MITS4002

OBJECT-ORIENTED SOFTWARE DEVELOPMENT

Project (25%)

Tattslotto

50% deduction for Late Submission within one week
0 mark for Late Submission more than one week
0 mark for duplicated Submission or Shared Work

You will be marked based on your submitted zipped file on Moodle. You are most welcome to check your file with your lab tutor before your submission. **No excuse will be accepted** due to file corruption, absence from lecture or lab classes where details of lab requirements may be given.
Please make sure that you attend Lecture EVERY WEEK as low attendance may result in academic penalty or failure of this unit.

Programming Project
Part 1: Arrays & Inheritance
Submission deadline: 5 pm, Monday, Lesson 12

Marks: The project is assessment for 25% of final mark for the subject. The project is composed of two parts: **Part 1** and **Part 2**.

Problem Description

This project is based on the design, and implementation in Java, of the seven different Lottery games being Saturday Tattslotto, Oz Lotto and Powerball.
Details of these games can be found at http://www.thelott.com

In **Part 1** of the project, you are asked to use arrays and inheritance to code versions of these games and in **Part 2** to create an appropriate GUI that writes to a report file.

Note: GAMBLING can be a serious problem for some people.
Your lecturer DOES NOT encourage you to gamble.

Summary of Some of the Lottery Games

(From the Help pages of the above web site)

Game	Day	Description
Tattslotto	Saturday	45 balls numbered 1 to 45, from which 8 balls are randomly selected. The first 6 balls are the winning numbers and the last two balls drawn are the supplementary numbers.
Oz Lotto	Tuesday	45 balls numbered 1 to 45, from which 9 balls are randomly selected. The first 7 balls are the winning numbers and the last two balls drawn are the supplementary numbers.
Powerball	Thursday	35 balls numbered 1 to 35 from which 7 are randomly selected. An eighth ball, the Powerball, is then drawn from a separate machine containing 20 balls numbered 1 to 20.

You will notice from your research and examination of the table that all games have several things in common.
All games

- have a name,
- run on a day of the week
- have a set of randomly generated numbers.

Also, each of the randomly generated numbers have a minimum and maximum possible value, for example: For Powerball, the minimum value is 1 and the maximum value is 35.

An abstract class, LuckyGame can be used to represent the generic concept of a game of chance. A suitable partial design is shown in the following UML diagram. In the UML diagram:

- The LuckyGame class represents the generic concept of a game of chance and therefore, is to be declared as an abstract class. It contains two abstract methods

setNumberOfRandoms( ) and *collectUserInput (String input).* The LuckyGame class also contains an array of String which is used to record the randomly generated numbers used by each object of LuckyGame type.

- The class TattslottoGame is a LuckyGame and it thus should implement code for LuckyGame's two abstract methods as well as its own constructor and any other methods.

- The games SaturdayTattslotto and Oz Lotto are all instances of the class TattslottoGame.

- You can implement the remaining class games in any manner you believe appropriate. For example, class PowerBallGame can be implemented using either using

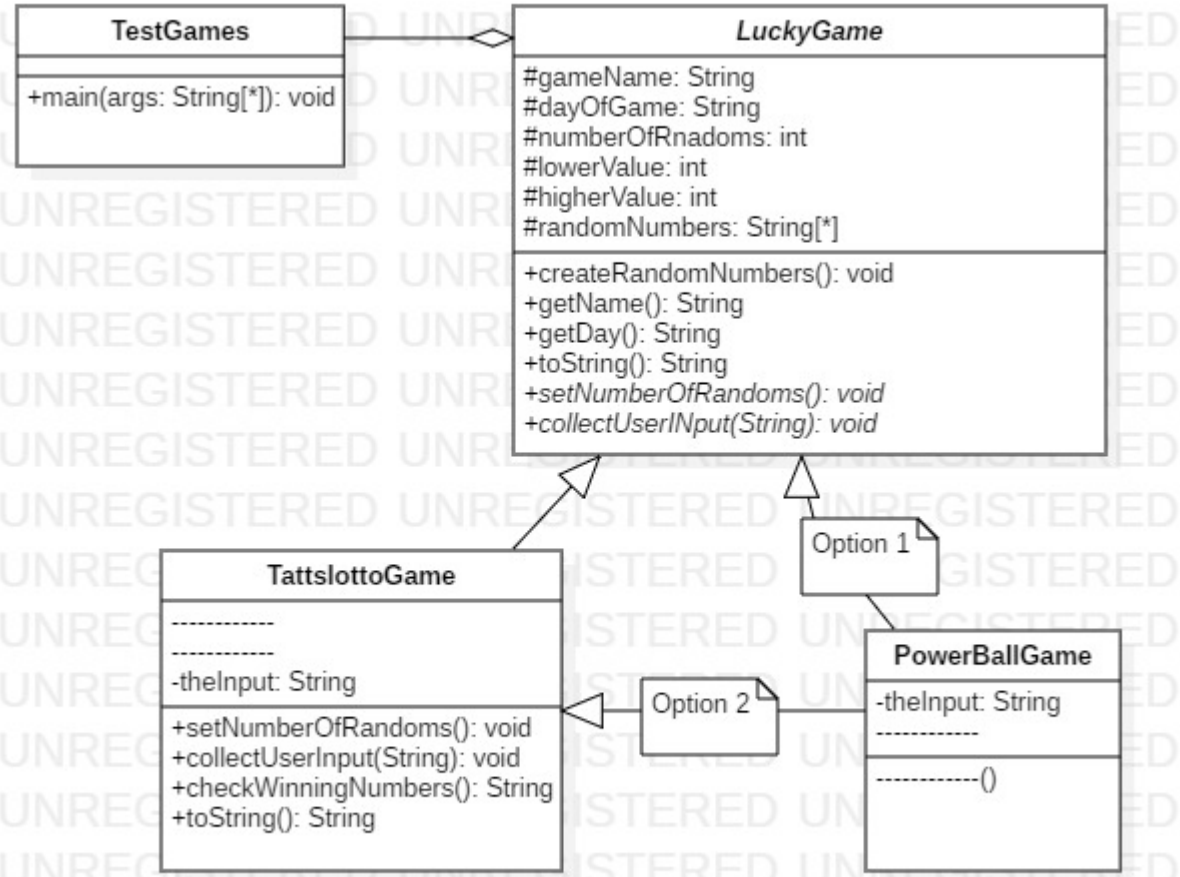
    - Option 1: the class PowerBallGame is a LuckyGame and thus should implement code for LuckyGame's two abstract methods as well its own constructor and other methods.

    - Option 2: the class PowerBallGame is a TattslottoGame and thus should implement code for its own constructor and other methods.

- You have been provided with some code for the class TestGames, which is a text-based application used to create instances of games and test all of their respective behaviours.

- To collect user input for all games, a class UserInput is needed. It should collect user input and deal with any problems in the input, such as repeat numbers or numbers outside the possible range of values.

UML Diagram



Code for TestGames

```
// A class that tests the instances of each class of game
//*******************************************************
*******
class TestGames
{ public static void main ( String [ ] args )
     {
          final int NO_OF_GAMES = 7 ;
          LuckyGame [ ] games = new LuckyGame [NO_OF_GAMES];
          TattslottoGame myGame = new TattslottoGame
("Tattslotto", "Saturday", 1 , 45 ) ;
          TattslottoGame OzLotto = new TattslottoGame ( "OZ
Lotto", "Tuesday", 1, 45 ) ;
          TattslottoGame WedsLotto = new TattslottoGame (
"Tattslotto", "Wednesday", 1, 40 ) ;
          Tatts2Game tatts2 = new Tatts2Game ("Tatts2",
"everyday", 1 , 99 ) ;
          PowerBallGame powerBall = new PowerBallGame
("PowerBall", "Thursday", 1 , 45 ) ;
          ……..// Super 66
instance
          ……..// Pools instance
          games[0] = myGame ;
          games[1] = OzLotto ;
          games[2] = WedsLotto ;
          games[3] = tatts2 ;
          games[4] = powerBall ;
          ……..// Super 66
instance
          ……..// Pools instance
          for (int i = 0 ; i < NO_OF_GAMES ; i++)
          { System.out.print
("\n\n*****************************************\n" );
               System.out.print ("Input your numbers for " +
games[i].getDay( ) + " " + games[i].getName( ) + " :- " ) ;
               String input = Keyboard.readString( ) ;
               games[i].collectUserInput( input );
               System.out.println ( games[i] ) ;
          }// end for
     }//end main
```

Sample output

An example of some output from a run of TestGames. NOTE: ALL OUTPUTS ARE ONLY SUGGESTIONS and should be used as a guide to your implementations.

```
*****************************************
Input your numbers for Saturday Tattslotto :- 1 2 3 3 4 78 1 5 6
Invalid input
Saturday Tattslotto numbers are:
31 29 1 6 10 41 supplementary numbers: 25 38
6 user picks between 1 and 45 are
1 2 3 4 5 6
No. of winners 2 + 0 supps match
*****************************************
Input your numbers for Tuesday OZ Lotto :- 1 2 3 4 5 6 7
Tuesday OZ Lotto numbers are:
2 29 19 21 36 31 40 supplementary numbers: 5 35
7 user picks between 1 and 45 are
1 2 3 4 5 6 7
No. of winners 1 + 1 supps match
*****************************************
Input your numbers for Thursday PowerBall :- 1 2 2 3 4 5 5
Invalid input
The user chosen powerball is 5
Thursday PowerBall numbers are:
1 32 34 5 35 33 and the POWERBALL is:- 5
7 user picks between 1 and 35 are
1 2 3 4 5 6 7 and the user chosen POWERBALL is:- 5
No. of winners 2 and you have the POWERBALL etc…..
```

Programming Project
Part 2: Graphical User Interfaces/Applets & Files & Exceptions
Submission deadline: 5 pm, Monday, Lesson 12

In **Part 1** of the project, you are asked to use arrays and inheritance to code versions of 7 Lottery games. In **Part 2**, using swing classes wherever possible, you need to create an applet for players. The applet should allow a player to choose and run a game of their choice and find out the results of their game. Each time a game is played, the applet writes information about the game to a file called **report.txt.** This information would be similar to that shown as sample output for Part 1.

For example,

report.txt

```
Saturday Tattslotto numbers are:
31 29 1 6 10 41 supplementary numbers: 25 38

6 user picks between 1 and 45 are
1 2 3 4 5 6

No. of winners 2 + 0 supps match
```

If there are any problems creating this file, then the applet reports the problem to the user.

NOTE: It is not expected that you should need to rewrite any of the classes from **Part 1** of the Project. Rather you will create instances of the relevant classes when you need them for **Part 2**. The application for this project should create *at least one* new class LuckyGameApplet, however a good design should divide the computational workload and create/use classes where needed. You may even decide to use Threads and create animation!

Applet Appearance

The design of the applet's appearance is totally up to you; the more colourful and well organised, the better (use layout managers). Choose components that reduce user error, e.g., use check boxes or radio buttons when asking the user to choose the game they wish to play.

As the designer, the functionality of the applet is also up to you, as long as the basic problem description is covered. Feel free to add any extra features that you feel are useful. For instance, you may want to add a quick pick selection for the player, or continually add data to the file over a number of games, run statistics of the games played, etc., etc.

It is necessary to draw a component hierarchy for your final Applet design.

Submission Details

Submission deadline: 5 pm, Monday Lesson 12
Attach the following page to the front of your submission and submit all materials to your lecturer.

What You Have to Submit

Each student submits

1. A document detailing the *design* of your solution in as much detail. It should include an updated UML diagram of the inheritance hierarchy as well as the component hierarchy for the applet's appearance

2. The code for each class in your design. Each class listing should be fully documented commencing with a heading that includes your **name, student number, date written, and lecturer's name,** along with a **brief description of the class**. At the start of each method, there should be a comment clearly describing what the method does.

3. A readme.doc file with information on how to run your program. Include any extra information about your design and program that you wish the marker to know.

4. A word document with evidence of trial runs of your program, i.e. screen printouts of the results where you have tested all the features of your code.

5. Put 1, 2 ,3 and 4 together in one zipped folder. Submit this zipped folder to Moodle

Project Grading

Programs are graded on a 100 marks scale. The marks will be assigned as follows:

- 10 marks- Program meets specification and is OOP in design. Extra functionality is encouraged and rewarded in these marks.

- 5 marks – Inheritance hierarchy

- 5 marks – Component hierarchy of the LuckyGameApplet class

- 5 marks – Evidence of error checking in code, trial runs and screen outputs.

MITS4002 – Project

Student full name:
Student ID:

Markers Guideline

Program meets specification and is OOP in design	10 marks	
Inheritance hierarchy	5 marks	
Component hierarchy of the LuckyGameApplet class	5 marks	
Error checking. Evidence of trial runs & output	5 marks	
Total	25 marks	