CST2110 Summative Assessment Part 1

Deadline for submission **Thursday 15th December, 2022. 12:00**

Please read all the assignment specification carefully first, before asking your tutor any questions.

General information

You are required to submit your work via the dedicated assignment link in the 'week 12' folder of the module myUnihub space by the specified deadline. This link will 'timeout' at the submission deadline.

Your work will not be accepted as an email attachment.

Therefore, you are strongly advised to allow plenty of time to upload your work prior to the deadline.

Submission should comprise a single 'ZIP' file. This file should contain a separate, cleaned[1], NetBeans project for the programming task described below. The work will be compiled and run in a Windows environment, i.e., the same configuration as the University networked labs and it is strongly advised that you test your work using the same configuration prior to cleaning and submission.

Important notes (please read carefully)

- Work submitted that is not programmed with the correct Java version (i.e., Java 8) will not be assessed, and will attain zero marks

- Work submitted that Is not configured correctly as a NetBeans 'Ant' build (as described in the lectures) will not be assessed and attain zero marks.

- Your submission (ZIP file) must also include a completed declaration of authenticity using the form provided in the module myUnihub space. Your work will not be marked if you do not complete and submit the declaration.

Additional note (please read very carefully)

Please refer to the presentations on plagiarism and collusion in the module myUnihub space. All assessment for CST2110 is individual. The CST2110 module teaching team regards plagiarism and collusion as very serious matters and apply a zero-tolerance policy. Accordingly, effort is made to check the authenticity of student work which may include the use of (program code) plagiarism detection tools.

[1] In the NetBeans project navigator window, right-click on the project and select 'clean' from the drop-down menu. This will remove .class files and reduce the project file size for submission.

Task (25 marks)

Your task is to write a Java 8 program (as a NetBeans project) to simulate a simple dice game for three players sitting at the console, i.e., taking turns by sharing the keyboard. The game is a scoring game that requires eight dice. The aim of the game is to be the player with the highest cumulative score after three rounds of play. A description of the game follows.

Game description

This game must be entirely text-based, operating at the NetBeans console. This is a requirement of the task. Any submission that does not adhere to this requirement will receive zero marks.

The first player throws all eight dice. After the throw, the player must choose a number that is shown on at least one die and set aside one or more dice with this number showing. Following this, the player has two options: to conclude their turn, or rethrow the remaining dice. The player may only rethrow if there are remaining dice available, but after each throw must choose a different number and set aside at least one corresponding die. The player can continue taking throws but must always set aside at least one die each throw (with a number that has not already been set aside for that turn) until their turn completes. The turn ends in one of the following two ways:

- If all the player's dice show numbers that have previously been set aside (in that turn), the player cannot keep a die and loses their turn (scoring no points – they have effectively *bust* their turn).
- The player can decide to finish their turn after setting aside some dice and the program records the total of all the dice set aside to that point (in the turn) for that player.

After the first player's turn has finished, the next player takes their turn (in the same way). The game comprises all three players taking exactly three turns. The winner is the player with the highest cumulative score at the end of the game.

Example

The first player's initial throw is 6-6-6-5-5-3-3-1 (it is a good idea for the program to display the initial throw in the default randomised order first, but then show the throw sorted to help the player see how many of each die value has been thrown). The player decides to set aside value six. At this point the program must allow the player to specify how many dice showing value six to set aside (this is part of the strategy i.e., the player can elect to set aside just one of the dice showing six, two, or all three dice that are showing six, but must select at least one). In this example, let's suppose the player decides to set aside all three dice showing value six (which would give a cumulative score of eighteen points to this point).

```
Player 1

First throw of this turn, starting with 8 dice.

Enter 't' to throw > t

Throw:  [ 6 ] [ 6 ] [ 6 ] [ 5 ] [ 1 ] [ 3 ] [ 3 ] [ 5 ]

Sorted: [ 6 ] [ 6 ] [ 6 ] [ 5 ] [ 5 ] [ 3 ] [ 3 ] [ 1 ]

Select die value to set aside > 6

There are 3 dice that have that value

You can choose to keep 1, 2, or 3 dice of value 6
```

```
How many do you want to set aside > 3

Score so far = 18

You have kept 3 dice so far.

Finish turn or continue (enter 'f' to finish turn or 'c' to continue and throw
again) >
```

At this point the player might decide (tactically) to accept the eighteen points acquired and end their turn. Alternatively, the player can elect to continue and rethrow the remaining five dice. Let's assume this is what the player decides to do:

```
Score so far = 18

You have kept 3 dice so far.

Finish turn or continue (enter 'f' to finish turn or 'c' to continue and throw
again) > c

Taking 5 dice forward to next throw.

Next throw of this turn.

Enter 't' to throw > t

Throw:  [ 5 ] [ 3 ] [ 5 ] [ 6 ] [ 2 ]

Sorted: [ 6 ] [ 5 ] [ 5 ] [ 3 ] [ 2 ]

You have already set aside: [ 6 ]

You must now select a different die value

You can now select one of the following: [ 5 ] [ 3 ] [ 2 ]

Select die value to set aside >
```

In this example, the value six has already been set aside from the first throw, therefore the player must choose between one of the other values showing i.e., five, three or two (the program has determined this as part of its logic and displays an appropriate prompt). The player decides to set aside value 5, and specifies to set aside the two fives that are showing, as follows:

```
Select die value to set aside > 5

There are 2 dice that have that value

You can choose to keep 1, or 2 dice of value 5

How many do you want to set aside > 2

Score so far = 28

You have kept 5 dice so far.

Finish turn or continue (enter 'f' to finish turn or 'c' to continue and throw
again) > f
```

```
Final score for that turn for Player 1 = 28
```

In this example, the first player has set aside five dice to this point (with values five and six). This leaves three remaining dice with which to rethrow if the player wishes to effectively gamble that the rethrow will result in other values that have not been set aside to this point. The danger is that the rethrow will not enable further setting aside, and the player will have bust their score (which would score zero). In this case, the player decides to finish their turn, retaining the 28 points scored so far. Play then moves on to the second player:

```
Player 2

First throw of this turn, starting with 8 dice.

Enter 't' to throw >
```

Play continues in this manner until all three players have had three turns. The winner is the player with the highest score after three turns each.

Updating the display (console)

After each round (i.e., each of the three players have all completed one turn) the program should display the cumulative score in a tabular format. For example, after the players have all taken their first turn, the program should display something like the following:

```
-----------------------------------------------------
|   Round    |  Player 1  |  Player 2  |  Player 3  |
-----------------------------------------------------
|     1      |     28     |     21     |     19     |
-----------------------------------------------------
|     2      |    ---     |    ---     |    ---     |
-----------------------------------------------------
|     3      |    ---     |    ---     |    ---     |
-----------------------------------------------------
|   Total    |     28     |     21     |     19     |
-----------------------------------------------------
```

After two rounds have been completed, the program should display something like the following:

```
-----------------------------------------------------
|   Round    |  Player 1  |  Player 2  |  Player 3  |
-----------------------------------------------------
|     1      |     28     |     21     |     19     |
-----------------------------------------------------
|     2      |     13     |     18     |     31     |
-----------------------------------------------------
|     3      |    ---     |    ---     |    ---     |
-----------------------------------------------------
|   Total    |     41     |     39     |     50     |
-----------------------------------------------------
```

You can see that after two rounds, Player 3 is leading with 50 points. After three rounds are completed the game finishes and the program displays the final table and announces the winner.

```
-----------------------------------------------------
|   Round    |  Player 1  |  Player 2  |  Player 3  |
-----------------------------------------------------
|     1      |     28     |     21     |     19     |
-----------------------------------------------------
|     2      |     13     |     18     |     31     |
-----------------------------------------------------
|     3      |     33     |     0      |     22     |
-----------------------------------------------------
|   Total    |     74     |     39     |     72     |
-----------------------------------------------------

Player 1 wins that game.
```

You can see in round three that Player 2 ‘busted’ and scored zero points. When this happens, the program should display an appropriate message to the console. Appendix A lists this example of Player 2 busting (tactically poor but done deliberately for demonstration purposes).

Assessment

An assessment rubric is provided in the module handbook.

Your solution will be assessed according to robustness, functionality, and the correctness of program output. In addition, the assessment of this task will consider program modularity, formatting, and code style. In other words, credit will be gained for good use of methods, neat console-based text formatting, appropriate use of Java naming conventions, and appropriate code commenting. Note that you are only required to provide inline code commenting (not ‘Javadoc’ commenting). Code commenting should focus on explaining the key complexities of any methods and algorithms that you have used.

All grades are provisional and subject to adjustment during moderation of the overall module assessment which takes place at the end of the year with external examiner approval.

Appendix A: Example (program trace) of a player busting their turn

```
Player 2

First throw of this turn, starting with 8 dice.

Enter 't' to throw > t

Throw:  [ 1 ] [ 6 ] [ 5 ] [ 6 ] [ 3 ] [ 2 ] [ 2 ] [ 6 ]

Sorted: [ 6 ] [ 6 ] [ 6 ] [ 5 ] [ 3 ] [ 2 ] [ 2 ] [ 1 ]

Select die value to set aside > 6

There are 3 dice that have that value

You can choose to keep 1, 2, or 3 dice of value 6

How many do you want to set aside > 3

Score so far = 18

You have kept 3 dice so far.

Finish turn or continue (enter 'f' to finish turn or 'c' to continue and throw
again) > c

Taking 5 dice forward to next throw.

Next throw of this turn.

Enter 't' to throw > t

Throw:  [ 6 ] [ 5 ] [ 2 ] [ 3 ] [ 4 ]

Sorted: [ 6 ] [ 5 ] [ 4 ] [ 3 ] [ 2 ]

You have already set aside: [ 6 ]

You must now select a different die value

You can now select one of the following: [ 5 ] [ 4 ] [ 3 ] [ 2 ]

Select die value to set aside > 5

Only one die has that value, setting aside the one die with value 5

Score so far = 23

You have kept 4 dice so far.

Finish turn or continue (enter 'f' to finish turn or 'c' to continue and throw
again) > c

Taking 4 dice forward to next throw.

Next throw of this turn.

Enter 't' to throw > t

Throw:  [ 1 ] [ 1 ] [ 2 ] [ 6 ]
```

```
Sorted: [ 6 ] [ 2 ] [ 1 ] [ 1 ]

You have already set aside: [ 6 ] [ 5 ]

You must now select a different die value

You can now select one of the following: [ 2 ] [ 1 ]

Select die value to set aside > 2

Only one die has that value, setting aside the one die with value 2

Score so far = 25

You have kept 5 dice so far.

Finish turn or continue (enter 'f' to finish turn or 'c' to continue and throw
again) > c

Taking 3 dice forward to next throw.

Next throw of this turn.

Enter 't' to throw > t

Throw:  [ 3 ] [ 3 ] [ 4 ]

Sorted: [ 4 ] [ 3 ] [ 3 ]

You have already set aside: [ 6 ] [ 5 ] [ 2 ]

You must now select a different die value

You can now select one of the following: [ 4 ] [ 3 ]

Select die value to set aside > 4

Only one die has that value, setting aside the one die with value 4

Score so far = 29

You have kept 6 dice so far.

Finish turn or continue (enter 'f' to finish turn or 'c' to continue and throw
again) > c

Taking 2 dice forward to next throw.

Next throw of this turn.

Enter 't' to throw > t

Throw:  [ 3 ] [ 6 ]

Sorted: [ 6 ] [ 3 ]

You have already set aside: [ 6 ] [ 5 ] [ 2 ] [ 4 ]

You must now select a different die value

You can now select one of the following: [ 3 ]
```

```
Select die value to set aside > 3

Only one die has that value, setting aside the one die with value 3

Score so far = 32

You have kept 7 dice so far.

Finish turn or continue (enter 'f' to finish turn or 'c' to continue and throw
again) > c

Taking 1 dice forward to next throw.

Next throw of this turn.

Enter 't' to throw > t

Throw:  [ 6 ]

Sorted: [ 6 ]

Sorry, you have busted with that throw.

This ends your turn with no score.

Final score for that turn for Player 2 = 0
```