements***
5. Displaying the solution of problems on screen using methods / operations in the System class

Background

1. **Form a program skeleton by defining a class with a main method / operation**

   - **Syntax:**

     ```
     // The filename of the Java source file should be exactly the same
     // as the <class name>
     public class <class name>
     {
         // Define a main method / operation, which is the starting point of the program
         public static void main(String[] args)
         {
             // Program statements here
         }
     }
     ```

2. **Declare variables for storing values**

   - **Syntax:**

     ```
     // Declare a variable with name <name of variable> in type <type>
     <type> <name of variable>;
     // Declare a variable with name <name of variable> in type <type> with
     // initial value <initial value>
     <type> <name of variable> = <initial value>;
     // Declare a number of variables with name <variable name1>,
     // <variable name2>, … all in type <type>
     <type> <variable name1>, <variable name2>, …;
     // Declare a number of variables with name <variable name1>,
     // <variable name2>, … all in type <type> with initial value <initial value1>,
     // <initial value2>, …, respectively
     <type> <variable name1> = <initial value1>, <variable name2> = <initial value2>, …;
     ```

     where <type> could be in one of the following types:
     byte, short, int, long, float, double, char, boolean

3. **Obtain inputs from keyboard and store them in variables:**

   Syntax:

   ```
   // Get the class definition of the Scanner class from java.util package
   import java.util.Scanner;
   // Create a Scanner object with name <object name>
   // System.in is the "standard" input stream, which is keyboard
   Scanner <object name> = new Scanner(System.in);
   // Use the Scanner class's operations (methods) to read input
   <variable name> = <object name>.<method name>();
   ```

4. **Perform calculation using arithmetic operations:**

- **Syntax:**

```
// Add the two values stored in variable 1 and variable 2
<name of variable for the result> = <name of variable 1> + <name of variable 2>;
// Subtract the value stored in variable 2 from variable 1
<name of variable for the result> = <name of variable 1> - <name of variable 2>;
// Multiply the value stored in variable 1 by the value stored in variable 2
<name of variable for the result> = <name of variable 1> * <name of variable 2>;
// Divide the value stored in variable 1 by the value stored in variable 2
<name of variable for the result> = <name of variable 1> / <name of variable 2>;
// Find the remainder of value stored in variable 1 divided by the value stored in
// variable 2
<name of variable for the result> = <name of variable 1> % <name of variable 2>;
```

5. **Do checking using branching statements:**

- **Syntax:**

```
// If <boolean expression> is evaluated to true, <statement> is executed.
// Otherwise, just skip it.
if(<boolean expression>)
    <statement>;
// If <boolean expression> is evaluated to true, <statement 1>, <statement 2>,
// <statement 3>, …,    are executed. Otherwise, just skip them.
if(<boolean expression>) {
    <statement 1>;
    <statement 2>;
    <statement 3>;
    // …
}


// If <boolean expression> is evaluated to true, <statement A> is executed.
// Otherwise, <statement B> is executed.
if(<boolean expression>)
    <statement A>;
else
    <statement B>;
// If <boolean expression> is evaluated to true, <statement A1>, <statement A2>,
// <statement A3>, …,    are executed. Otherwise, <statement B1>,
// <statement B2>, <statement B3>, …, are executed.
if(<boolean expression>) {
    <statement A1>;
    <statement A2>;
    <statement A3>;
    // …
}
```

```
else {
   <statement B1>;
   <statement B2>;
   <statement B3>;
   // …
}
```

- **Syntax:**

```
// switch <expression> matches a case, all the statement(s) of that case is/are executed
// until a break statement is found. If none of the case matches, the statement(s) at
// the default case is/are executed.
switch(<expression>) {
   case <constant a>:
       <statement a1>;
       // …
       break;
   // …
   default:
       <statement c1>;
       // …
}
```

6. **Display information on screen:**

- **Syntax:**

```
// Print data on screen and the cursor will stay at the end of the string printed
System.out.print(<data> [+ <data>]);
// Print the data on screen and move the cursor to the beginning of the next line
System.out.println(<data> [+ <data>]);
// Print nothing, but move the cursor to the beginning of the next line
System.out.println(“”);
```

Lab Exercises

In this lab, you will be asked to solve **TWO** problems using Java. Please refer to the problem specifications below for the requirements.

Question 1:

Write a Java program “CheckHoroscope.java” that reads in user input on birth year and prints out the corresponding Chinese horoscope sign. The program allows a user to enquire about horoscope signs. There are 12 signs in the Chinese horoscope. The calculation of a horoscope sign is based on birth year. The 12 Chinese horoscope signs and some of their corresponding years are given below:

Horoscope Sign	**Birth Year**
Rat	…1972, 1984, 1996…
Cow	…1973, 1985, 1997…
Tiger	…1974, 1986, 1998…
Rabbit	…1975, 1987, 1999…
Dragon	…1976, 1988, 2000…
Snake	…1977, 1989, 2001…
Horse	…1978, 1990, 2002…
Goat	…1979, 1991, 2003…
Monkey	…1980, 1992, 2004…
Rooster	…1981, 1993, 2005…
Dog	…1982, 1994, 2006…
Pig	…1983, 1995, 2007…

Expected outputs of your program:

Input / output session 1:
```
Enter your birth year: 1973
Your horoscope sign: Cow
```

Input / output session 2:
```
Enter your birth year: 2006
Your horoscope sign: Dog
```

Input / output session 3:
```
Enter your birth year: 2010
Your horoscope sign: Tiger
```

<u>Question 2:</u>

Write a Java program "AdvancedQuadraticCalculator.java" that reads the coefficients of a quadratic equation (i.e. a, b & c) from the user and then computes and prints the solution(s) of x.

Recall a quadratic equation is an equation defined as a polynomial of degree two and it could be represented by:

$$ax^2 + bx + c = 0$$

where a$\neq$ 0, b, c are called coefficient of the equation and x is the unknown value we would like to find.
To solve the quadratic equation, the first step is to use the following equation to compute the discriminant:

$$\Delta = b^2 - 4ac$$

Then, the (real) solution of the quadratic equation is given by:

- If $\Delta > 0$, then $x = (-b + \sqrt{\Delta})/(2a)$ or $x = (-b - \sqrt{\Delta})/(2a)$.
- If $\Delta = 0$, then $x = (-b)/(2a)$.
- If $\Delta < 0$, then no real solution exist.

Note that the input number can be assumed to be correct (i.e. user won't input some invalid things, like "a", "abc", "-12-0.-2", etc.). Therefore, no input validation is needed. However, you must check whether the input a is equal to zero or not. If it is zero, then you must print

Invalid input

and exit the program immediately. Otherwise, the program should print out the equation to be solved and the corresponding (real) solutions according to the above given method.

The expected outputs of your program are as follows:

<u>Distinct Solutions:</u>

```
Enter coefficient a: 1
Enter coefficient b: 3.5
Enter coefficient c: 2
Solving 1.0x^2 + 3.5x + 2.0 = 0
Two distinct solutions:
x = -0.7192235935955849 &
x = -2.7807764064044154
```

<u>Repeated Solution</u>

```
Enter coefficient a: 1
Enter coefficient b: 2
Enter coefficient c: 1
Solving 1.0x^2 + 2.0x + 1.0 = 0
Repeated solution: x = -1.0
```

No solution

```
Enter coefficient a: 1
Enter coefficient b: 0
Enter coefficient c: 1
Solving 1.0x^2 + 0.0x + 1.0 = 0
There is no real solution
```

Invalid input

```
Enter coefficient a: 0
Invalid input
```

Marking Scheme

The marking of this exercise will be based on the following criteria.

Graded items	Weighting
1. Correctness of program (i.e. whether your code is implemented in a way according to the requirements as specified.)	50%
2. Presentation of the Java codes (i.e. whether the program is properly indented, how close you follow the common conventions as mentioned in class, etc.)	30%
3. Documentation (with reasonable amount of comments embedded in the code to enhance the readability)	20%
	100%

Be aware of plagiarism! DON'T copy the program file from your friends or classmates. If any identical copy is found, 5% of the coursework marks will be deducted for each of the file owner.

Program Submission Checklist

Before submitting your work, please check the following items to see you have done a decent job.

Items to be checked	☑ / ☒
1. Did I put my name, student ID and lab section at the beginning of all the source files as comment?	☐
2. Did I put reasonable amount of comments to describe my program?	☐
3. Are they all in .java extension and named according to the specification?	☐
4. Have I checked that all the submitted code are compliable and run without any errors?	☐
5. Did I zip my source files using Winzip / zip provided by Microsoft Windows? Also, did I check the zip file and see if it could be opened? ***(Only applicable if the work has to be submitted in zip format.)***	☐
6. Did I submit my lab assignment to Canvas before the deadline?	☐