**Euromillions Project**

This project was developed to test the odds of winning the EuroMillions jackpot. The winning numbers and Lucky Stars were generated randomly for both the game and the player. This had to happen at the start of each game and, the players numbers were then checked against the winning numbers at the end. The odds of winning the EuroMillions Jackpot are 139,838,160. This is taken directly from the Lottery.ie website. As such, this is the number of times the program would run. I found this interesting for two reasons:

1. I could test how accurate these odds were
2. It would give me an idea of how fast java is along with the performance of my laptop

A screen shot of a computer screen

Description automatically generated

As the first attempt was in the early days of my programming journey, it proved to be very inefficient. Also, there were 187 lines of code. This was not a disappointment as it provided me with an opportunity to research and learn even more. The code above shows the logic for generating the winning numbers, and this logic was also applied for the Luck Stars and the players numbers.

When I sat down and reviewed this program, it became clear that it needed to be improved. The program had to do too many checks and perform too many calculations. I was also using several lists and searching each of them for every Quickpick.

We can see the results of the full program run, in the image above. I had checked for anything from 2 matching numbers up to the 7 matching numbers for the jackpot. Although the results are accurate, the important Information I got from the output is the very last line, 302069. This number represents the programs total runtime in milliseconds. This equates to ~302 seconds or,

5 minutes! This wasn’t good enough.

**EuroMillions Part 2**

I took a bit of time and thought about how I could make the program more efficient. As I could see from reviewing it, there were too many cross checks being performed between ArrayLists that held the winning numbers, winning lucky stars, players numbers and players lucky stars. I sat down and started looking into different types of lists until I found the one most suitable for my use case. I began writing the program again but this time, I used a HashSet. This was ideal a HashSet does not allow duplicates which meant there was no need to check if the winning number was already in the list. It couldn’t be. This had a huge impact on the number of lines and complexity in the code. The lines of code reduced from 187 to 99.

A computer screen shot of a program code

Description automatically generated

As we can see from the screenshot above, the complexity in generating the winning numbers have been reduced significantly. Also, I had taken out the checks for 2,3,4 numbers etc. as I just wanted to focus on how many times the jackpot was won.

The really satisfying part of this second program is the time. The first program had a runtime of ~5 minutes but this time, the runtime had been reduced to 97 seconds – 1 min 37 seconds.

To finish talking about this project I guess I should convey the overall results. As the second program was much quicker and ran almost 140 million times, I repeatedly ran it a number of times to see if the results were consistent. This did not prove to be the case, on the first 2 runs the jackpot was not hit. On the third run the jackpot was hit once and on the fourth run the jackpot was hit 3 times. I continued until I had run the project 10 times. After the tenth run, the number of times the jackpot was won divided by the number runs was ~1. In other words, although I did not hit the jackpot on every run of the program, over the course of multiple runs the odds of winning the jackpot proved to be incredibly accurate.