Java Sets Assignment

By Aleksander Barczak 2497555

**An introduction stating the problem:**

Create a program for the user to experience the thrill of the lottery and watch the infinitely exponential growth of decreasing win ubiquity as the range of available numbers increases towards infinity, adapt the program from then on to allow multiple, users to try their lottery numbers, allow lottery numbers to be tried several times in a row to simulate a weekly expenditure certain to cause haemorrhaging of anyone's savings. Then display the losses to the poor souls.

**Requirements:**

1. Your program SHALL use the java.util.Set interface. Suggested is using the Set interface with the HashSet implementation.

Completed with no issues

2. Allow a player to select 6 lottery numbers, validate them and store them in a Set.

Completed with no issues

3. Run the lottery, i.e., generate 6 numbers between 1 and LOTTERY\_MAX using a random number generator, placing these in a set. Note: make sure that you end up with 6 numbers in your Set!

Completed with no issues

4. Find out if the user has won, and if so, how much they have won. Display this to screen.

Completed with no issues, though the Eclipse display does not render the ‘£’ symbol

5. Enable the user to define the range of lottery numbers to use, e.g. a valid range might run from 1 to 11 for a small lottery, which would give each player a greater chance of winning, your program should be able to run with the range specified by the user. (This makes testing and marking easier, too!)

Completed with no issues

6. Extend your program to ask the user for a number of weeks of lottery draws they would like you to run with the same set of lottery numbers. Then run the draw this number of times with the same set of user lottery numbers, checking whether they have won anything. This means that the user keeps the same set of lottery numbers for each week, but a different set of lottery numbers are randomly generated for each week. Finally output their total winnings and calculate if this exceeds the amount they spent on tickets.

Completed with no issues

7. [Optional] Extend your program to cater for more than one player with each lottery run. Ideally this should be a variable number of players specified by the user, but some marks will be awarded for a fixed number of players if this requirement proves too challenging to implement. Each player should have their numbers checked against the same set of “winning” numbers each week.

Completed with no issues

8. [Optional] Modify your design to create your own set class (e.g. MySet) to handle the complexity of the set operations, including an intersection method. For example so that you could call MySet setc = seta.intersection(setb), where seta and setb are instances of MySet.

Design has been modified to ‘create’ my own class, but not use, I find the use of the wrapper classes for such simple operations to be convoluted and unnecessary, it also brings about difficulty when using “retainall” for the intersection because to create a copy you need to clone() the inner set variable, which is not possible when you use a ‘set’ interface because it does not inherently hold that it implements the ‘cloneable’ interface and then starts to throw errors while compiling, avoidable but unnecessary and annoying

**Your designs using including pseudocode (don’t forget your class design) and in particular,**

Lottery class:

Fields:

Integer lotteryMax - defines the maximum range of the lottery numbers

LinkedList<LotteryPlayer> Players - a list of all the weekly players

Methods:

Lottery() -

Initialising method, sets base values

Main - startup method, creates lottery object instance and launches menu

menu - allows user to pick from a number of options including:

1. Running the lottery once
2. Changing the range of the lottery numbers
3. Adding a player to the weekly players
4. Running the weekly lottery for a number of weeks
5. Clearing the lottery of players
6. Leaving

runOnce() - creates a player instance, asks for numbers, returns winning if applicable

changeMax() - changes the maximum value of the lottery numbers and clears the weekly players to prevent numbers beyond the acceptable range

clearPlayers() - allows the user to clear the list of weekly players

addPlayer() - creates player instance and adds it to the list of players after getting all the numbers for the player

runLottery() - runs the lottery for a number of weeks decided by the user comparing the fixed numbers of weekly players to the changing winning numbers and displaying the winnings and numbers

LotteryPlayer class:

Fields:

Integer lotteryMax - to keep track of the highest number the player can try

Integer ID - keeping track of the individual weekly players

Integer spendings - how much money lost

Integer winnings - how much money gained

Methods:

LotteryPlayer() - initialising method to create base values

getID() - returns the playerID

getPlayerNumbers() - choosing the six numbers the player hopes to win with

getLotteryNumbers() - returns the player’s numbers

displayPlayerAndNumbers() - does what it says on the box

getWinning() - returns winnings

getSpendings() - returns spendings

getProfit() - get winnings - spendings

runGame() - have the player see if his numbers win against the passed in winning numbers

**make sure you explain how you are implementing your set operations**

Through the javadocs docs.oracle.com/javase/7/docs/api/java/util/Set.html ?

**Your test plans and completed test sheets**

Not a test plan because I can’t stand it but has been tested thoroughly to make sure all the inputs are validated, all the logical errors such as setting the range too low are not possible and every feature is working as intended

**A short, critical self-evaluation of how you got on during this assignment. What were the key things that you learned? Did you manage to complete all the tasks? What difficulties did you have? How did you solve any problems that you had? Give some examples. What are you most pleased with in your work? What topics do you need to review again?**

I’m likely taking the documentation aspect of this too casually, due to my sheer hatred for pseudocode and testing plans, they just ruin my momentum and keep me from working in an AGILE manner, very much a preferable approach to this kind of work in my opinion. In terms of evaluating the actual code I’ve provided it is undercommented, under ‘javadoced’ and I would say still very readable. The only gripe I have is the insistence from your part to use a wrapper class when it is just unnecessary for this task, the intersection method of copying and using retainAll() is cumbersome due to having taught us do declare fields as Set, and not HashSet, which leads to compilation errors telling us that the Set interface is not cloneable, due to it not being a class no despite the fact that we force it into benign a HashSet, casting does not work either, tried it.

I hope you appreciate the fun I was trying to have while writing up this document, it’s not perfectly serious, but it still has all the information there… unless it was something like the test plan which i just refuse to do