Lab 3 – Making Decisions in Programming

In this lab you will practice developing programs that involve making decisions based on user input and displaying output messages as appropriate.

Instructions:

1. Do question 1
2. Choose TWO questions from 2, 3 and 4 **OR** do question 5 only.
3. Comment your code appropriately.

Complete the following question:

1. Use the note: **2.0 - Problem Solving Notes and Examples** in your course notes to review the problem in **Example 2: Pizza Deal.**

Once you have reviewed the problem, use the **if – else if – else** structure to code it in Java.

Choose **TWO** questions from 2, 3, and 4 **OR** do only question 5

Use any combination of **if - else; if-else if, etc.** as well as the **&& / ||** logical operators to design, develop and test the following programs:

1. Design, develop and test a mark conversion program that converts a percentage entered by the user to the corresponding level. The percentage should allow for decimals. Use the following relationship: Note

Level 4 – 80% and up and including 100%

Level 3 – 70% and up to but not including 80%

Level 2 – 60% and up to but not including 70%

Level 1 – 50% and up to but not including 60%

Level R – 0% and up to but not including 50%

1. Design, develop and test a program that calculates and displays the speeding fine that a driver would have to pay given the following criteria:

|  |  |
| --- | --- |
| **Speed** | **Fine** |
| Below 0 | Invalid |
| 100 km/h or below | 0 |
| Over 100 km/h up to and including 120 km/h | $80.00 |
| Over 120 km/h up to and including 130 km/h | $150.00 |
| Over 130 km/h up to and including 140km/h | $300.00 |
| Over 140 km/h up to and including 200km/h | $500.00 |
| Over 200 km/h | $10,000.00 |

1. Sandra owns a computer store and she sells computer packages with different options of memory size and hard drive size. Design and test a program that calculates the price of the computer based on a selection entered by the customer.

The prices are the following:

|  |  |
| --- | --- |
| **Hard Drive Size** | |
| **Memory (RAM)** | **500GB** | **750GB** |
| **4GB** | **$599.99** | **$799.50** |
| **8GB** | **$649.99** | **$899.99** |
| **16GB** | **$999.99** | **$1229.99** |

1. The Yahtze Challenge **(can be done instead of questions 2 to 4)**:

In order to complete this challenge question, the Math.random() function must be used to make the dice roll random numbers from 1 to 6. To store the value of a random number from 1 to 6 in the integer variable dieValue1 the following code is required:

**int dieValue1;**

**dieValue1 = (int)(Math.random()\*6 + 1);**

You can find out more about the Math.random() function from the course notes.

The game of Yahtzee is played with five dice. Certain combinations of the five dice are worth extra points:

Three of a kind, four of a kind, full house (two of a kind and three of a kind)

Small straight (four in a sequence – for example 2, 3, 4, 5),

Large straight (five in a sequence),

Yahtzee (five of a kind).

**Note:** if you have rolled a four of a kind, it is also be a three of a kind. If you have rolled a Yahtzee, it is also a four of a kind and a full house. You should only display the best one.

Design, develop and test a program that displays a message based on the randomly generated dice values and their sequences.

**Hints:**

Create variables to hold the values of the dice as follows:

**dieValue1**, **dieValue2**, **dieValue3**, **dieValue4**, **dieValue5**.

You may want to store the number of rolls for each numeric value in the variables such as

**numOnes, numTwos, numThrees, numFours, numFives, numSixes**.

You may assume that the dice are in order from smallest to largest.

Peer Assessment:

1. Have your assigned classmate check your program and fill the table below according to the following legend:

**Evaluation Legend:**

Poor – program not user friendly, no documentation, program does not work.

Fair – Program is user friendly, some documentation, but program works poorly.

Good – User friendly, full documentation, program works with minor errors.

Excellent – Excellent interface, documentation is flawless, program is flawless.

1. Fix any problems identified by your partner.
2. Submit the fixed programs online with this table completed.

Programmer’s Name: Gurpreet Lubana

Assessor’s Name: Bheesham Ramkissoon

|  |  |  |  |
| --- | --- | --- | --- |
| Programs Attempted | Overall feedback  (poor, fair, good, excellent) | Identified problems | Fixed?  Y/N |
| **Mark Converter** | **Poor; Very little of the actual program is in the file.** | **Only one comparison is there, he needs the missing 5 comparisons. He also has no comments whatsoever, and neither a program description.** |  |
| **Pizza Converter** | **Good! Program is fully completed.** | **He has minor spelling mistakes (on the pizza slices, he says “wnat” and not want. He could also use more comments, as he only has 4, and the bottom half of his program is just code, with no explaining as to what he did. An alt tab for some alignment would be nice too, plus a program description at the top. If a negative pizza number is inputted, his program doesn’t catch the error.** |  |
|  |  |  |  |