Matthew Regan

Object-Oriented Programming

**CONTENTS**

[Task I – Requirements 2](#_Toc508373535)

[Task I – Analysis and Design 3](#_Toc508373536)

[Task I – Testing 5](#_Toc508373537)

[Task II – Requirements 7](#_Toc508373538)

[Task II – Analysis and Design 8](#_Toc508373539)

# Task I – Requirements

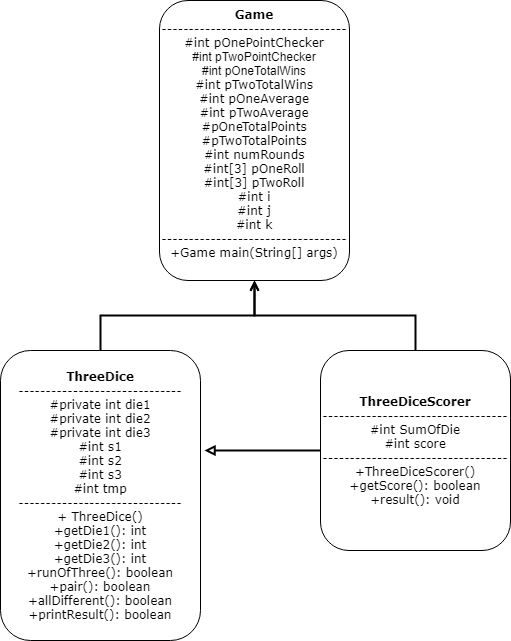
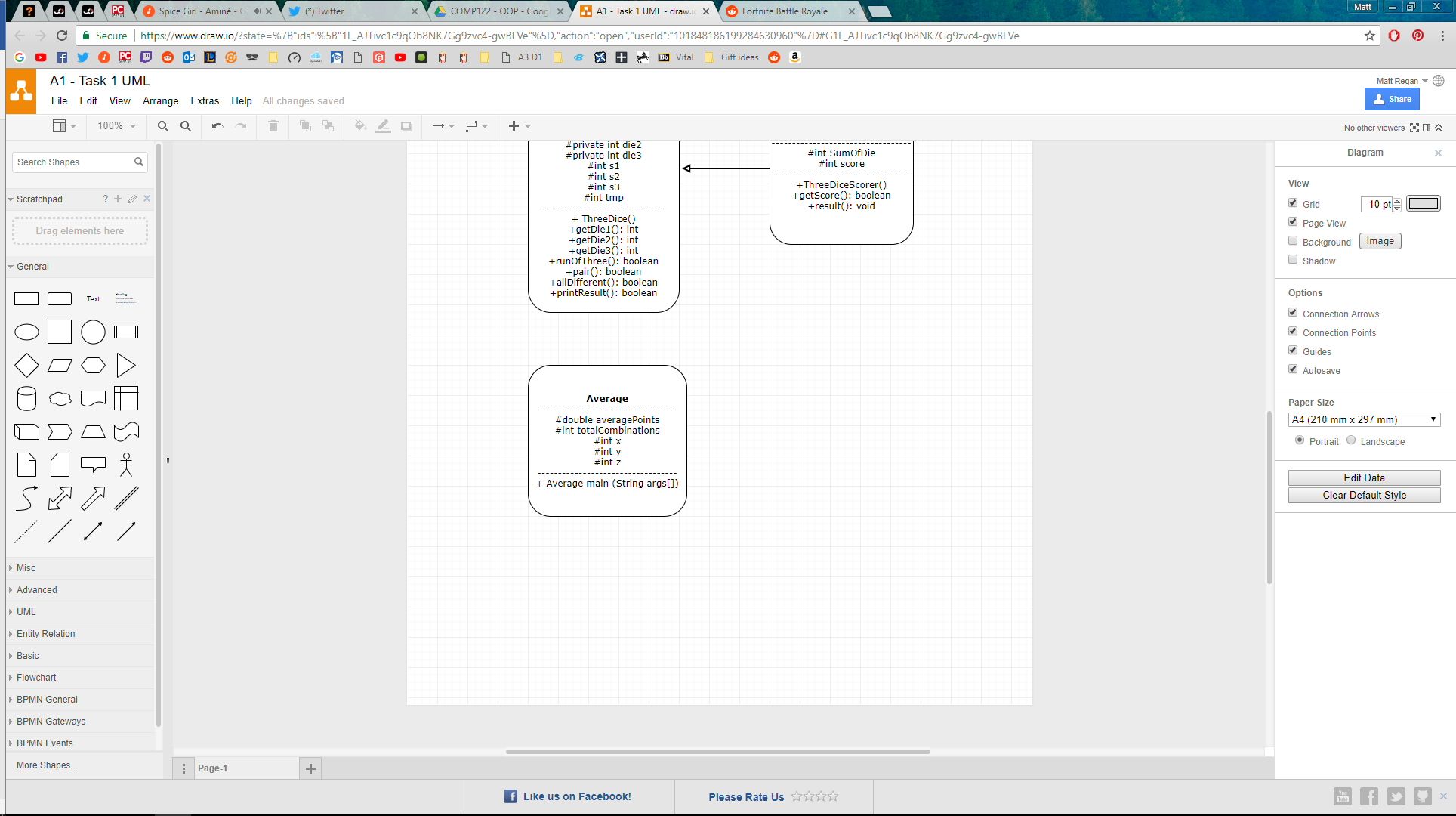
Task I

I have been tasked with creating a game that simulates the rolling of three dice. The game will be played between two players who chose the number of times the set of three dice will be rolled (rounds). Each player will accumulate points at the end of the round based on the outcome of their dice roll. The winner of the game is the player with the most points at the end of the last round.

# Task I – Analysis and Design

Analysis of the problem

Continuing from my summary of the requirements above, during the runtime of the game my program needs to highlight the number of rounds each player has won, the average points each player rolls and the highest point roll a player has had. To help me start I have been given a java class file which create variables and methods for handling the three dice rolls as well as the definitions of winning rolls (pair, three same, etc.). This will likely act as the superclass for my program with classes like ThreeDiceScorer.java inheriting the methods that determine if the player has a winning roll.



Psuedocode

ThreeDice.java

Three objects are created to represent the individual dice. Each die is assigned an integer as its object type. This is because the dice will handle a value between 1 and 6.

Using if statements and placeholder values (s1, s2, s3) we can organise the dice in ascending order. Die one is assigned to the lowest outcome from the randomly generated numbers and die three is assigned to the highest outcome.

A public method is created for each of the die so that when we can call the method from another class the value of the called upon die is output.

Boolean methods are created to return true or false values depending on if the outcome of a dice roll meets the requirement for the roll (threeSame, pair, etc.).

PrintResult method is created at end of class to test if the Boolean methods are correctly identifying dice rolls.

ThreeDiceScorer.java

Extends the parent class ThreeDice.java as we will need all three dice variables.

Declare the constructor of the subclass. We need to pass the die rolls from threeDice.java into this class.

Public method with “if” and “else if” to calculate the score of a dice roll. We use the methods declared in threeDice.java to determine if the most recent dice roll was threeSame, runOfThree etc.

At the bottom of the class is a private method which calculates the sum of the dice roll by adding the three dice variables that we inherited earlier in the class. This method is called in each calculation of the “if… else if” method above.

Game.java

The main method will exist within this class. This means we need to initialise the game state such as number of rounds to be played as well as clarifying the range of numbers that can be rolled from a dice (currently any number can occur).

We need to read in user input here, so we will import the “java.until” function in order to utilise the scanner function.

Ask user the number of rounds they want to play, (zero or above) validate and save their input into a variable. This variable will be used to validate their input and ensure the game ends after the given number of rounds.

While loop will be used to check the users input is valid (zero or greater).

# Task I – Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test No. | Test data | Output | Expectation/Comments |
| 1. | User enters zero for the number of rounds to play  numRounds = 0 |  | As expected |
| 2. | User enters a minus integer  numRounds = -1 |  | As expected |
| 3. | User enters letter for the number of rounds to play  numRounds = “abc” | Error, program crashes | Currently no exception handling in place if a user enters a string into a field expecting an integer.  Likewise entering a floating-point number will case the same error. |
| 4. | Dice roll = 1 3 5  All different |  | As expected |
| 5. | Dice roll = 1 2 2  A pair |  | As expected |
| 6. | Dice roll = 2 3 4  A run of three |  | As expected |
| 7. | Dice roll = 6 6 6  Three the same |  | As expected |
| 8. | Game length of 2 rounds |  | As expected |
| 9. | Total scores for player one and two over 3 rounds |  | As expected  Average is also calculated but I think because I am using exclusively integers the decimal point isn’t stored so the average is slightly out. |