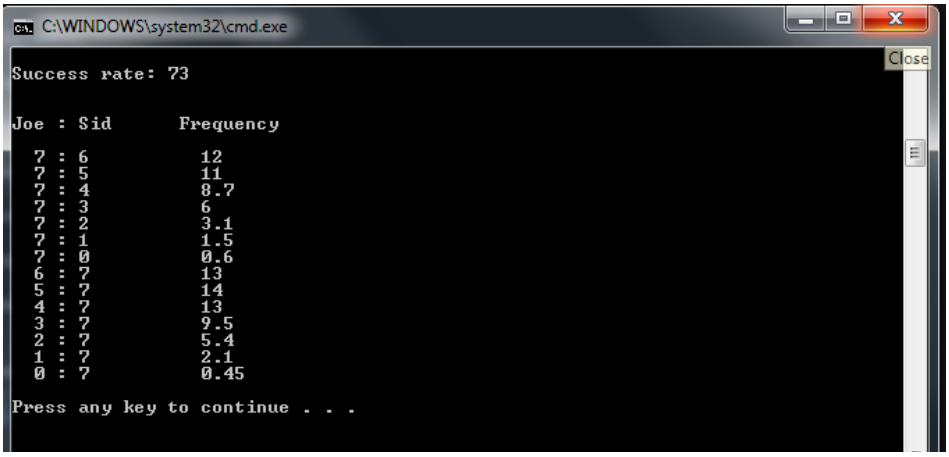
Report – Software Design CMP102 Ethan Hastie

Results (debugged 5 times) of if Sid goes first



Joe : Sid	Frequency
7 : 6	12
7 : 5	11
7 : 4	9
7 : 3	5.7
7 : 2	3
7 : 1	1.3
7 : 0	0.49
6 : 7	13
5 : 7	14
4 : 7	13
3 : 7	9.2
2 : 7	5.6
1 : 7	2.4
0 : 7	0.45

Joe : Sid	Frequency
7 : 6	13
7 : 5	10
7 : 4	9
7 : 3	5.7
7 : 2	3.3
7 : 1	1.2
7 : 0	0.44
6 : 7	13
5 : 7	14
4 : 7	12
3 : 7	9.6
2 : 7	5.3
1 : 7	2

0 : 7	0.62

Joe : Sid	Frequency
7 : 6	13
7 : 5	11
7 : 4	9.2
7 : 3	5.6
7 : 2	3.2
7 : 1	1.2
7 : 0	0.38
6 : 7	14
5 : 7	13
4 : 7	12
3 : 7	9.4
2 : 7	5.7
1 : 7	2.1
0 : 7	0.51

Joe : Sid	Frequency
7 : 6	12
7 : 5	11
7 : 4	8.3
7 : 3	6.1
7 : 2	3
7 : 1	1.3
7 : 0	0.4
6 : 7	14
5 : 7	15
4 : 7	13
3 : 7	9.4
2 : 7	5.2
1 : 7	2.2
0 : 7	0.51

Joe : Sid	Frequency
7 : 6	12
7 : 5	11
7 : 4	8.9
7 : 3	5.6
7 : 2	3.1
7 : 1	1.3
7 : 0	0.37
6 : 7	13
5 : 7	14
4 : 7	13
3 : 7	9.8
2 : 7	5.5
1 : 7	2.4
0 : 7	0.52

Averages:

Joe : Sid	Average Frequency (with program debugged 5 times)
7 : 6	12.4
7 : 5	10.8
7 : 4	8.88
7 : 3	5.74
7 : 2	3.12
7 : 1	1.26
7 : 0	0.416
6 : 7	13.4
5 : 7	14
4 : 7	12.6
3 : 7	9.48
2 : 7	5.46
1 : 7	2.22
0 : 7	0.522

Therefore, the most common set score over 10,000 matches is 5 : 7 with 14%; Sid will usually win 7 sets with Joe winning 5 sets. However, this doesn't compare with the mock data that was sampled to me in the assignment documentation as it was stated that 4 : 7 had the highest frequency compared to the others, while my program displayed otherwise. This could be due to the effectiveness of our algorithms: was my algorithm that I implemented more simpler in determining how the player threw their dart or was the algorithm that was used to output the mock data more effective? Despite this, I find that this must be an average output of data or it could be that it's ambiguous in that it doesn't matter what the data looks like – it could be anything and it depends on the effectiveness of the algorithms used to throw the darts, but it still is a typical sample of data in this case and I have managed to complete this task so it's correct.

Documentation

Note: I only tackled the first task (simulate a simple game of 501 darts over 10,000 matches) and not the second or third that was supplied to us.

When I was first given the darts tasks, I found it quite difficult. This was noticeable at the start of the semester when I was given tasks that were needed to build on our knowledge that would help us complete this assignment. I initially struggled to understand classes and how it was structured which led me to fall behind at the very start of the semester. I spent so much time on trying to understand how to figure out why my week 1 darts program wasn't working, which led me not to focus on the bank of abertay work we had to do. I

had to take a step backwards and I realised what I was doing was way too complicated, especially for a simple program or what they wanted us to believe was a simple program. I restructured my program and it finally managed to work. After I managed to get it working, I realised I could use this same format to help in the later weeks that would help me to build on what we needed to do for the assignment, after all the whole purpose of this assignment was to reinforce the same constructs we used last semester, how we could use pseudocode to plan out our code and new concepts that we learnt such as object orientated programming. The darts programs may have seemed scary at first but it was to get us understanding the same concepts we had learnt beforehand. It then came to Darts 301 and I had to plan a UML diagram and pseudocode before I began to implement the program. At first, I didn't want to do pseudocode because I felt it would consume a lot of time and I already felt I knew what I was doing, because I understood how my darts week 1 program worked but I eventually did so in order to help me later on. For my Darts 501, rather than immediately moving onto coding the matches system, I decided to plan my code in stages because there was so much to this assignment that I didn't want to lose track of my progress. I also did this because I needed to code an algorithm that would determine what type of dart the player would aim for and what dart score, so if I were to immediately implement matches I might have had errors with my algorithm and it would take a long time to find out the error as using breakpoints to find the errors was simply ineffective. My first step was to implement the program interactively – having one player decide where to throw the dart and what number they wanted to aim for. My second step was to add another player so they could both play one game interactively, and it worked. The next big step was to implement games (i.e. simulations into my game) and having simply no input to throw a dart. So I had to implement an algorithm that would determine what type of dart and score the player wanted to go for. I initially thought my algorithm worked until I realised it didn't. I came to the conclusion that my score would get 'stuck' – this meant that if the player had say a score of 3 the player would keep aiming for random scores causing the score to go bust or if the player had a score of 8 and they weren't on the final dart they would keep throwing double 4 causing their score to be reset. For weeks, I tried to figure out how the score was getting stuck but I couldn't reliably tell if it worked because it only sometimes worked. If I were to tell it to run 1 game it would work and print the necessary data, but at other times it wouldn't work – this was because we were supposed to randomly generate numbers so my program wouldn't reliably work half the time. I then decided that the algorithm I used in the Game class called the throw dart function I would implement inside the actual source file. This was to make it more simple and effective and as a result it reliably simulated games correctly and always ran correctly. The next stage was to implement the sets system which I implemented in the same way as simulations with a bit more thinking as it needed to run 13 sets which would be one game and the first person to reach 7 sets would win that match. It wasn't too complicated but I had to think a lot of what code I needed to plan out. The final stage was to implement matches as I needed to implement the matches for the output. This was a bit like the sets as it was somewhat easy to implement but it did require a bit of thinking as to how I was going to change my code. Eventually I did manage to get it working and

displaying the data over 10,000 matches that displays all the set ratios and the frequencies associated those ratios. I got relatively the same data as that in the assignment.

As for the benefits of object orientated programming, I struggled to see the benefits of it at the start of the semester. When we were first taught it, I said to myself that I'm able to actually do this procedurally rather than using object orientated concepts. I also found it difficult to understand it at first because it felt like we jumped into the deep end, but I eventually understood why we needed to use object orientated programming. Had I not used object orientated programming in my project, my source file would be covered in functions and it would be too hard to manage them. Therefore by using object orientated programming it's made it so that my functions are more spread out between the different files and reduced my variable usage. It's also encouraged me to think more logically of programming by breaking problems down and implementing more effective code. Overall, while I did struggle to see the benefits of using object orientated programming, I eventually realised that it was to help me break down problems into pseudocode more effectively rather than using procedural programming to clutter my source file with a lot of functions and keeping my code more organised.