Report Document – Daniel Dixon

Contents:

Application Features:

Game Type:

When the user initially enters the program, they should decide what game type they want to play, whether they want to play versus another player or the AI created for the game. This then decides how the game will go forward.

Player Name Selection:

After the previous choice, Player 1 will be given the choice of giving themselves a name if they choose to. They can enter no or a variation of that word to instead keep the name player one. This is then extended to player two if the game type was not versus the AI. If the AI is part of the game it will be given the name Computer.

Dice Throw:

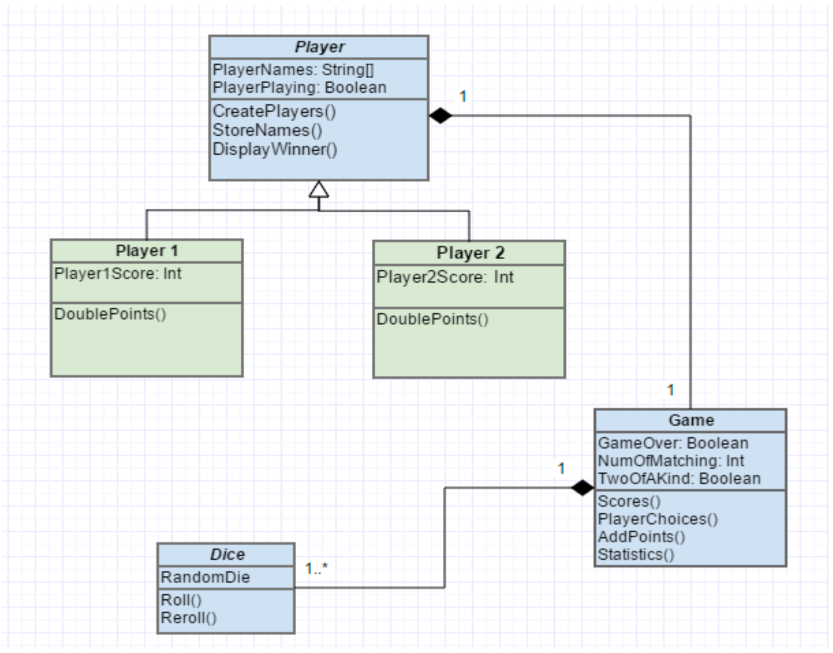
The game will create new random instances five times for the first roll and then calculate any points in coordination with how many matches occur. If only two of the die match three more instances should be called to reroll. If this second roll occurs the matches calculation will be calculated afterwards.

Points System:

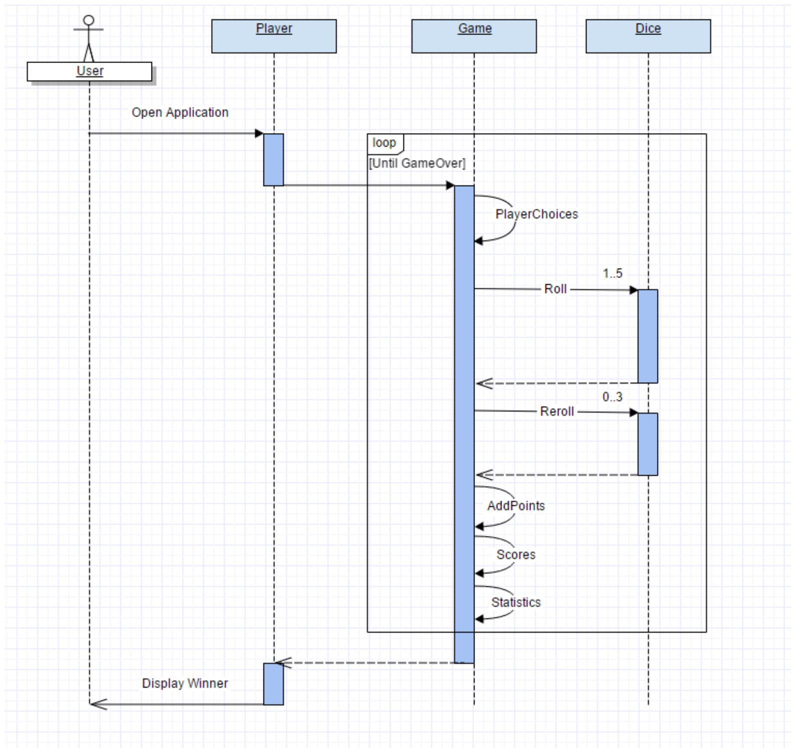
Points will be divvied out as stated in the brief, 3 points for 3 matches, 6 for 4 matches and 12 for 5 matches. It will also be doubled if double points are chosen by the players.

UML Diagrams:

Class Diagram:



Sequence Diagram:



Stretch Exercises:

Player History and Statistics:

Every turn the players dice roll will be stored with their name and then displayed at the end of the turn how the brief states as well as also including the total for each player. Statistics for the game are also provided, this includes the average of each dice being rolled that turn, the total of all the dice per throw and the average for all dice thrown. It also includes the number of throws where no points were gained. This was done by creating global variables and assigning them where necessary in several places in the code.

Double Point Gamble:

The player gets to decide whether they wish to only roll the dice once for the chance to win double points, this feature was implemented by adding onto the points system in place with a few if statements.

AI Opponent:

If there is only one player then a computer opponent will instead be created. This opponent has very basic AI in that there name will automatically be named Computer and their choice of doubling their points will be chosen using a random instance, the if statement implemented with this instance is not equal for both doubling scores or not as an attempt to make it appear more random a choice.

Data Structures Applied:

To effectively store and retrieve data depending on the data being used, several data structures were used.

Dictionary:

To store the player names a Dictionary was used, this is due to it being needed to be accessed depending on who the player was that was playing at the time. It also meant a second piece of information could be stored with it if necessary such as the Player score or their most recent dice roll. In this case, it was used to ensure the correct player was being used by giving each player a number.

Array:

As for the dice rolls themselves, an array was the most apt as it is easy to implement in this case and efficiency is not important for five values at most. An array was also used to store the number on the dice if a two-of-a-kind was found but did not reach three-of-a-kind, this was later going to be used to allow the player to choose which two-of-a-kind they wanted to use for the reroll but was never used to its full extent in the end

List:

Used to keep the history of the player's dice rolls and scores. This was used as the quantity of data to be put into whatever data structure was chosen wasn't known from the outset, this meant an array was harder to use and go through and so a list was used for simplicity.

Testing and Future Improvements:

When testing the code using the black box method, the executable was given to a family member who was told to run the code to completion and try black box testing. As well as testing the code myself it was also given to a person on the computer science course and both of us attempted to enter in values they believed might cause issues for white box testing.

Black Box Testing:

Test Number	Purpose of test	Test Values	Expected Result	Actual Results	Corrections
1	Options for game type	GameType = "Five"	While loop will ensure that user has to enter	As Expected	None
2	User ability to choose names	Name1 = "Terrabithia", Name2 = "No"	Both Values will go through correctly and "No" will give player 2 Name2 = "Player 2"	AS Expected	None

3	Choice of double points or not	DoubPoints = "Whoops"	Player will be given no points	As Expected	None
4	Choice of double points or not	DoubPoints = "Yes"	Player will be given double points	Gave no points for not saying yes or no	Added to the if statements to make sure "Yes" is acceptable

White Box Testing:

Test Number	**Purpose of test**	**Test Values**	**Expected Result**	**Actual Results**	**Corrections**
1	Checking that the matches in first position were counted correctly	DiceRoll[0], DiceRoll[1] and DiceRoll[2] = 1	Player will receive 3 points (As Double points weren't selected)	As Expected	None
2	Checking that the matches in last position were counted correctly	DiceRoll[2], DiceRoll[3] and DiceRoll[4] = 6	Player will receive 3 points (As Double points weren't selected)	Player received 0 points	Edited so that the NoOfAKind counted the matches correctly
3	Further check of gaining specific matches in the last position	DiceRoll[3] and DiceRoll[4] = 6	Player will Reroll and 6 will be stored to be used for the reroll	In the place of the two 6 values were two 0s instead (Not stored in TwoMatches)	An extra if statement was used to put the end values into TwoMatches

Possible Improvements:

There are several ways I would improve the code if more time was available. This includes the implementation of a wiping of the list containing the dice roll history as it gets too big for the screen, which was attempted during the time given but failed. Another possible future improvement could be allowing the player to choose their own parameters for a custom game, this could include number of dice, how many sides they have and how many people they are playing against.

Video URL:

https://youtu.be/5x7Q7CoGL9Q

References:

Chris Kohlhardt. 2005. *Gliffy*. Available at: https://www.gliffy.com/. [Accessed 2 March 2017]

Sam Allen. 2007. *DotNetPerls*. Available at: https://www.dotnetperls.com/. [Accessed 3 May 2017]