Contents

Assignment 0 – Formative Task

Functionalities

1. The colour of the LED in the beak of each of the four finches should be set to a different colour (that is, red, green, blue, and yellow).
2. In the first round, the program should generate a colour randomly (red, green, blue, or yellow) and display it to the user by blinking (flashing on and off) or flickering the beak of the corresponding finch.
3. The user should attempt to repeat this colour by obstructing the corresponding finch.
4. The program should compare the user's input with the correct colour.
5. If the user input is correct, in the second round, the program should display a sequence of two colours: the first colour plus another randomly generated colour. In the third round, the program should display a sequence of three colours: the previous two colours plus another randomly generated colour. That is, after each round, the program should add one colour to the previous sequence.
6. The program increases the user's score by one, every time the user correctly repeats the sequence.
7. If the user input is not correct, the program should output (speak or print) 'Game Over' and the program terminates.
8. The user has a specific number of lives. That is, if the user makes an error, the program should not terminate, but it should deduct one life and allow the user to continue.
9. Use a fifth robot as an ON/OFF button. That is, the game is ON as long as the fifth finch is not with its beak up. The user can quit or 'switch off' the game by placing the fifth finch beak up.

Introduction

The Formative Task set. aimed for groups to ultimately work together, design and implement a computer program using Finch robots. The computer program would imitate the well-known game 'Simon Says', in which a group would use their robots to emulate the typical instructions given by Simon using their touch functions and light abilities. To achieve such an outcome, would mean the use of teamwork, planning and programming to ensure the implementation stage would be reached and fulfilled accordingly. This report will cover these bases, outlining and reflecting on the processes and experiences of teamwork and design.

Planning and management

To achieve a group task effectively and efficiently, naturally demands the use of teamwork. Therefore, the first task set upon our group was to elect a leader in order to have a straightforward direction and understanding of what aims and tasks parties should move toward. Also, just as important, finding a time in which all members would be happy to meet throughout the work week, discuss plans and track current progress beset upon each faculty of the group. This seemingly simple task later became a problem as there was consistent absence from group meetings, meaning there was not only a lack in group cohesion, but individual understanding. The lack of interaction meant that parties were not engaging with the project and therefore causing an imbalance and disturbance within the desired tasks set by the elected group leader. All in all, a lack of group cohesion caused a more stressful work environment and set too much pressure upon individuals due to the lack of equally distributed tasks and aims being worked on and being met. This could have been improved by forcing a more serious

discussion and stressing the importance of attendance or finding a new time that may have been more convenient for everyone.

Communication

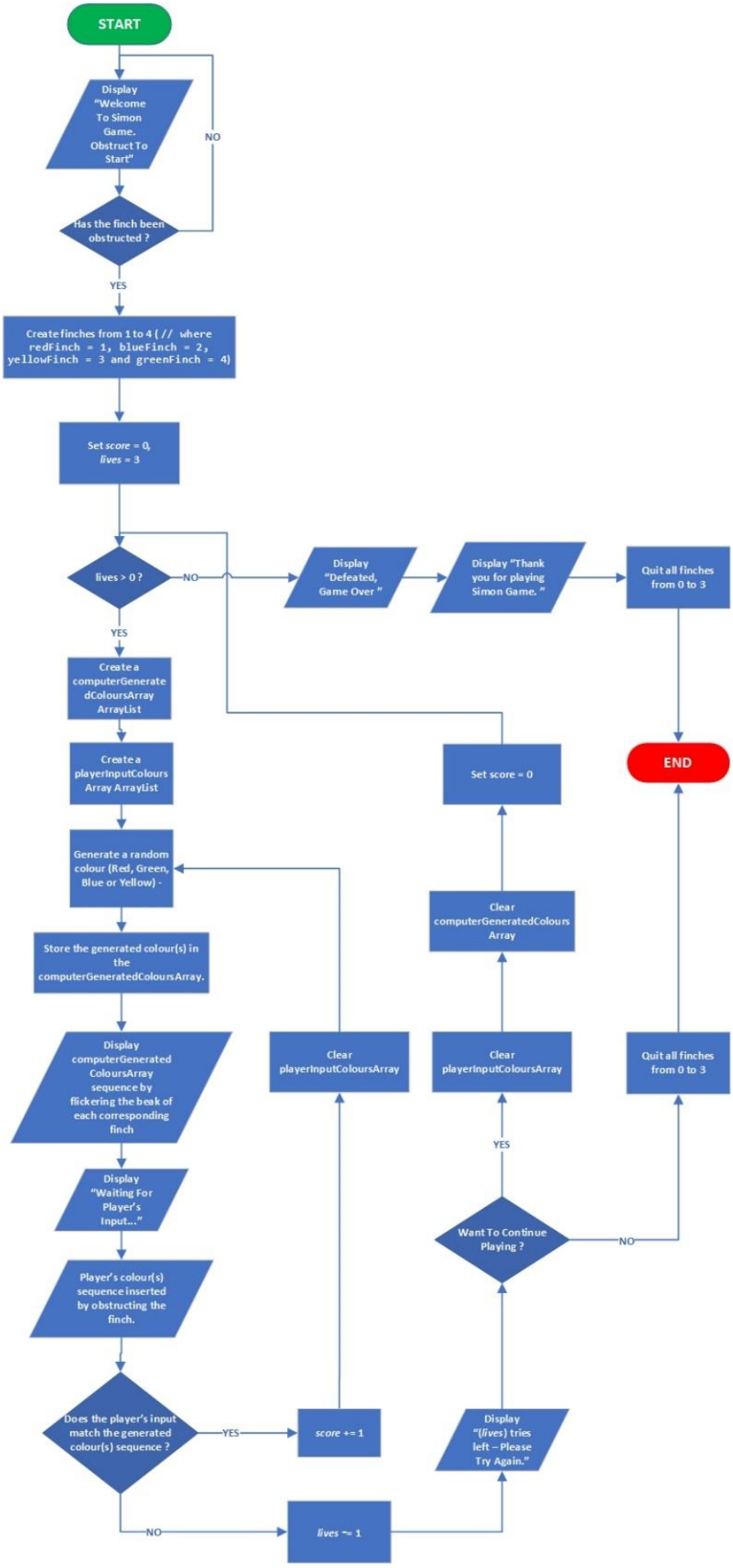
Throughout the process of this assignment, communication was initialised from the offset of this task. It was established that group meetings and social media apps like 'WhatsApp' would be used to communicate, discuss matters and plan future details. Also, the sharing of documents and files was achieved using applications like 'Google Drive' which allowed for everyone in the group to have access, add, edit or remove content from the files as they wished. This allowed for everyone to see the progress being made task wise and act accordingly. Although the chosen medium is not the problem itself, social media was not enough to consistently provide information as it relies on people checking messages manually or having their phone available. Using the cloud drives was used well as it allowed for files to be moved freely and easily with few potential manual interferences. Communication could have been improved by discussing in person, rather than online and emphasising the importance of checking and replying to group related messages.

Practical Work

Although, the ultimate goal of code implementation was set, it was not necessary. Nevertheless, the completion of all tasks was achieved, as all tasks were completed, and the final code was implemented. This was achieved by distributing necessary tasks amongst different members in the group, and then said members would check each other's progress and work the following week to ensure requirements were met. This was aided by group work and having the necessary resources from the University and information on the internet. Although, as it approached the code implementation stage, the group found that not everyone was equally knowledgeable in terms of programming and so naturally the programming beset itself on the parties most able.

Despite the lack of cohesion and attendance, the tasks were completed, and the final code was implemented, which was fully functioning. Achieving said tasks could have better been met if communication and planning was clearer throughout the group. Moreover, there were some technical setbacks on top of interpersonal issues, although Finch robots were working and in order, they were inconsistent in their reaction to the code. An instance of this was the touch function in the finch, it would not consistently read and register the input and therefore made implementing code tiresome as the process became more drawn out, reducing efficiency. This was overcome by working around the Finch's inconsistencies and instead implementing a hover function which was more consistent and allowed for workflow to be optimised and tasks to be met. We were also instructed to create flowcharts and pseudocode, the process for both of these was similar as the workload was distributed evenly throughout the group. Although some problems arose as the initial pseudocode was too language specific and the flowchart was also too complicated for the average user to read. This was overcome by reviewing both pieces of work and amending what we felt was appropriate. This was done by reducing the complexity of the diagrams and ensuring that they met the specification.

Flowchart


START
Display "Welcome To Simon Game. Obstruct To Start"
Has the finch been obstructed ?
NO
YES
Create finches from 1 to 4 (// where redFinch = 1, blueFinch = 2, yellowFinch = 3 and greenFinch = 4)
Set score = 0, lives = 3
lives > 0 ?
NO
Display "Defeated, Game Over "
Display "Thank you for playing Simon Game. "
Quit all finches from 0 to 3
END
YES
Create a computerGenerate dColoursArray ArrayList
Create a playerInputColours Array ArrayList
Generate a random colour (Red, Green, Blue or Yellow) -
Store the generated colour(s) in the computerGeneratedColoursArray.
Display computerGenerated ColoursArray sequence by flickering the beak of each corresponding finch
Display "Waiting For Player's Input..."
Player's colour(s) sequence inserted by obstructing the finch.
Does the player's input match the generated colour(s) sequence ?
YES
score += 1
Clear playerInputColoursArray
NO
lives -= 1
Display "(lives) tries left – Please Try Again."
Clear playerInputColoursArray
Clear computerGeneratedColours Array
Set score = 0
YES
Want To Continue Playing ?
NO
Quit all finches from 0 to 3

Pseudocode

Algorithm 1: Simon's Game (redFinch, blueFinch, greenFinch, yellowFinch, score, lives, gameIsOn)

Input: redFinch, blueFinch, greenfinch and yellowFinch are Finches. Score and lives are positive integers. gameIsOn is a Boolean value. The players colour input (by obstructing the finch beak) once the generated colour sequence is displayed. Integer value if the player wants to continue playing.

```
Set gameIsOn = false
Set score = 0
Set lives = 3
Set NumFinches = input "number of finches the player have: "
Create array[NumFinches] finches

/*Create all the finches by using a for loop to load them into an array, where
redFinch = 1, blueFinch = 2, yellowFinch = 3 and greenFinch = 4*/

For i=1 to NumFinches
      Create new finch
      Load finch into an finches
      Increment i
End for
Set finches[1] = redFinch
Set finches[2] = blueFinch
Set finches[3] = yellowFinch
Set finches[4] = greenfinch

/*Once incrementation is complete 4 different finches will be available to
allow for the 4 colours 'Simon says' demands to play the game once all finches
are initialised.*/

Finch setLED to colour RED
Repeat
      Let finch wait for 300ms
      Print "Looks like something went wrong..."
      Finch buzz
Until finch is obstructed

Set gameIsOn = true
Create an ArrayList called computerGeneratedColoursArray
Create an ArrayList called playerInputColoursArray
Print "Welcome To Simon Game. Tap To Start"

/*Arrays are created in order to allow for the game to be played,
"computerGeneratedColoursArray" allows for colours to be stored and displayed
```

```
later as you play the game. The ”playerInputColoursArray” enables the user to
input what colour they remember Simon telling them.*/

While gameIsOn is true
      If lives > 0
            Generate a random colour (Red, Green, Blue, Yellow)
            Store the generated colour in the computerGeneratedColoursArray
            For i=0 to size(computerGeneratedColoursArray) //length of ArrayList
                  Display computerGeneratedColoursArray[i] //displays the colour
            on the corresponding finch
                  Increment i
            End For
            For i=0 to size(computerGeneratedColoursArray)
                  For j=0 to NumFinches
                        If finches[i] = obstructed then
                              playerInputColoursArray add(finches[i])
                        End If
                        Increment j
                  End For
                  Increment i
            End For

            If computerGeneratedColoursArray = playerInputColoursArray then
                  Score = score + 1
                  Print “Correct – Score:” + score
                  Clear playerInputColoursArray
            End If

/*If the colour Simon has ordered to be repeated is remembered and executed by
the user they are rewarded with a “+1” score allowing them to continue to play.
*/

            Else
                  lives = lives - 1
                  Print “Game Over! Remaining Lives:” + lives
                  Print “Final Score:” + score
                  Set wantToPlayAgain = input “Want To Play Again? 1 = Yes / 2 =
      No: ”

/*If the colour Simon has ordered to be repeated is not remembered and is
unsuccessfully executed by the user they are punished with a “-1” score until
they lose all their lives and are asked to “play again”. */

                  If wantToPlayAgain = 1 then
                        Clear computerGeneratedColoursArray
                        Clear playerInputColoursArray
                        Set score = 0
```

```
                Else
                      Print “Thank you for playing, see you soon !!!”
                      Set gameIsOn = false
                      For i = 1 to NumFinches
                            Quit finch
                            Increment i
                End If
          End If
```

```
/*Once user has lost all of their lives they are asked if they would like to
play again, in which case the score is cleared and the game loops back to its
initial state. Otherwise, the game realises the users desire to not play again
and sets itself off, quits the program and displays a “Thank you” message”. */
```

```
      Else
            Print “Defeated, Thank You For Playing, See You Soon !!!”
            Set gameIsOn = false
      End if
End While
```

Output: computerGeneratedColoursArray by flickering the beak of each corresponding finch, warning sound, score and number of lives remaining.

Code Implementation

```
import edu.cmu.ri.createlab.terk.robot.finch.Finch;
import java.awt.*;
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

class SimonGame {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        System.out.print("How many finches do you have: ");
        int howManyFinches = input.nextInt();
        Finch[] finches = new Finch[howManyFinches];

        // Function to create finches.
        for (int i = 1; i <= finches.length; i++) finches[i] = new Finch();

        boolean gameIsOn;
        int score = 0;
        int lives = 3;

        // Creates String ArrayLists.
        ArrayList<String> computerGeneratedColoursArray = new ArrayList<>();
        ArrayList<String> playerInputColoursArray = new ArrayList<>();

        System.out.println("\n\n          Welcome To Simon Game. \nPlace Your Hand In front Of The Finch To Start.");

        // Function to check if the finch is blocked to start the game. If there is no movement then it will display the
error message in every 3 secs.
        if (!finches[1].isObstacle()) {
            int i = 0;
```

```
        do {
            i++;
            finches[1].sleep(300);
            //System.out.println(i);
            if (i == 10) {
                System.out.println("     Looks like something went wrong... \n           Please try again.");
                finches[1].buzz(2500, 1000);
                i = 0;
            }
            finches[1].setLED(Color.RED, 1000);
        }
        while (!finches[1].isObstacle());
    }

    gameIsOn = true;
    System.out.println("           Game Starting.");

    while (gameIsOn) {

        if (lives > 0) {

            // It generates a random colour and adds it to the computerGeneratedColoursArray.
            computerGeneratedColoursArray.add(getRandomColour());

            //  *Debugging - disconsider this comment* System.out.println(computerGeneratedColoursArray);

            // Function to read each index in the array and flicker each corresponding colour on the allocated finch.
            for (int j = 0; j < computerGeneratedColoursArray.size(); ++j) {
                String currentIndexColour = computerGeneratedColoursArray.get(j);
                INNER_LOOP:
                if ("RED".equals(currentIndexColour)) {
                    finches[1].setLED(Color.RED, 1500);
                    finches[1].sleep(500);
                    break INNER_LOOP;
                } else if ("BLUE".equals(currentIndexColour)) {
```

```
                        finches[2].setLED(Color.BLUE, 1500);
                        finches[2].sleep(500);
                        break INNER_LOOP;
                    } else if ("YELLOW".equals(currentIndexColour))  {
                        finches[3].setLED(Color.YELLOW, 1500);
                        finches[3].sleep(500);
                        break INNER_LOOP;
                    } else {
                        finches[4].setLED(Color.GREEN, 1500);
                        finches[4].sleep(500);
                        break INNER_LOOP;
                    }
                }

                System.out.println("       Players Turn.......");

                // Function to take players' input by checking which finch is blocked and then adding colour to the
playerInputColoursArray.
                if (!finches[1].isObstacle() || !finches[2].isObstacle() || !finches[3].isObstacle() ||
!finches[4].isObstacle()) {
                    for (int j = 0; j < computerGeneratedColoursArray.size(); ++j) {
                        INNER_LOOP2:
                        while (true) {
                            if (finches[1].isObstacle()) {
                                finches[1].setLED(Color.RED, 1000);
                                playerInputColoursArray.add("RED");
                                finches[1].sleep(500);
                                break INNER_LOOP2;
                            } else if (finches[2].isObstacle()) {
                                finches[2].setLED(Color.BLUE, 1000);
                                playerInputColoursArray.add("BLUE");
                                finches[2].sleep(500);
                                break INNER_LOOP2;
                            } else if (finches[3].isObstacle()) {
                                finches[3].setLED(Color.YELLOW, 1000);
```

```
                                playerInputColoursArray.add("YELLOW");
                                finches[3].sleep(500);
                                break INNER_LOOP2;
                            } else if (finches[4].isObstacle()) {
                                finches[4].setLED(Color.GREEN, 1000);
                                playerInputColoursArray.add("GREEN");
                                finches[4].sleep(500);
                                break INNER_LOOP2;
                            }
                        }
                    }
                    // *Debugging - disconsider this comment* System.out.println(playerInputColoursArray);
                }

                // Check if the computerGeneratedColoursArrays matches the playerInputColoursArray.
                boolean checkingArrays = computerGeneratedColoursArray.equals(playerInputColoursArray); //returns true
because lists are equal

                // If the players' input is correct, then it increments the score and continues.
                if (checkingArrays == true) {
                    score++;
                    System.out.print("Correct - Score : " + score + "\n");
                    playerInputColoursArray.clear();
                }

                // If the input is incorrect, decrements live by 1 and clears both the arrays.
                else {
                    lives--;
                    System.out.println("Game Over");
                    System.out.print("Final Score : " + score + "\n");
                    System.out.println("Remaining Lives : " + lives + "\nTry Again");

                    // It ask if the players want to continue playing further, else exits the game.
                    Scanner input2 = new Scanner(System.in);
                    System.out.print("Want To Play Again? (1 = Yes / 2 = No) :  ");
```

```
                    int wantToPlayAgain = input2.nextInt();

                    if (wantToPlayAgain == 1) {
                        computerGeneratedColoursArray.clear();
                        playerInputColoursArray.clear();
                        score = 0;
                        gameIsOn = true;
                    } else {
                        System.out.println("Thank You For Playing, See You Soon !!!");
                        gameIsOn = false;
                        for (int i = 1; i <= finches.length; i++) finches[i].quit();
                        System.exit(0);
                    }
                }
            }
            // Exits the game if not enough attempt left (lives = 0).
            else {
                System.out.println("Defeated \nThank You For Playing, See You Soon !!!");
                gameIsOn = false;
            }
        }
        // Function quits all the finch created that were created at the start and exits.
        for (int i = 1; i <= finches.length; i++) finches[i].quit();
        System.exit(0);
    }

    //Function to generate a random colour.
    private static String getRandomColour() {
        Random rand = new Random();
        String[] colour = {"RED", "YELLOW", "GREEN", "BLUE"};
        return colour[rand.nextInt(4)];
    }
}
```