Lotto Project

**Module:** Programming, Year 1 – Games Development, Software Development

**Type:** INDIVIDUAL assignment – ~~unless otherwise agreed by your lecturer~~. Guidance and feedback will be given on-line.

**Module Credit: 50% in total for the Lotto project**

**Breakdown: V1-V3 (15%), V4-V5 (25%), Screencast and Technical Interview (10%)**

**Upload Date: Friday** May 1st 2020 @ 10pm. (On-line interview during following week)

**Objective(s):** To practice the design, implementation and testing of an object-oriented software system

DkIT Lottery wants you to design, code, document and test a software system for their internal Lottery syndicate.

* Four numbers are picked at random every week in the range [1,30]
* Lotto players choose 4 numbers
* Player(s) who match 4 numbers share the Jackpot
* Player(s) who match 3 numbers get a consolation prize

***Version 1***

Design and code a simple **Ticket** class for a single entry in the lotto. Test this class in **MainAppV1**.

1. **Ticket** fields: name, address, phone number, 4 numbers chosen for this week’s draw. Choose appropriate datatypes for each field.
2. Constructor, getters(), setters(), toString() as appropriate.
3. Ensure the numbers are sorted when the **Ticket** is created.
4. Write the method ***howManyMatches()*** that determines how many numbers the **Ticket** has matched against the actual numbers drawn in a particular week.

**MainAppV1**

Test the **Ticket** functionality in MainAppV1. Hard-code a number of **Ticket** objects. Also prompt for user-entry of one new **Ticket**.

***Version 2***

1. Design and code a basic **Game** class for a single run of the lotto. Test this class in MainAppV2. Game fields: Array of **Tickets**, Winning numbers, Jackpot amount, Match3 amount.
2. Include appropriate constructor, getters() setters(), toString().
3. Implement Game functionality. For example:
4. ~~addTicket()~~
5. displayAllTickets()
6. displayTicket() – belonging to a particular person
7. drawNumbers() (~~return~~ fill the array of winning numbers, random, sorted)
8. displayMatch() - display the details on tickets who match x winning numbers

Note: when testing your Game class, it may be useful to hard-code the winning numbers rather than always randomising them.

***Version 3***

Improve the functionality of the system – testing as you go in MainAppV3.

Following are some suggestions. **To obtain maximum marks you should include additional similar improvements in functionality.**

1. displayLocalPlayers() – i.e. names on tickets of those whose address contains “Dundalk”.
2. countWhoChoseNumber() – i.e. determine how many tickets had a particular number.
3. Display a menu of possible options for the system user. Code this in MainAppV3.
4. Ensure that **ticket** numbers are in the valid range.
5. Ensure unique numbers only (both winning & ticket numbers).
6. Allow different ranges of random numbers for different games – allowing the user to specify this in main() when creating the Game.

***Version 4***

Test in MainAppV4. This is a substantial change so make sure you complete and back-up V3 before attempting V4:

1. Use an ArrayList for the Tickets in the Game. You will need to include addTicket() in the Game class.
2. Store the Ticket data in a text file and read it to initialise the ticket data.
3. Create a password-protected administrator-only options. Only admin users can create a new game, load the ticket data from file and draw the winning numbers for a game.

***Version 5***

***Implement a substantial documented enhancement of V4 of your project. This version will NOT be DIRECTED or SUPPORTED. Students are encouraged to investigate and implement something unique that we have not covered directly in class contact.***

Some suggestions:

1. Analyse multiple runs of the game for any evidence of bias, showing your results as a simple barchart.
2. Deal with payment for the game – temporarily excluding those who have not paid for this run. Keep track of advance payments - possibly using the date of each run of the game as a reference. Allow this week’s players to use their numbers the following week.
3. Display player names (with their corresponding selected numbers) sorted in alphabetical order.
4. Consider the recording of the payment for tickets and Jackpot calculation (e.g. splitting between winners, increase by %, default, cap).
5. Keep player details on the system even if they didn’t buy a ticket this week.

Note:

* The assignment must be entirely your own work. Plagiarised assignments will not be marked and will be reported to the Head of Department. This applies equally to the individual copying and the individual whose work is copied.
* **Technical assistance, guidance and supervision of the project will be provided on-line~~. If you are working in pairs, you should work with your partner during class time. It is therefore important that you attend class to make the best use of this learning exercise~~. As this is no longer possible, the project is an individual project only.**
* **Students MUST engage with their lecturer in an on-line manner for the duration of this project. *Failure to do so will result in the final project not being marked.***

Software Design

Your solution should apply best practices in the design of the software. These include:

1. appropriate use of methods instead of repeated blocks of code;
2. use of parameters and return types in method definition;
3. decomposition of the functionality of your solution into methods;
4. appropriate use of data types;
5. easy-to-understand user-interface. e.g. easily navigable menu and prompts for invalid user input.

Documentation

The application should be documented appropriately. You must include:

1. source code **comments** in Javadoc style – at a minimum for every class and method;
2. a short description (maximum 2 pages) of any **tests** carried out;
3. ~~sample runs of the system - in the form of~~ **~~screen shots~~**; Replaced by Screencast
4. a short description of any functionality left **unfinished** or not working properly.

Version 5 - Enhancement

Version 4 of the application must be working for version 5 to be attempted. The additional functionality MUST be documented. Marks awarded will vary depending on degree of difficulty.

Interview

You will be required to defend your work in an on-line interview with the lecturer, in the week beginning Apr 27th. During the interview, each student will be asked a number of technical questions relating to the code submitted. Attendance in this interview is mandatory. Failure to attend will attract a mark of 0% in the project. A timetable with names and interview times will be made available in Moodle.

Screencast

Please use Screencast-o-matic <https://screencast-o-matic.com/screen-recorder> or Open Broadcaster Software https://obsproject.com/ or a similar app to produce a screencast demonstrating your code working. You should upload the screencast to Moodle as an MP4. Screen casts should be no longer than five minutes. You must use a voiceover to explain what you are doing.

Submission Requirements

1. The working source code must be submitted (\*.java). Comment any code that is not working.
2. All source files and documentation must be submitted in a single (zip format) file.
3. Assignments should be submitted through Moodle by the due date. Failure to do so will result in a penalty being applied.