Siana S. Kong

Mr. Petti

ICS3U

1 June 2018

ICS3U Culminating Proposal: Bingo

To present my knowledge of coding that I have gained through ICS3U course at the fullest, I have decided to make a game that is based on the basic idea of Bingo. The user will be versing the computer, in order to have an opponent to win against. Although computers function much faster than human reaction time, Bingo is a completely fair game because it doesn’t require any logical skills. To win the game, a player has to get 2 Bingos before the other.

In this game, Both the user and the computer will be given a Bingo chart with randomly generated numbers. However, they will be created in a variable with an initial value of zero, so that the numbers aren’t the same for every game. Because the game requires to show a ‘chart’ with numbers in rows and columns, these randomly generated numbers should be arranged with 2d array method and printf. This will allow the program to recognize the location of the number using more than one set of numbers, just like finding a coordinate on a graph with both of its x and y values. To generate random values in general, the Random class will be imported, along with the Scanner class. There is not much use for the Scanner class. It will be used in the beginning of the game when the game asks the user their name, and by the end of each game to see if the user wants to play another game or not. All the numbers on the chart will be created within a same range of numbers to make the game much less time consuming; there are much less chances of gaining a number that is on each player’s chart-each player refers to the user and the computer playing the game- when the range is larger, especially when all the numbers are generated randomly. Also, because all the numbers are generated randomly within a same range, there could be an overlap of numbers on a Bingo chart. To prevent this, all the numbers on the chart will be stored in an ArrayList, which is separate from its original 2d arrays set. A for loop will be used to illiterate through each value in the ArrayList and check if the new number that is being generated on the chart overlaps with any pre-generated numbers that are already on the chart. To make a decision to create a new number and go through the loop again or to stick with the current number, if statements will be written within the for loop. For example, if there is a same value of the newly generated number in the ArrayList, then generate a new value for this variable, or continue the loop without making any changes if there is no same value. Because the entire loop illiterates bunch of times and requires quite a lot of work, it could be written in a separate void type method and be called to the main method when needed.

The game will provide with one randomly generated number for each round, which would be the number that the user and the computer can use to eliminate their numbers on the chart if they match. These numbers will be programmed to be generated with values within the same range as the numbers so that the values will be at least relevant to each other even when they don’t match. Although these numbers are called one at a time and separately, they will be stored in ArrayList class, so that all the played numbers can be recalled by the end of the game. It will be also used to eliminate numbers so that a number that has already been generated once can’t be played again. To eliminate any same number that has been created in the chart, a for loop will be used to check and find any same number in the 2d array of each player; which is the list of values in each Bingo charts. The variables of the 2d array with a matching value with the randomly generated value will be declared to have a value of zero. ; the updated chart will be displayed each round. To function the game properly, all the randomly generated values should be programmed to have higher value than zero. This will help the program to recognize number of eliminated numbers in a row or a column, or even in a diagonal set, by checking which variables have turned to zero in each set for each round. This is definitely helpful when recognizing a Bingo. For example, if the the Bingo chart is 5 by 5, the program will check if the first five variable are either all zero or not to see if the first line of the chart is a Bingo or not. If it is, a certain boolean will turn true and the program will announce who’s got the first Bingo; there is a bit more to the output for the second Bingo, for the second one is the winning one. Setting the 2d array set with variables becomes highly useful when looking for diagonal Bingo. To check if there is a diagonal Bingo, the program can just look at the value of the variables in each diagonal set. However, all of this checking process requires to be written over and over again if using if statements since it is required for each rows, columns, and diagonals. A for loop will require much less coding, but a separate loop should be called when checking for diagonal Bingos; it will require a different increment value(increasing value depends on the size of the Bingo). Still, using a for loop seems much simpler than using decision structures for each rows, columns, and diagonals. Therefore the if statements can be used only if an unexpected error occurs with the for loop since it is the surest way. This block of code should be called after the Bingo chart has been created because all the variables in the 2d array are initialized to zero at the beginning of the program. If it is called before the creation of the charts and right after the variables have been initialized to zero, the game would end right away by having both of the players winning the game right away. Whenever a number in each chart is eliminated, a certain boolean will turn true and the program will announce the number that is eliminated along with who’s chart it was from. The boolean will stay false if there are no matching numbers. Setting the eliminated numbers as zeros will also help the user to visually recognize how far or close they are from having Bingo; it’s just like circling the eliminated numbers in offline Bingo.

When a user or the computer gets their first Bingo, the game will send out an encouraging line to the player that is losing to make the game look more alive. For example, if the user is losing, the program will say things like ‘Username, keep up! You can still do it!’. If it is the computer that wins the first Bingo, the game will give user a chance to gain a Bingo. It will ask the user to provide with three numbers on the chart that they want to eliminate, which will most likely to be the numbers that’ll give them a Bingo. The game will select one of the three numbers randomly and eliminate that number from the user’s chart. This can be done by creating three separate variables in an Arrays set, execute a random variable that has a range of three numbers(0, 1, 2). Then decide which number to be given as a freebie to the user by using the if statement. For example, if random variable gets 0, the user gets their first pick. There is no need for loop for this one since it’ll be only applied to the game just once or even never; unless the number of Bingos to win the game changes afterward.

At the end, when one of the players wins the game by gaining two Bingos first, the program will announce the winner, number of rounds played, and all the random numbers that were given each round and even the eliminated numbers on the winner’s Bingo chart. Again, all the random numbers that were generated would be stored in an ArrayList already, therefore displaying all those numbers that were provided for each round would be an easy work. A separate ArrayList is necessary for gathering eliminated numbers. The value of the these numbers should be stored in the ArrayList before setting it to zero. Otherwise, all the eliminated numbers will be shown as zeros, which are not the number that the variables used to hold at the time of getting eliminated. The number of rounds can be counted by putting a variable that increments by the end of the while loop. This while loop will contain all the codes that is needed for a single round, and keep on looping until the games gets the winner. By incrementing the number of rounds variable at the end of the loop, the number of rounds will be updated every round by current value of the variable plus one. Therefore at the end, the game can simply call the variable, and it will display how many rounds has been played, which is a same thing as how many times has the loop been repeated. This while loop will be controlled by a boolean, where it repeats until the game getting the winner becomes true. After everything has been announced, the program will ask the user if they want to play another game of Bingo. Because the entire game has to be executed again, all the codes will be written under a while loop, where it will loop itself only if the user types in ‘yes’ when they are asked if they want to play the game again. If statement won’t work by itself and it’ll need at least one while loop to function the desired processing. Although a if statement can bring an entire block of code again with one line of code if the codes were written in a different method, it can’t go through the codes more than once without a while loop. With the while loop, a single boolean can be putted in as a condition for the loop to run. For example, if the user says ‘yes’, the boolean will turn true and run the entire loop again, or exit the loop if the user says ‘no’.

Although there is nothing impossible to code, it would be hard to stick to the plans 100% completely if there are really complicated challenges, due to the limited amount of time given and knowledge. Some of the challenges that may occur are keeping up with the ArrayLists and eliminating overlapping values, for they’ll require a lot of loops, followed by lots of nested loops within them. But so far, it seems like the game would work out the way it should as long as the plans and the ideas that are represented in this proposal are followed accordingly. Small little details may be added throughout the creation of the game. Also, some changes may follow unintentionally, depending on the difficulty of the programming. Or some new things that may not have been covered in class could be added to make the game more exciting or function to its fullest.