Assignment 2 (due 2 pm Monday of week 5)
Sets: Lottery simulation

New this week: *Sets. Library classes: HashSet, Set, Iterator, Random*

Deadline for Submission: Monday 17th February, 2020 2pm.

Hand in Method: Submit on MyDundee. The submission link will not be made available until Thursday 13th February. Note that this is an individual assignment (not groupwork!).

Date Feedback will be Received by: Latest date to receive feedback will be Monday 9th March.

Penalty for Late Submission: One grade point per day late (meaning if a submission is one day late and marked as a C2 it will receive a C3 grade) A day is defined as each 24-hour period following the submission deadline including weekends and holidays. Assignments submitted more than 5 days after the agreed deadline will receive a zero mark (AB).

Percentage of Module: This assignment is worth 10% of this module for AC12001 and 7% for AC22007.

Overview of Assignment

- Sets & HashSet
- Java collection classes
- Random number generation

Assessment Details

Everyone should attempt the first five requirements below (out of 80%); you should only attempt the optional requirements if you feel confident that you have finished all the others and wish to try some more.
Remember, your report must state which requirements were met and what problems, if any, you encountered.

Aim

The aim of this lab is to learn how to design and code a program that allows you to ‘play’ a lottery game using the Set interface from the Java collection classes (`java.util.Set`).

Background

The lottery you are simulating is a simplified version of a National Lottery.

- Each player chooses 6 unique numbers (between 1 & lottery_max) for £2.00
  - Define lottery_max max as a low value **int** variable (field) (e.g. 10) to test your program and then you could set it to whatever value you think sensible.
- When the draw is held the computer randomly selects 6 unique numbers (between 1 & LOTTERY_MAX)
- You win a prize of £25 if you have matched 3 numbers, £100 if you have matched 4 numbers, £1,000 for 5 numbers and £1,000,000 (or your ‘jackpot’ value) for 6 numbers.

Requirements

Produce a program to simulate the lottery above. Your program MUST use the `java.util.Set` interface. *I suggest using the Set interface with the HashSet implementation.*

1. Allow a player to select 6 lottery numbers, validate them and store them in a Set.
2. Run the lottery, i.e. generate 6 numbers between 1 & lottery_max using a random number generator and place these in a set too. *Note: make sure that you end up with 6 numbers in your Set!).*
3. Find out if the user has won, and if so, how much they have won, display this to screen.
4. 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!).
5. Extend your program to ask the user for a number of weeks of lottery draws they would like you to run. Then run the draw this number of times, checking whether they have won anything. This mens 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.
6. *[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.
7. *[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`.

Method

You may implement the above requirements in whatever manner that you feel is most appropriate with the only stipulation that you must use the `java.util.Set` interface to implement your sets. The steps below lead you towards one possible design implementation. Remember, start by designing a solution then code it in small sections, define a test plan, test as you code and complete test sheets after implementation.

1) Design your Lottery class, so that it can perform the tasks required by Requirements 1-5.
2) Design a test plan
3) Create a new Java project and add Lottery and Tester classes to it.
4) Implement your Lottery class:

Start your implementation one bit at a time, for example:

- Create two Set objects and add integers to them (e.g. sets to hold the user numbers and the lottery numbers).
- Start by just hard coding what the user's choice of lottery numbers is and the numbers which the computer has drawn.

- Check your program can add these numbers correctly to a set and print out the sets by writing automated test code to create the set objects, put data into them and print them out.
- Implement the method to perform the intersection. Ensure that this does not lose any information (either the user numbers or the draw lottery numbers). Write test code to test this.
- Implement the method to calculate the winnings from the intersection set. Write more test code to test this!
- Extend your program to request lottery numbers from the user and validate them before then comparing them to the drawn numbers and calculating the winnings.
- Now use the built-in **`Random`** class to automatically generate the numbers for the lottery draw. Check that the numbers produced are in the range requested, there are no duplicates, and that they are added correctly to the set. Now when you are testing you will win less often ☹.

5) Implement your Menu.

6) Enable the user to set the valid range of lottery numbers from a Menu option. E.g. a valid range might run from 1 to 21 for a small lottery which would give each player a greater chance of winning.

7) Enable the user to run the lottery over a period of multiple weeks (let the user specify the number of weeks) and output the winnings minus the ticket costs. Between weeks, don't forget to clear out the draw & intersection sets.

8) Re-run your tests your automated tests and fill in the results in your test plan

9) Ensure your Classes and methods all have Javadoc comments and regenerate the Javadoc HTML documentation.

10) Create a JAR file and test your program outside the IDE.

[Possible methods for optional extras]

11) [*Optional*] Add the capability for several players - you may wish to use a fixed number at first and extend to a variable number if you have time. At the end, display the total winnings versus the total cost to the player.
12) [*Optional*] Design a new class which will be your set class. Note that this can't be called 'Set' otherwise it will conflict with the Java library class which is also called Set. So, call it 'MySet' or 'LotterySet' or something similar. Add a field to this class of interface type *Set* to store and process your set data.

- Your set class design could include the following methods…

  i) printSet() // display the contents of the set data to the user

  ii) addToSet(int number) // Add a number to the set

  iii) isSetEmpty(), // returns true if the set is empty, false otherwise

  iv) getCardinality() // returns the cardinality of the set

  v) isInSet(int number) // returns true if the given number is in the set already

  vi) intersection(MySet setb) // performs a set intersection operation by creating a copy of the current set, doing a retainAll with setB and returning the result as a new MySet object

- Implement your Set class and check that it works by replacing the Set operations in your Lottery class with MySet objects and method calls.

- Complete test sheets for the Set functionality crucial to your Lottery program.

Marking Scheme

Requirements 1-5	Report & Designs (pseudo code and/or flowcharts)	out of 10%
	Test plan and completed test sheets	out of 10%
	Code	
	Take user input and add to a set	out of 10%
	Generate random lottery numbers and add to a set	out of 10%
	Calculate winnings (including set intersection)	out of 15%
	Enable user to define range of lottery numbers.	out of 5%
	Simulating a lottery over several weeks.	out of 10%
	User interface, including Menu and input validation	out of 10%
Requirement 6	Extension to several users	out of 10%
	Implement a MySet wrapper class to handle the set operations.	out of 10%

Please note that well-written code, commenting and a main method have not been specifically allocated marks. This is because it is expected that you will include these in all assignments and that marks will be deducted for any type of bad practice (e.g. poor use of static keyword, lack of commenting or missing Javadoc html comments, badly written code or no runnable JAR file).

Submission

Submit a ZIP file by the deadline given to the link on My Dundee (Assignment→Assignment 2)
Your .zip file will be labelled with your surname, first name and the assignment number, e.g.
smith_fred_assignment_2.zip

This file will include:

- A single written report which will include:
  - An introduction stating the problem
  - A summary of the requirements – saying which you were able to tackle and which were successful
  - Your designs using including pseudo code (don't forget your class design) and in particular make sure you explain how you are implementing your **set** operations.
  - Your test plans & completed test sheets
  - An evaluation of the lab – detailing problems and how you tried to solve them, stating anything you couldn't fix
- All the JAVA source code files you have used for your assignment (i.e. just include the full eclipse project folder if you are using Eclipse, otherwise ensure that you have included all your java source code files).
- The Javadoc html, remember to regenerate this after all coding changes have bene completed.
- A runnable JAR file and batch file to run it.

This lab contributes 10% of your coursework marks for AC12001 and 7% for AC22007. Remember to SUBMIT whatever you have done by the deadline, rather than over-running and handing it in late - this should avoid you getting behind.