CS1701 Group Project Lectures & Tutorials - Assignment CS1810 Software Implementation (Individual Submission) 2023/4

Task 1 : Zigzag
Name : Tarif Aziz
Student ID : 2364202
Tutor : Ferdoos Hossein Nezhad
Group: A16

Table of Contents

Implementation Summary:

Required Functionalities :

No.	Functionality	Status	Notes
1	User scans a code to receive zigzag number and zigzag length	Passed	Successfully Scans a qr with 2 strings. First one is Zigzag length and the second one is zigzag number
2	Validation : Zigzag length between 15 to 85 cm	Passed	Error Handling is done correctly where it will accept only values between 15 to 85 (in cm)
3	Validation : Zigzag Number at least 2 to 12 and even	Passed	Error Handling is done correctly where it will accept only values between 2 to 12 (all even)
4	Generating random wheel speed	Passed	Generated a random wheel speed between 40 to 100 and the velocity of the mentioned speed is not accurate. For say when the speed is set to 100 for 5 seconds the velocity is 32cm/s and when set to 90 for 5 seconds its travelling 32cm/s (it should not be same) for 80 it should less but its 31cm/s just decreasing 1cm/s So for accurate result, I took 7 possible values which are multiples of 10 upto 100 and starts at 30 as the bot does not move until 30 so I took 7 possible cases and generated random speed as discussed with module leader (Yasooda) in the lab.
5	Set first Colour to Green for first Zigzag section	Passed	As mentioned in the Functional requirement it sets the colour to green at first followed by the second colour
6	Record Start time	Passed	Starts recording time as soon as it starts travelling with random generated speed
8	Move Forward with Specified Zigzag Length	Passed	It moves forward with random generated speed for a specific time to cover the zigzag length received from the qr code scanning
9	Stop bot for 1 second after each zigzag length	Passed	Makes the bot stop for 1 second and takes a turn

10	Take a 90 degree turn	Passed	
11	Set Second Colour to Blue for second Zigzag section	Passed	As mentioned in the Functional requirement it sets the colour to Blue followed by the first colour (Green)
12	Retrace the same movement after specific Zigzag Number with same zigzag length and colour	Passed	Retraces back the same path with specific colours
13	Record Stop time	Passed	Stops the timer and records it
14	Compute Duration of a single journey	Passed	It computes the duration by a formula
15	Store zigzag number, zigzag length, random wheel speed, duration of zigzag journeys,Traversed path in a straight line and Distance travelled for a single journey in a text file	Passed	Stores all the data to a text file when Button X is pressed on the bot
16	Press Button “X”on Bot to terminate	Passed	It waits for 5 seconds if the user does not press X button on the bot it turns and goes back to loop to complete another journey
17	Display Zigzag journey numbers before Terminating	Passed	It displays the total number of Zigzag journeys completed as it goes into a loop until X is pressed on the bot . So the user gets to see the total number of zigzag journeys completed
18	Goes to a loop for another journey after taking a turn	Passed	After completing a journey if button X is not pressed on the swiftbot then it should go do the zigzag journey again. It goes

Additional Functionalities:

No.	Functionality	Status	Notes
01	Customizable colours (Red Green Blue Yellow)	Passed	User gets to pick colours for the zigzag sections. It will ask the user to pick first and second colour for the journey
02	SquareLoop Pattern	Passed	Included a new pattern where the bot creates a square. User gets to pick between a single square and double square
03	Energy Efficiency Mode	Passed	This mode will disable all the underlights to save energy. So the zigzag journey/ squareloop journey will run without underlights
04	Capture a photo during Journey	Failed	Spent too much time on fixing the wheels thus could not implement this feature

Changes to Algorithm and Design:

No.	Algorithm Design (Before)	Algorithm Design (After)	Reason
1	Random speed generation = 1 to 100	Random speed generation = Multiples of 10 starting from 40 to 100	• The bot does not move until it reaches a speed of 30, which itself is very inaccurate, making it nearly impossible to use for calculating journeys. Additionally, the swiftbot's wheels are inaccurate, so conducting 60 to 70 cases without any pattern would have been very time-consuming.
2	Added new variables	Boolean mode Boolean squareloopcheck Int wheelSpeed	• Boolean mode to check if its Normal or Energy Efficiency mode • Boolean square loopcheck to check if its a square loop or not if its a squareloop then it will not go in a loop in other words it will not retrace • Combined rightWheelSpeed and leftWheelSpeed into a single variable
3	rightWheelSpeed leftWheelSpeed	Variables removed	• Combined it to a single variable
4	Take a photo during the journey	Could not implement it	• Did not have enough time to implement the Photo capture additional functionality

5	Not Displaying RandomWheel Speed	Displaying RandomWheel speed to the user	• User can know the randomly generated speed

Testing:

Functionality	Input	Output(Expected)	Output(Observed)	Test Result
Start	Any key	Start the Program	Starts the Program	Passed
Scan QR	55, 02	zigLength= 55 zigNumber = 02	zigLength= 55 zigNumber = 02	Passed
Scan QR	12,02	Error Message	Error Message	Passed
Scan QR	15, 01	Error Message	Error Message	Passed
Scan QR	Empty QR	Error “Empty QR”	Error “Empty QR”	Passed
Scan QR	No QR	Error: QR not Found	Error: QR not Found	Passed
Random wheel speed	N/A	Randomly generating 40 to 100(multiples of 10 only)	Randomly generating 40 to 100(multiples of 10 only)	Passed
Random wheel speed	N/A	Complete journeys with randomly generated wheel speed	Complete journeys with randomly generated wheel speed	Passed
Mode Selection	1	Run Normal Mode	Running Normal Mode	Passed
Mode Selection	2	Run Energy Efficiency Mode	Running Energy Efficiency Mode	Passed
Mode Selection	3, a , A	Show Mode Selection Message again	Show Mode Selection Message again	Passed
Pattern Selection	1	Run Zigzag Pattern	Running Zigzag Pattern	Passed
Pattern Selection	2	Run Squareloop Pattern	Running Squareloop Pattern	Passed
Pattern Selection	3, a , A	Show Pattern Selection Message again	Show Pattern Selection Message again	Passed

Zigzag	55, 02	Travel 55 cm with number of section zigzag 2	Travel 55 cm and Number of zigzag section 2	Passed
Zigzag Retracing	For 55, 02	Retrace back 55cm length and 2 zigzag sections	Retrace back 55cm length and 2 zigzag sections	Passed
Zigzag	55, 04	Travel 55 cm with number of section zigzag 4	Travel 55 cm and Number of zigzag section 4	Passed
Zigzag Retracing	For 55, 04	Retrace back 55cm length and 4 zigzag sections	Retrace back 55cm length and 2 zigzag sections	Passed
Squareloop	55, 04	Length of a side of square= 55cm	Length of a side of square= 55cm	Passed
Squareloop	N/A	Prints a Squareloop completed after completion	Prints a Squareloop completed after completion	Passed
Squareloop(Loop Selection)	1	Create 1 square	Create 1 square	Passed
Squareloop(Loop Selection)	2	Create 2 squares	Create 2 squares	Passed
Zigzag(Normal Mode)	N/A	Let's user pick colours for first section and second section	Let's user pick colours for first section and second section	Passed
SquareLoop(Normal Mode)	N/A	Let's user pick colours for first section and second section	Let's user pick colours for first section and second section	Passed
Zigzag(Energy Efficiency Mode)	N/A	Run Zigzag disabling all the underlights	Run Zigzag disabling all the underlights	Passed
SquareLoop(Energy Efficiency Mode)	N/A	Run SquareLoop disabling all the underlights	Run SquareLoop disabling all the underlights	Passed
Colour Selection	1	Select Red	Selects Red	Passed
Colour Selection	2	Select Green	Selects Green	Passed
Colour Selection	3	Select Blue	Selects Blue	Passed
Colour Selection	4	Select Yellow	Selects Yellow	Passed
Colour Selection	5, A, a	Show Colour Selection again	Shows Colour Selection again	Passed
Travel Zigzag with colours	N/A	Travel Zigzag with Selected colours	Travel Zigzag with Selected colours	Passed

Travel Squareloop with colours	N/A	Travel Squareloop with Selected colours	Travel Squareloop with Selected colours	Passed
Termination	N/A	Write Zigzag Length,Zigzag Number, Random wheel speed, Traversed path length, duration and Distance travelled in a straight line to a text file in raspberry pi when button X is pressed on the bot	Write Zigzag Length,Zigzag Number, Random wheel speed, Traversed path length, duration and Distance travelled in a straight line to a text file in raspberry pi when button X is pressed on the bot	Passed
Button X not pressed (Normal mode)	N/A	Should take a turn again and ask for colours	Takes a turn again and asks the user for colours	Passed
Button X not pressed (Energy Efficiency Mode)	N/A	Should take a turn again and retrace	takes a turn again and retraces	Passed

Planning and Monitoring:

Start Date	Task to Complete	Task Completed	Problems Encountered	Solution	End Date
21/01/24	QR Scan	QR Scan	None	N/A	27/01/24
28/01/24	Qr Scan Error Error handling	Qr Scan Error Error handling	QR accepting non integer values	a condition to accept integers only added	3/02/24
4/02/24	Develop the Zigzag Pattern method	Developed the Zigzag Pattern method	WheelSpeed inaccurate	One wheel was was moving faster than the other so deducted a certain value to balance out	7/02/24
11/02/24	Develop the Mode Selection method	Developed the Mode Selection method	None	N/A	13/02/24
14/02/24	Develop Pattern Selection	Developed Pattern Selection	None	N/A	17/02/24
14/02/24	Develop Energy Efficiency Mode	Developed Energy Efficiency Mode	None	N/A	17/02/24
14/02/24	Develop Colour Selection for Journeys	Developed Colour Selection for Journeys	None	N/A	17/02/24
17/02/24	Develop Retrace Journey	Developed Retrace Journey	Say for a speed of 70, I need to subtract a certain amount 'x'. For a speed of 80, the amount to subtract should be 'y'. And for a speed of 90, the deduction should be 'z'. Yes, each speed requires a new condition. I've even attempted to directly assign these values, but the issue arises because the specific variable that corresponds to each speed fluctuates as the	Issue with bot (No Solution)	19/02/24

			bot operates over an extended time period during my tests.		
19/02/24	Add User Instruction	Added User Instruction	None	N/A	21/02/24
19/02/24	Develop RandomSpeed Generator	Developed RandomSpeed Generator	None	N/A	21/02/24
19/02/24	Develop Mechanism for the zigzag journey to go in a loop if X is not pressed	Developed Mechanism for the zigzag journey to go in a loop if X is not pressed	None	N/A	21/02/24
22/02/24	Termination Log File Storing	Developed method for Termination and store Log File	Could not find the text file	Added escape sequence to deal the problem when defining directory	24/02/24
22/02/24	Add AdjustedWheelSpeed Method	Added AdjustedWheelSpeed Method	None	N/A	24/02/24
22/02/24	Separate Classes	Implemented Separate Classes with separate methods	Could not access certain variables	Learned access modifiers and implemented properly	24/02/24
23/02/24	Add Additional Conditions for Zigzag and Squareloop(not to run retracing if Squareloop is selected)	Added Additional Conditions for Zigzag and Squareloop(not to run retracing if Squareloop is selected)	Unknown methods were running for different conditions	Fixed the conditions added validation(boolean) checker to identify certain methods	27/02/24
23/02/24	Testing	Testing	Found a lot of bugs	Fixed all the bugs	27/02/24
23/02/24	Error Handling	Error Handling	Some parts of the code was not handling error properly	Fixed error handling	27/02/24

Source Code

```
import swiftbot.*;
import java.util.Scanner;
public class Movement {

  /*
   * Public means it can be accessed by any class
   * static means that a new creation of an instance is not
   *
   * No private variables as all of the variables are used in different classes and hence set to
static
   */
  static int zigLength;
  static int zigNumber;
  static int wheelSpeed;
  static int Calc;
  public static SwiftBotAPI swiftbot;
  static long duration = 0;
  static final Scanner scanner = new Scanner(System.in);
  static int zigzagJourney = 0;
  static boolean mode = false;
  static boolean squareloopcheck = false;

  public static void main(String[] args) {
      swiftbot = new SwiftBotAPI();

      System.out.println("Press any key to start...");
      /*
       * Abstraction: My methods (modeSelection, patternSelection, zigZag, squareLoop, etc.)
       * abstract away the complexities of operations. Users of my class don't need to know the
inner workings of
       * these methods to use them, they just need to know what they do. This is a form of
Abstraction as
       * complex operations are hidden from the user and makes it easy to understand
       */
      scanner.nextLine();
      userInstructions();  // calling user instructions method
      QrCodeHandling.qrScan(swiftbot); //calling qrScan method from QrCodeHandling Class

      modeSelection();   //calling modeSelection method
```

```
      ProgramTermination.termination(swiftbot); // calling termination method from
ProgramTermination class
   }

   /*
    * This method is for mode selection and asks the user to select a mode if
    * invalid mode is selected say 3 or 4 it will run the same method again so Error handling is
done properly
    */
   public static void modeSelection() {
      while (true) { // loop until a valid input is provided
         System.out.println("Select Mode:");
         System.out.println("Press 1: Normal Mode");
         System.out.println("Press 2: Energy Efficiency Mode");
         String ModeSelection = scanner.nextLine(); // asks the user for an input for Mode
Selection and stores it in a string
         switch (ModeSelection) {
            case "1":
               normalMode(); // here the switch takes userInput and if its 1 then it will run Normal
Mode
               return; // returning
            case "2":
               energyEfficiencyMode(); // here the switch takes userInput and if its 1 then it will run
energyEfficiencyMode Mode
               return; // returning
         }
      }
   }

   /*
    * This method is for Pattern selection and asks the user to select a pattern if
    * invalid pattern is selected say 3 or 4 it will run the same method again so Error handling is
done properly
    */
   public static void patternSelection() {
      while (true) { // Loop until a valid input is provided
         System.out.println("Select Pattern:");
         System.out.println("Press 1: ZigZag");
         System.out.println("Press 2: SquareLoop");

         String patternSelection = scanner.nextLine(); // asks the user for an input for Pattern
Selection and stores it in a string
         switch (patternSelection) {
```

```
            case "1":
                zigZag();
                return; //  exiting after a valid input
            case "2":
                squareLoop();
                return; //  exiting after a valid input
        }
    }
}

/*
 * This method is for the squareLoop pattern and it will run only if it is called as after mode selection
 * user gets to choose the pattern and
 * if squareLoop is selected it will run this part and the swiftbot will create a squareLoop pattern
 *
 */
private static void squareLoop() {
    squareloopcheck = true; // This boolean checks if the swiftbot is running squareLoop and will set it to true
                            // and if its true then it will not retrace and will terminate after a single or double loop
    System.out.println("SquareLoop selected");
    System.out.println("Select Number of Loops");
    System.out.println("Press 1: Single Loop");
    System.out.println("Press 2: Double Loop");
    /*
     * This is a condition set to identify if its Normal Mode or Energy Efficiency mode if its set to true
     * then it will ask the user for colours and if its set to false it wont ask for colours
     */
    if (mode == true) {
        int noLoop = scanner.nextInt(); // takes input from the user, valid inputs are 1 and 2
        SetColours.setfirstColour(); // method for setting first colour
        SetColours.setSecondColour(); // method for setting Second colour
        // switch condition is 1 and 2 if its 1 then SquareLoop will run once and if its 2 then it will run twice
        switch (noLoop) {
            case 1:
                for (int i = 1; i <= 2; i++)// condition to run half of square twice
                {
                    swiftbot.fillUnderlights(SetColours.selectedColor); // sets first colour
```

```
                adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength); // moves forward with
the adjusted wheel speed
                swiftbot.move(0, 0, 1000); // stops for 1 second
                swiftbot.move(-100 , 100, 400); // left turn
                swiftbot.fillUnderlights(SetColours.selectedColor2);// sets Second colour
                adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength); // moves forward with
the adjusted wheel speed
                swiftbot.move(0, 0, 1000);// stops for 1 second
                swiftbot.move(-100 , 100, 400); // left turn
            }
            break;
        case 2:
            for (int i = 1; i <= 4; i++)  // condition to run half of square 4 times
            {
                swiftbot.fillUnderlights(SetColours.selectedColor);// sets first colour
                adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength);// moves forward with
the adjusted wheel speed
                swiftbot.move(0, 0, 1000);// stops for 1 second
                swiftbot.move(-100 , 100, 400); // left turn
                swiftbot.fillUnderlights(SetColours.selectedColor2);// sets Second colour
                adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength); // moves forward with
the adjusted wheel speed
                swiftbot.move(0, 0, 1000);// stops for 1 second
                swiftbot.move(-100 , 100, 400); // left turn
            }
            break;
    }
}
// else run the same thing without colours
else {
    int noLoop = scanner.nextInt();
    switch (noLoop) {
        case 1:
            for (int i = 1; i <= 2; i++)// condition to run half of square twice
            {
                adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength);// moves forward with
the adjusted wheel speed
                swiftbot.move(0, 0, 1000);// stops for 1 second
                swiftbot.move(-100 , 100, 400); // left turn
                adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength);// moves forward with
the adjusted wheel speed
                swiftbot.move(0, 0, 1000);// stops for 1 second
                swiftbot.move(-100 , 100, 400); // left turn
            }
```

```
                break;
            case 2:
                for (int i = 1; i <= 4; i++) // condition to run half of square 4 times
                {
                    adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength);// moves forward with
the adjusted wheel speed
                    swiftbot.move(0, 0, 1000);// stops for 1 second
                    swiftbot.move(-100 , 100, 400); // left turn
                    adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength);// moves forward with
the adjusted wheel speed
                    swiftbot.move(0, 0, 1000);// stops for 1 second
                    swiftbot.move(-100 , 100, 400); // left turn
                }
                break;
        }
    }
  }

  static void zigZag() {

    long StartTime = 0;
    if (mode == true && squareloopcheck==false) // Here the condition checks if its Normal
mode or not followed by squareLoop check
    {
        System.out.println("Zigzag Selected");
        SetColours.setfirstColour();              // asks the user to input the first colour
        SetColours.setSecondColour();                                    // asks the user to input the
Second colour
        for (int i = 1; i <= Calc; i++) {
         System.out.println("Clocking current time...");
            StartTime = System.currentTimeMillis();  // records start time
            swiftbot.fillUnderlights(SetColours.selectedColor);    // sets first colour
            adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength);   // moves forward with the
adjusted wheel speed
            swiftbot.move(0, 0, 1000); // pause for 1 second
            swiftbot.move(-100 , 100, 400); // left turn
            swiftbot.fillUnderlights(SetColours.selectedColor2); // sets Second colour
            adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength);// moves forward with the
adjusted wheel speed
            swiftbot.move(0, 0, 1000); // pause for 1 second
            swiftbot.move(100 , -100, 400); // right turn
        }
        Reverse.retrace(swiftbot);  // calling retrace method from Reverse class to retrace
journey
```

```
        swiftbot.disableUnderlights(); // disabling all underLights
        long StopTime = System.currentTimeMillis();  //after retracing it should Stop the timer and
calculate duration
        duration = duration + (StopTime - StartTime); //calculating duration
        System.out.println("Duration: " + duration / 1000 + " seconds");
        zigzagJourney = zigzagJourney + 1; // adding +1 to zigzagJourney after a complete
journey
    } else {
        for (int i = 1; i <= Calc; i++) {
            StartTime = System.currentTimeMillis(); // records start time
            adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength);// moves forward with the
adjusted wheel speed
            swiftbot.move(0, 0, 1000); // pause for 1 second
            swiftbot.move(-100 , 100, 400); // left turn
            adjustMoveBasedOnWheelSpeed(wheelSpeed, zigLength);// moves forward with the
adjusted wheel speed
            swiftbot.move(0, 0, 1000); // pause for 1 second
            swiftbot.move(100 , -100, 400); // right turn
        }
        Reverse.retrace(swiftbot);
        swiftbot.disableUnderlights();// disabling all underLights
        long StopTime = System.currentTimeMillis();//after retracing it should Stop the timer and
calculate duration
        duration = duration + (StopTime - StartTime);//calculating duration
        System.out.println("Duration: " + duration / 1000 + " seconds");
        zigzagJourney = zigzagJourney + 1;// adding +1 to zigzagJourney after a complete
journey
    }
    }

  static void adjustMoveBasedOnWheelSpeed(int wheelSpeed, int zigLength) {  /* this method
helps the swiftbot to adjust the

                                        wheel speed as for each wheel speed the

                                        adjustment is different*/
    switch (wheelSpeed) {
        case 40:
         swiftbot.move(40-4, 40, 60 * zigLength);
            break;
        case 50:
         swiftbot.move(50 - 7, 50, 52 * zigLength);
            break;
        case 60:
```

```
            swiftbot.move(60 - 9, 60, 47 * zigLength);
              break;
            case 70:
             swiftbot.move(70 - 11, 70, 45 * zigLength);
              break;
            case 80:
             swiftbot.move(80 - 11, 80, 42 * zigLength);
              break;
            case 90:
             swiftbot.move(90 - 19, 90, 41 * zigLength);
              break;
            case 100:
             swiftbot.move(100 - 23, 100, 41 * zigLength);
              break;
        }
    }

    public static void energyEfficiencyMode() {
        mode = false;  // setting boolean mode to false will make it run the parts where colour is not
included
        System.out.println("Energy Efficiency Mode Selected");
        wheelSpeed = RandomSpeedGeneration.randomSpeed();  //calling randomSpeed method
from RandomSpeedGenration class to generate the wheel speed
        System.out.println("Disabling all underlights");
        System.out.println("Random Wheel Speed: " + wheelSpeed);

        patternSelection();  // goes to pattern selection  after Energy Efficiency Mode and asks the
user to select a pattern
    }

    public static void normalMode() {
        mode = true; // setting boolean mode to true will make it run the parts where colour is
included
        System.out.println("Normal Mode Selected");
        wheelSpeed = RandomSpeedGeneration.randomSpeed(); /* wheelSpeed is generated from
randomSpeed method of RandomSpeedGeneration class */
        System.out.println("Random Wheel Speed: " + wheelSpeed);
        patternSelection();                 // goes to pattern selection  after normal mode and asks the
user to select a pattern
    }

    /* Method for displaying the userInstructions */
    public static void userInstructions() {
        System.out.println("                              ******************************");
```

```
      System.out.println("                          Welcome to ZigZag Journey");
      System.out.println("                          ****************************");
      System.out.println(" User instructions:");
      System.out.println("- The SwiftBot will scan a QR code to begin the journey");
      System.out.println("- The SwiftBot has 2 modes, Normal Mode and Energy Efficiency
Mode");
      System.out.println("- The SwiftBot has 2 Patterns (Zigzag and Squareloop)");
      System.out.println("- The SwiftBot has the option to set 4 different colours for the
underlight");
      System.out.println("- The SwiftBot can capture photo during the journey if 'C' key is
pressed");
      System.out.println("- Use button 'X' to terminate the program anytime");
      System.out.println("- Ensure that the Swiftbot is placed in a location free of obstructions");
      System.out.println("- Confirm that the Swiftbot's battery is fully charged before initiating the
program.");
      System.out.println("                          ****************************");
   }
}

import swiftbot.Button;
import swiftbot.SwiftBotAPI;
public class ProgramTermination {
   private static boolean buttonXPressed = false; // by default buttonXPressed is set to false
   public static void termination(SwiftBotAPI swiftbot) { // passing parameters
      /* This method runs when the pattern is not squareLoop */
      if (Movement.squareloopcheck == false) {
         swiftbot.enableButton(Button.X, () -> {
            buttonXPressed = true; // setting to true means it has been pressed
            System.out.println("Button X Pressed.");
            System.out.println("Number of Zigzag completed: " + Movement.zigzagJourney); //
prints out Number of zigzag completed
            System.out.println("Terminating....");
            StartPrinting.writeToFile(); // calling writeToFile method from the StartPrinting Class
            System.out.println("Terminated");
            System.exit(0); // Exits
         });
      } else {
         System.out.println("Squareloop Completed");
         System.exit(0); // Exits
      }
      // Continuous check for the button press
      while (!buttonXPressed) {
         try {
            // Wait for 5 seconds to check if button X is pressed
```

```
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt(); // handle interrupted status
            System.out.println("Thread was interrupted, failed to complete operation");
        }
        // if button is not pressed this bit of code will run
        if ((buttonXPressed == false) && ((Movement.mode == false) || (Movement.mode ==
true))
                && (Movement.squareloopcheck == false)) {
            System.out.println("Button X was not pressed within 5 seconds. Continuing...");
            swiftbot.move(-100, 100, 400); // taking a left so that it can go in a loop
            Movement.zigZag(); // run zigzag again after taking a turn
        }
        /*
         * This condition checks if its a squareLoop or not and if its a squareLoop it will exist
         * instead of running an endless loop like zigzag
         */
        if ((buttonXPressed == false) && ((Movement.mode == false) || (Movement.mode ==
true))
                && (Movement.squareloopcheck == true)) {
            System.out.println("Squareloop Completed");
            System.exit(0);
        }
    }
  }
}


import java.awt.image.BufferedImage;
import swiftbot.SwiftBotAPI;
public class QrCodeHandling {
  public static void QRCodeDetection(SwiftBotAPI swiftbot) {


      try {
          System.out.println("Taking a capture in 2 seconds...");
          Thread.sleep(2000);  // waits 2 seconds before taking a capture
          BufferedImage img = swiftbot.getQRImage();   // image stored in an BufferedImage object
          /* Decoding the image and extract the message then
                                      finally stored in decodedMessage String*/
          String decodedMessage = swiftbot.decodeQRImage(img);
          if (decodedMessage.isEmpty()) {
              System.out.println("\u001B[31mNo QR Code was found. Please try
again.\u001B[0m");
```

```
            qrScan(swiftbot);   // will run qr scan method again and ask the user to scan again if Qr
is empty
        } else {
            String[] parts = decodedMessage.split(",");  // splits the decoded message into parts
until it finds a comma
            if (parts.length == 2 && isInteger(parts[0].trim()) && isInteger(parts[1].trim())) {
                Movement.zigLength = Integer.parseInt(parts[0].trim());  // first part is set to an
integer after trimming
                Movement.zigNumber = Integer.parseInt(parts[1].trim());// Second part is set to an
integer after trimming
                // validation check for zigLength and zigNumber
                if (Movement.zigLength < 15 || Movement.zigLength > 85 || Movement.zigNumber <
2 || Movement.zigNumber > 12 || Movement.zigNumber % 2 != 0) {
                    System.out.println("\u001B[31mFirst input must be any number from 15 to 85 and
second input must be even and can't be more than 12\u001B[0m");
                    qrScan(swiftbot);  // will be asked to scan QR again if the length or section
number is invalid
                } else {
                    System.out.println("QR Scan successful ^-^");
                    System.out.println("Zigzag Length: " + Movement.zigLength);
                    System.out.println("Number of Zigzags: " + Movement.zigNumber);
                    Movement.Calc = Movement.zigNumber / 2;     // formula from Software design
                }
            } else {
                // if parts not integers error will be shown
                    System.out.println("\u001B[31mFirst input must be any number from 15 to 85
and second input must be even and can't be more than 12\u001B[0m");
                qrScan(swiftbot); // goes to qrScn method if input is invalid
            }
        }
    } catch (Exception e) {
        qrScan(swiftbot);
    }
  }
  public static void qrScan(SwiftBotAPI swiftbot) {
    System.out.println("Press 'Q' to Scan");
    String input = Movement.scanner.nextLine().trim();
    /*Makes sure both lower case and upper case q is accepted*/
    if (input.equalsIgnoreCase("Q")) {
        QRCodeDetection(swiftbot);  // goes to QRCodeDetection method
    } else {
        qrScan(swiftbot);     // if something else is pressed it will recursively run the same method
    }
  }
```

```
 /*method to check if string is integer or not*/
  public static boolean isInteger(String str) {
     try {
        Integer.parseInt(str);
        return true;
     } catch (NumberFormatException e) {
        return false;
     }
  }
}

import java.util.Random;
                          /*Method for generating random speed between 40 to 100 ( all multiples
of 10)*/
public class RandomSpeedGeneration {
  public static int randomSpeed() {
     Random random = new Random();
     // Generate a number from 0 to 6, then add 4 to it, resulting in a range from 4 to 10
     int randomNumber = (random.nextInt(7) + 4) * 10;
     return randomNumber;
  }
}



import swiftbot.SwiftBotAPI;
public class Reverse {
        public static void retrace(SwiftBotAPI swiftbot) {
        System.out.println("Starting to Retrace. . . ");

        System.out.println("Turning around. . . ");
        swiftbot.move(100 , -100, 400); // right turn
          /*Condition to check if its Normal Mode or not and this will run if its zigzag not
squareLoop so
           * thats why squareLoopcheck is false*/
        for (int i = 1; i <= Movement.Calc; i++) {
        if (Movement.mode==true && Movement.squareloopcheck==false) {
        swiftbot.fillUnderlights(SetColours.selectedColor2); // fill with second colour
        Movement.adjustMoveBasedOnWheelSpeed(Movement.wheelSpeed,
Movement.zigLength);// moves forward with the adjusted wheel speed
     swiftbot.move(0, 0, 1000 );//pause for 1 second
     swiftbot.move(100 , -100, 400); // right turn
     swiftbot.fillUnderlights(SetColours.selectedColor);// fill with First colour
```

```
      Movement.adjustMoveBasedOnWheelSpeed(Movement.wheelSpeed,
Movement.zigLength);// moves forward with the adjusted wheel speed
      swiftbot.move(0, 0, 1000 );//pause for 1 second
      swiftbot.move(-100 , 100, 400); // left turn
         }/* This else will run if its zigzag and Energy saving mode*/
         else {
                  Movement.adjustMoveBasedOnWheelSpeed(Movement.wheelSpeed,
Movement.zigLength);// moves forward with the adjusted wheel speed
                  swiftbot.move(0, 0, 1000 );//pause for 1 second
                  swiftbot.move(100 , -100, 400); // right turn
        Movement.adjustMoveBasedOnWheelSpeed(Movement.wheelSpeed,
Movement.zigLength);// moves forward with the adjusted wheel speed
         swiftbot.move(0, 0, 1000 );//pause for 1 second
         swiftbot.move(-100 , 100, 400); // left turn

         }



         }
         System.out.println("Zigzag Completed");
  }


}

public class SetColours {
public static int[] selectedColor2;
public static int[] selectedColor;
                  /*Method for selecting first colour*/
public static void setfirstColour() {

                  System.out.println("Select a First colour:");
                  System.out.println("Press 1: Red");
                  System.out.println("Press 2: Green");
                  System.out.println("Press 3: Blue");
                  System.out.println("Press 4: Yellow");
                  int underLightSection1 = Movement.scanner.nextInt();
                  try {
                           switch (underLightSection1) {
                           case 1:
                                    selectedColor = new int[] { 255, 0, 0 }; // Red
                                    break;
                           case 2:
```

```
                selectedColor = new int[] { 0, 0, 255 }; // Green
                break;
            case 3:
                selectedColor = new int[] { 0, 255, 0 }; // Blue
                break;
            case 4:
                selectedColor = new int[] { 200, 0, 150 }; // Yellow
                break;
            default:
                setfirstColour();
                break;
            }

        } catch (Exception e) {

            Movement.scanner.next();
            setfirstColour();
        }
}



/*Method for Picking Second Colour */
public static void setSecondColour() {

        System.out.println("Select a Second colour:");
        System.out.println("Press 1: Red");
        System.out.println("Press 2: Green");
        System.out.println("Press 3: Blue");
        System.out.println("Press 4: Yellow");
        int underLightSection2 = Movement.scanner.nextInt();
        try {
            switch (underLightSection2) {
            case 1:
                selectedColor2 = new int[] { 255, 0, 0 }; // Red
                break;
            case 2:
                selectedColor2 = new int[] { 0, 0, 255 }; // Green
                break;
            case 3:
                selectedColor2 = new int[] { 0, 255, 0 }; // Blue
                break;
            case 4:
                selectedColor2 = new int[] { 200, 0, 150 }; // Yellow
```

```
                        break;
                    default:
                        setSecondColour();
                        break;
                    }
                }
                catch (Exception e) {
                    Movement.scanner.next();
                    setSecondColour();    // recursively calling method in Exception
                }
            }
}

import java.io.BufferedWriter;
import java.io.FileWriter;
public class StartPrinting {
  static double updatedzigLength = Movement.zigLength;
        /*Method for writing text to a text file */
  static void writeToFile() {

        double straightDis = (Math.sqrt(Math.pow(updatedzigLength, 2) +
Math.pow(updatedzigLength, 2)));
         straightDis = (updatedzigLength * Math.sqrt(2))*(Movement.zigNumber/2);       //
calculating straightDis as per Design
     try  {
        FileWriter writehandle = new FileWriter("//home//pi//Documents//log.txt"); // writehandle
object created to write in mentioned path
        BufferedWriter bw = new BufferedWriter(writehandle);

        bw.write("Zigzag length: " + Movement.zigLength + "cm");
        bw.newLine();
        bw.write("Number of sections: " + Movement.zigNumber);
        bw.newLine();
        bw.write("Random wheel speed: " + Movement.wheelSpeed);
        bw.newLine();
        bw.write("Traversed path Length: " + (Movement.zigLength * Movement.zigNumber) +
"cm");
        bw.newLine();
        bw.write("Duration:" + Movement.duration/1000 + " seconds");
        bw.newLine();
        bw.write("Distance travelled(Sraightline):" + straightDis + " cm");

        bw.close();  // closing bw
        writehandle.close();  // closing writehandle
```

```
        } catch (Exception e) {
            System.out.println("\u001B[31mError writing data to a text file. Check available storage
space or permissions\u001B[0m");
        }
    }
}
```