Assignment 3 - CS1810 Software Implementation

CS1701 Group Project Lectures & Tutorials

Task 6 - Search for Light

Name : Nabina Nujhat Chowdhury

ID number : 2377045

Tutor name : Anette Payne

Group Name : B29

Required Functionalities

1. UI shown indicating the buttons required to initiate(Button A) and end the programme (Button X).
2. The user initiates the program by pressing button 'A' on the Swiftbot.
3. The user terminates the program by pressing button 'X' on the Swiftbot.
4. Swiftbot's camera is activated and swiftbot function is used to record the time when the program is started.
5. The program captures an image of the surrounding environment using the Swiftbot's camera.
6. Flash green when the image passes the quality check.
7. An error message: "The uploaded image does not meet the minimum quality requirements" is displayed to inform the user that the image taken is not of sufficient resolution.
8. After the error message is displayed,the Swiftbot is prompted to take another image that meets the minimum quality requirements.
9. Image captured by Swiftbot camera is converted to a grayscale image(black-and-white),that indicates the light intensity in each pixel.
10. Grayscale is divided into three columns representing left, centre, and right directions.
11. Swiftbot is designed to calculate the average light intensity for left, right, and centre columns(higher number of black pixels).
12. The resultant calculations of the left,right and centre average light intensities are recorded.
13. The average light intensity of each column is compared to determine the brightest direction, by dividing the recorded light intensities values by 3.
14. Move Swiftbot towards the direction with the highest intensity, by setting underlights to green,to indicate the user that light has been found .
15. Move Swiftbot to the left for 0.5s, if the left column had the highest intensity.
16. If no light is detected on the left, end the program.
17. Move swiftbot forward for 0.5s, if the centre column has the highest intensity,If no light is detected in the centre, end the program.
18. Move swiftbot to the right for 0.5s, if the right column had the highest light intensity,If no light is detected on the right ,end the programme.
19. Check for objects 50 cm ahead before moving,as they will be acting as obstacles preventing Swiftbot from moving.

20. If an object has not been detected within 50cm, Swiftbot should continue searching for light and, If an object is detected, blink underlights in red to indicate obstacle/object the user should be informed to remove the object by displaying:”Object detected please remove”.

21. After the Swiftbot has detected an object ahead it will stop and wait for 10s for the object to be removed. If it is removed, the robot should continue searching for light.

22. If the object has not been removed,the following interface would be displayed: “Object detected! Program ended”;to inform the user the reason for why the programme has ended.

23. If the object has been removed, Swiftbot should keep moving, searching for light for up to 5s.

24. If no light source is found for 5 seconds, the Swiftbot should stop moving for 0.5 seconds.Swiftbot should turn randomly 90° to the left or to the right to continue searching for light, after stopping.

25. If a light source has been found, flash green to provide the user with a visual prompt, then the end time is recorded,signalling that the robot has stopped moving to look for light.

26. Then the user is asked If they want the log information to be displayed.

27. If the user wants to display the log, they should press the 'Y' button, which leads to the display of :

    - The Total distance travelled by the Swiftbot is displayed.
    - The Duration of execution of the programme is displayed.
    - The overall highest average intensity for each column of the grayscale image is displayed.
    - The number of times Swiftbot detected light, is displayed by making reference to the number of times it has flashed green.
    - The movements of Swiftbot performed during the programme are displayed to the user.

27.Before the programme is terminated, the log file is converted into a text file.

28.If the user doesn't want to display the log information, button 'X' is pressed.
Log file is converted into a text file and the program is terminated.

Additional Requirements

Swiftbot is meant to flash green when the image taken passes the quality check and when the swiftbot detects the column with the highest intensity after taking an image.

Functionalities Changed or Not Included in Implementation

The image quality control has been changed from 240x240 to 480x480, as it improves the quality and accuracy of the pixels when detecting light, aiding to calculate the average light intensities.

Have been unable to implement code that indicates in the display in the log when button Y or X is pressed before terminating the programme the total movements done by the Swiftbot along its journey.

Changes to the Algorithm and the UI Interface

Have included the following additions in order to optimise the UI and to maintain the user informed, so that error handling is easier:

Button A has been pressed → Informes the user that button A has been pressed, so the programme will initiate shortly.

Button Y has been pressed→ Informes the user that button Y has been pressed and the program will terminate shortly and display log will be shown along with the log file being converted into text file.

Button X has been pressed→ Informes user that button X has been pressed and the programme will end shortly.

"The uploaded image does not meet the minimum quality requirements"→ indicates to the user that the image taken by the Swiftbot is less than 480x480.

"Error occurred when processing image!"→ indicates to the user that there has been an error when dividing the image into columns or when calculating the average light intensities.

"Object detected, please remove"→indicates to the user that an object has been detected within 50 cm of the Swiftbot ultrasound and prompts the user to remove it, If they want the robot to keep searching for light.

"Object detected!Programme ended"→indicates to the user that an object has been detected within 50 cm of the Swiftbot ultrasound and the user has decided to not move it,instigating the termination of the programme.

"Object removed"→indicates to the user that an object has been detected within 50 cm of the Swiftbot ultrasound and the user has decided to remove the object, so Swiftbot can keep searching for light.

"Light source not detected!"→Indicates that the average light intensities of all the columns are the same and so light source has not been detected.

"Swiftbot rotated 90 degrees to the left."→swiftbot has rotated 90 degrees to the left randomly to keep looking for light.

"Swiftbot rotated 90 degrees to the right."→swiftbot has rotated 90 degrees to the right randomly to keep looking for light.

"Log information has been written to the log file."→the log information has been successfully written into a log file.

"An error occurred while writing to the log file."→there has been an error when writing the log information into a log file.

"An error occurred while displaying the log."→ there has been an error when displaying the log information when the user has pressed button Y.

Testing

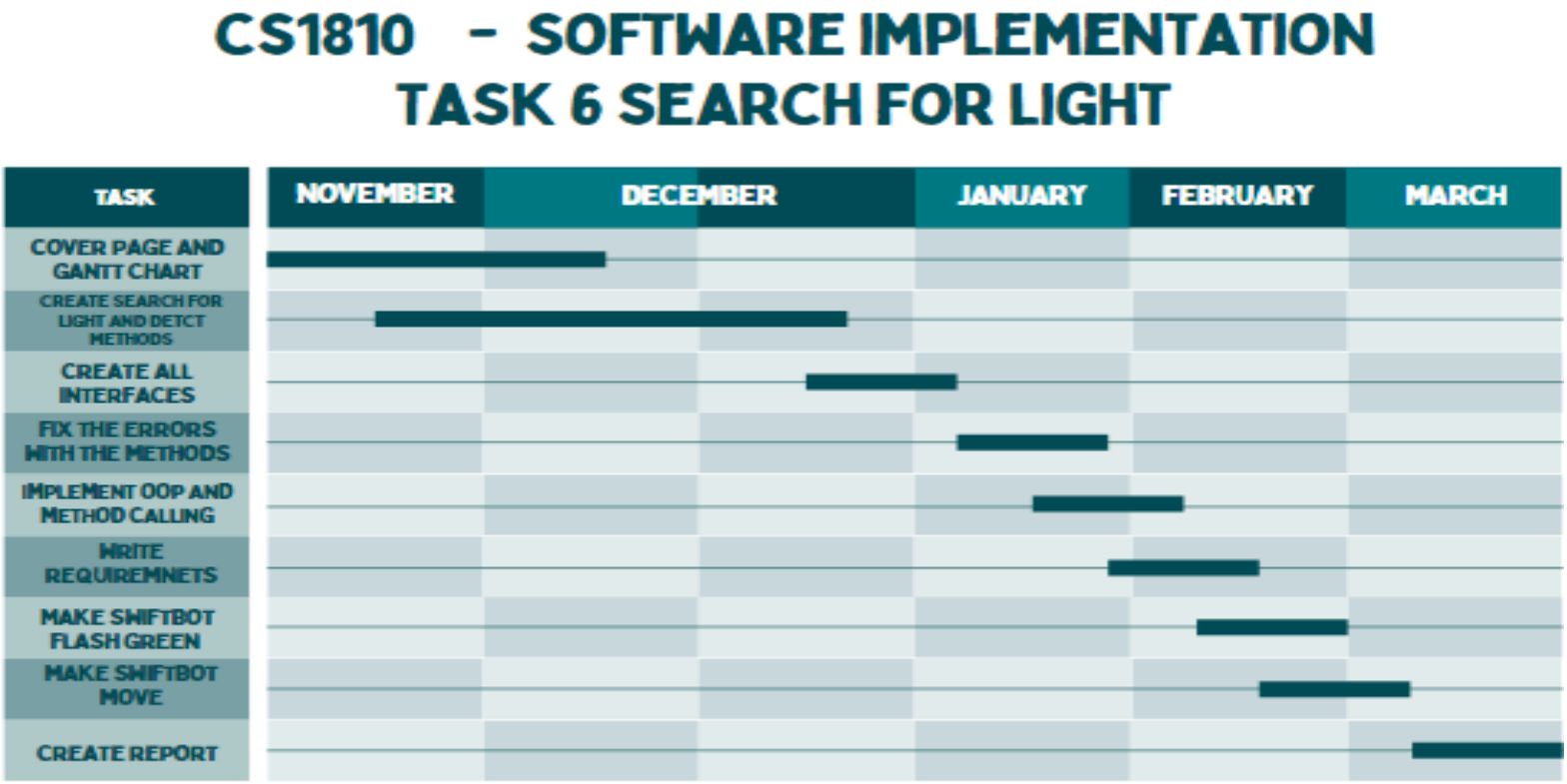
Test Case ID	Test Case Description	Requirement number	Test Data	Expected value/behaviour	Observed value/behaviour	Status	Comment
1	User interface displayed indicating the function of the indicated buttons to initiate and end programme	1	N/A	User interface is displayed to provide user with information on how to initiate or terminate programme	User interface is indeed displayed	pass	
2	Programme initiates when button A is pressed	2	Button A pressed initiates programme	Programme starts searching for light when Button A is pressed	Button A initiates the programme	pass	
3	Programme terminates when button X is pressed	3	Button X pressed terminates programme	Programme terminated when Button X	Button X exits the programme	fail	Disable the button with the respective API snippet

4	User interface displayed indicating when wrong button(not X or Y) is pressed	1	Button A pressed	User interface : “Invalid input! Please enter one of the following: …” Is displayed	Interface is displayed when wrong button is pressed	pass	
5	Flash green when image passes quality check	6	Block camera to incapacitate the Swiftbot from taking a image with good resolution	Swiftbot does not flash green as image quality falls below the minimum to pass quality check	Swiftbot does not flash green as image is not of enough quality	pass	
6	Display error message :“The uploaded image does not meet the minimum quality requirements”	7	Allow camera to take image with no obstacles blocking its way	Swiftbot display prompt to advise the user to set the swiftbot in an environment no obstacles	Swiftbot displays error message when placed in front of obstacle on all sides	pass	
7	Flash green before swiftbot moves to the column with the highest average light intensity	14	Situate the swiftbot in a environment in front of a white board where light is distributed evenly	Swiftbot flashes green before moving to the side with the highest intensity	Swiftbot flashes green before moving	pass	
8	Swiftbot moves left, if left column has the highest average light intensity	15	Flash light with the mobile torch to the swiftbot camera left side	Swiftbot moves left, if flash is lighted on the left side of the swiftbot camera	Swiftbot moves straight against the light source where flash is being lighted	fail	Increase the values of the right wheel movement and set the value of the left value to zero
9	Swiftbot moves forward, if centre column has the highest average light intensity	17	Flash light with the mobile torch to the swiftbot middle side	Swiftbot moves forward when flash light situated of the middle of the swiftbot camera	Swiftbot moves forward when flash lighted to the middle	pass	

10	Swiftbot moves right, if right column has the highest average light intensity	18	Flash light with the mobile torch to the swiftbot right side	Swiftbot moves Right when flash is shined to left side of the swiftbot camera	Swiftbot moves forward when flash is lighted on the right side of the camera	fail	Increase the values of the left wheel movement and set the value of the right value to zero
11	“No light detected” is displayed	24	Situate the swiftbot in a dark room with no light	Swiftbot should display :“No light detected”	“No light detected” is displayed	pass	
12	Flash red when obstacle is detected	20	Place obstacle close to the swiftbot front end	Swiftbot is expected to flash red to prompt user to remove object	Swiftbot bumps into object	fail	Implement the ultrasound to detect the object within 50cm
13	Turn randomly 90 degrees to right or left when no light source is found	24	Place light source in front of the swiftbot	Swiftbot is meant to turn to turn right or left in a 90 degree angle when no light source is found	Swiftbot continues to move forward instead of turning	fail	Create a separate method with java function random
14	User interface displayed indicating the functions of the button X or button Y to show or not the display log	26	N/A	interface displaying the functions of the button X or button Y to show or not the display log	Interface is displayed	pass	
15	User interface displayed when wrong button is pressed (not button X or button Y)	26	Press button B to test the interface	"...Invalid input!Please press one of the following:.. " displayed when wrong button pressed	"...Invalid input!Please press one of the following:.. " is displayed	pass	

16	Button Y is pressed to display log information before terminating programme	27,28	Press button Y to display the log information	Log information is displayed when button Y is pressed	Button y is pressed and log interface is shown	pass	

Planning and Monitoring Progress



Challenges faced	Overcome
Swiftbot not moving	Ask in labs, where TAs were helpful in targeting the code with errors and fixing the bug
Overcomplicating methods	Keep it simple with short length,reducing errors
Underestimating time required to write report	Booking study room to work silently where focusing is easier

Source Code Listing

```
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.util.Random;
import swiftbot.*;

public class swiftbot {

    private static final String IMAGE_FILE_NAME = "savedimage.jpg";
   private static SwiftBotAPI swiftbot = new SwiftBotAPI();
   private static int greentimes = 0;
   private static double OverallHighestaverageIntensityLeft=
Double.MAX_VALUE;
   private  static double  OverallHighestaverageIntensityCentre=
Double.MAX_VALUE;
   private  static double  OverallHighestaverageIntensityRight=
Double.MAX_VALUE;
   private static double totalDistance;
   private static double averageIntensityLeft;
   private static double averageIntensityCentre;
   private static double averageIntensityRight;
   private static long StartTimer;
   private static long EndTime;
   private static boolean objectDetected = false;
    private static double totaldistanceTravelled=0;



     public static void main(String[] args) {

         try {
            start();
            Thread.sleep(1500);
            searchForLight();
            lightsourcefound(averageIntensityLeft, averageIntensityCentre,
averageIntensityRight);
         } catch (Exception e) {
            e.printStackTrace();
         }
```

```
                }
// used to track the time where programme execution start
        public static void StartTimer() {
                StartTimer = System.currentTimeMillis();
        }

        public static void StopTimer() {
                EndTime= System.currentTimeMillis();
        }

        public static long Duration() {
                return EndTime-StartTimer;
        }

        //distance
        public static double calculateDistance(double startX, double startY, double
endX,double endY) {

                        double distanceX= endX-startX;
                        double distanceY= endY-startY;
                        return Math.sqrt(distanceX*distanceX +distanceY * distanceY);

                }
                //updated distance travelled
                public static void updateDistance(double startX, double startY, double
endX,double endY) {
                        double distance = calculateDistance(startX,startY,endX,endY);
                        totaldistanceTravelled+=distance;
                }
                //overall higher intensities

                 static double getOverallHighestaverageIntensityLeft() {
                        return OverallHighestaverageIntensityLeft;
                }
                 static double getOverallHighestaverageIntensityCentre() {
                return OverallHighestaverageIntensityCentre;
                }
                 static double getOverallHighestaverageIntensityRight() {
              return OverallHighestaverageIntensityRight;
                }

                //method run first including methods for enabling the buttons
                public static void start() throws InterruptedException {
                        inter1();
                        StartTimer();
                        try {
                                Thread.sleep(2000);
```

```
                        buttonA();
                        buttonX();
                        Thread.sleep(1100);
                } catch (InterruptedException e) {
                        e.printStackTrace();
                } catch (IllegalArgumentException e) {
                        e.printStackTrace();
                        // Handle IllegalArgumentException here
                }

                // Error handling when wrong button pressed
                swiftbot.enableButton(Button.Y, () -> {
                        inter1error();
                        swiftbot.disableButton(Button.Y);
                });
                swiftbot.enableButton(Button.B, () -> {
                        swiftbot.disableButton(Button.B);
                        inter1error();
                });

        }



        ///welcome for the first interface on command palette
        static void inter1() {

                System.out.println("
**************************************************");
                System.out.println("  Welcome to search for light programme");
                System.out.println("
**************************************************");
                System.out.println();
                System.out.println("Press button`A'to start the programme");
                System.out.println("Press  button`X'to terminate the programme");
                System.out.println();
                System.out.println("Thank you for your input the Swiftbot will start
the execution now");
        }

        // when button A has been pressed
        public static void buttonA() {
                swiftbot.enableButton(Button.A, () -> {
                        System.out.println("Button A has been pressed");
                        swiftbot.disableButton(Button.A);
                        swiftbot.disableButton(Button.X);
                        try {
```

```
                                searchForLight();
                            } catch (Exception e) {

                                e.printStackTrace();
                            }

                        });
                }
                //main method where light search and detecting objects take place
                 public static void searchForLight() {
                       while (!objectDetected) {
                          try {
                             BufferedImage img = camera(); // Capture image
                             if (img != null) {
                                // Image captured successfully
                                File outfile = new File(IMAGE_FILE_NAME);
                                ImageIO.write(img, "jpg", outfile);
                                underlightgreen();
                                greentimes++;
                                divideandaveragecolumns(img);
                                detectObj(); // Check for object after image capture
                                Thread.sleep(10000); // Wait for 10 seconds before capturing
next image
                             } else {
                                // Image capture failed
                                System.out.println("The uploaded image does not meet the
minimum quality requirements");
                             }
                          } catch (Exception e) {
                             // Exception occurred during image capture or processing
                             System.out.println("Error occurred when processing image!");
                             e.printStackTrace();
                          }
                       }
                    }
                 // when image fails to meet the requirement of 480 pixels
                public static BufferedImage camera() throws InterruptedException,
IOException {
                    BufferedImage img = null;
                    try {
                       img = swiftbot.takeGrayscaleStill(ImageSize.SQUARE_480x480);
                    } catch (Exception e) {
                       System.out.println("Error occurred when taking image!");
                       e.printStackTrace();
                    }
                    return img;
                }
```

```
		//dividing the gray scale into 3 columns
		public static String divideandaveragecolumns(BufferedImage img) {
			// Initialise variables to store the sum of RGB values for each
column
			int SumLeftColumn =0;
			int SumCentreColumn=0;
			int SumRightColumn=0;
		// Get the width and height of the image
			int w=img.getWidth();
			int h= img.getHeight();
		// Loop through each pixel in the image
			for (int y = 0; y < h; y++) {
				for (int x = 0; x < w; x++) {
				// Get the RGB value of the pixel
					int p = img.getRGB(x, y);
				// Extract the red, green, and blue components from the RGB
value
					int r = (p >> 16) & 0xFF;
					int g = (p >> 8) & 0xFF;
					int b = p & 0xFF;
				  // Determine which column the pixel belongs to and
accumulate the RGB values accordingly
					if (x<w/3) {
						SumLeftColumn+=(r+g+b);
					}else if(x<(2*w/3)) {
						SumCentreColumn+=(r+g+b);
					}else {
						SumRightColumn+=(r+g+b);
					}
				}
			}
		  // Calculate the average intensity for each column
			double averageIntensityLeft = (double) SumLeftColumn / (w * h / 3);
			double averageIntensityCentre = (double) SumCentreColumn / (w *
h / 3);
			double averageIntensityRight = (double) SumRightColumn / (w * h /
3);
		  // Calculate the average intensity for each column
			System.out.println("The average light intensity of Left column is: " +
averageIntensityLeft);
			System.out.println("The average light intensity of Centre column is:
" + averageIntensityCentre);
			System.out.println("The average light intensity of Right column is: "
+ averageIntensityRight);
```

```
                // Determine which column has the highest average intensity and move
the swiftbot accordingly

                        try {
                                if (averageIntensityLeft > averageIntensityCentre &&
averageIntensityLeft > averageIntensityRight) {
                                    System.out.println("The average light intensity of the left
column is the highest");
                                    swiftbot.move(0, 100, 450); // Move left
                                } else if (averageIntensityCentre > averageIntensityLeft &&
averageIntensityCentre > averageIntensityRight) {
                                    System.out.println("The average light intensity of the
centre column is the highest");
                                    swiftbot.move(100, 100, 450); // Move centre
                                } else if (averageIntensityRight > averageIntensityCentre &&
averageIntensityRight > averageIntensityLeft) {
                                    System.out.println("The average light intensity of the right
column is the highest");
                                    swiftbot.move(100, 0, 450); // Move right
                                }else {

                                }

                        } catch (IllegalArgumentException e) {
                                e.printStackTrace();
                        }
                        return null;
                }



                //is an object detected
                public static void detectObj() {
                try {
                    double distanceToObjectstill = swiftbot.useUltrasound();
                    System.out.println("Distance to object: " + distanceToObjectstill + " cm.");
                    if (distanceToObjectstill <= 50) {
                        System.out.println("Object detected, please remove");
                        underlightred();
                        Thread.sleep(500);
                        swiftbot.disableUnderlights();
                        objectDetected = true;
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
```

```
        }
    }


    // has object been removed
    public static void  hasobjectbeenremoved() throws InterruptedException,
IOException{
        double distanceToObjectstill =swiftbot.useUltrasound();
        if (distanceToObjectstill <= 50) {
            System.out.println("Object detected!Programme ended");
            try {
                underlightred();
                // Log file into text file??
                convertlogtotext();
                // Display log information
                displaylog();
            }catch (Exception e) {
                e.printStackTrace();
            }
            System.exit(0);

        }else {
            System.out.println("Object removed");
            searchForLight();
        }
    }

    public static void lightsourcefound(double averageIntensitycolumnLeft,
double averageIntensitycolumnCentre, double averageIntensitycolumnRight)
            throws IOException, InterruptedException {
//if light source is not found when all the columns have the same intensities
        if (averageIntensitycolumnLeft == averageIntensitycolumnCentre &&
                averageIntensitycolumnCentre ==
averageIntensitycolumnRight) {
            System.out.println("Light source not detected!");
            // Stop for 0.5s
            try {
                swiftbot.move(0, 0, 700);
            } catch (IllegalArgumentException e) {
                e.printStackTrace();
            }
            // Move randomly right or left 90 degrees
```

```
                }randomnumber();
                try {
                        searchForLight();
                } finally {
                        displaylog();
                }
        }

        // randomly select direction of movement
        static void randomnumber() throws InterruptedException {
        // Create a Random object to generate random numbers
                Random random = new Random();
         // Generate a random number between 0 (inclusive) and 2 (exclusive)
                int direction = random.nextInt(2);
         // Check the random number to determine the direction of movement
                if (direction == 0) {
                        //if direction is 0, move left
                        try {
                                swiftbot.move(0,70,1000);
                                Thread.sleep(1100);
                        } catch (IllegalArgumentException e) {
                                e.printStackTrace();
                        }
                        System.out.println("Swiftbot rotated 90 degrees to the left.");
                } else {
                        try {
                                swiftbot.move(70, 0, 1000);
                                Thread.sleep(1100);
                        } catch (IllegalArgumentException e) {
                                e.printStackTrace();
                        }

                        System.out.println("Swiftbot rotated 90 degrees to the
right.");
                }
        }

        static void convertlogtotext() {
                try {
                        // Create or append to the log file
                        FileWriter logWriter = new FileWriter("searchforlightlog.txt",
true);
                        logWriter.write("Log information:\n");
                        logWriter.write("Overall highest average intensity observed
in left section: " +OverallHighestaverageIntensityLeft + "\n");
                        logWriter.write("Overall highest average intensity observed
in centre section: " + OverallHighestaverageIntensityCentre + "\n");
```

```
                                logWriter.write("Overall highest average intensity observed
in right section: " + OverallHighestaverageIntensityRight + "\n");
                                logWriter.write("Number of times the Swiftbot detected
light: " + greentimes + "\n");
                                logWriter.write("Total distance travelled: " +
totaldistanceTravelled/100.0 + "metres");
                                logWriter.write("Duration of the execution: " + Duration() + "
milliseconds");
                                logWriter.close();
                                System.out.println("Log information has been written to the
log file.");
                        } catch (IOException e) {
                                System.out.println("An error occurred while writing to the log
file.");
                                e.printStackTrace();
                        }
                }

                public static void displaylog() {
                        try {
                                interfaceloginformation();
                                Thread.sleep(1000);

                                // Disable any previously attached functions before adding
new ones
                                swiftbot.disableButton(Button.X);
                                swiftbot.disableButton(Button.Y);
                                swiftbot.disableButton(Button.A);
                                swiftbot.disableButton(Button.B);

                                swiftbot.enableButton(Button.X, () -> {
                                        System.out.println("Button X has been pressed");
                                        System.exit(0);
                                });

                                swiftbot.enableButton(Button.Y, () -> {
                                        System.out.println("Button Y has been pressed");
                                        StopTimer();
                                        interfacebuttonypressed(averageIntensityLeft,
averageIntensityCentre, averageIntensityRight, greentimes, totalDistance, Duration());
                                        convertlogtotext();
                                        try {
                                                Thread.sleep(1100);
                                        } catch (InterruptedException e) {
                                                e.printStackTrace();
                                        }
```

```
                                        System.exit(0);
                                });

                                swiftbot.enableButton(Button.A, () -> {
                                        interfaceloginformationerror();
                                });

                                swiftbot.enableButton(Button.B, () -> {
                                        interfaceloginformationerror();
                                });

                        } catch (Exception e) {
                                System.out.println("An error occurred while displaying the
log.");
                                e.printStackTrace();
                        }
                }

                static void interfaceloginformation() {
                        System.out.println("
**************************************************");
                        System.out.println("Would you like to display the log of information
?");
                        System.out.println("
**************************************************");
                        System.out.println(" Press the‘X’button for NO \r\n"
                                        + " Press the‘Y’button for YES");
                        System.out.println(" Your chosen option will result in the conversion
of login formation to a text file.");
                        System.out.println(" Thank you for using the Search for Light
programme!");
                }
                static void interfaceloginformationerror () {
                        System.out.println("
**************************************************");
                        System.out.println("Would you like to display the log of information
?");
                        System.out.println("
**************************************************");
                        System.out.println(" Press the‘X’button for NO \r\n"
                                        + " Press the‘Y’button for YES");
                        System.out.println(" Invalid input! Please enter one of the following:
");
                        System.out.println(" Press the‘X’button for NO \r\n"
                                        + " Press the‘Y’button for YES");
                        System.out.println(" Your chosen option will result in the conversion
of login formation to a text file.");
```

```
                System.out.println(" Thank you for using the Search for Light
programme!");
        }

        static void interfacebuttonypressed(double averageIntensityLeft2, double
averageintensityCentre2, double averageIntensityRight2, int greentimes2, double
totalDistance2, long duration2) {
                System.out.println("
**************************************************");
                System.out.println(" Would you like to display the log of information
?");
                System.out.println("
**************************************************");
                System.out.println(" Press the'X'button for NO \r\n"
                                + " Press the'Y'button for YES");
                System.out.println(" Overall  highest average intensity observed in
left section: "+ OverallHighestaverageIntensityLeft);
                System.out.println(" Overall highest average intensity observed in
centre section: "+ OverallHighestaverageIntensityCentre);
                System.out.println(" Overall highest average intensity observed in
right section:  "+OverallHighestaverageIntensityRight );
                System.out.println(" Number of times the Swiftbot detected light: "
+ greentimes);
                System.out.println(" Movements of the Swiftbot: ");
                //converts the total distance travelled from centimetres to metres
                double totaldistanceTravelledmetres = totaldistanceTravelled/100.0;
                System.out.println(" Total distance travelled: "+
totaldistanceTravelledmetres + " metres");
                System.out.println(" Duration of the execution: " + Duration() + "
millisiseconds");
                System.out.println(" Your chosen option will result in the conversion
of login formation to a text file.");
                System.out.println(" Thank you for using the Search for Light
programme!");
        }



        //set under lights to green
        static void underlightgreen() throws IOException, InterruptedException {
                int[] colourToLightUp = {0, 0, 255};
                try {
                        swiftbot.fillUnderlights(colourToLightUp);
                        Thread.sleep(1000);
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
```

```
                swiftbot.disableUnderlights();

        }
        static void underlightred() {
                int[] colourToLightUp = {255, 0, 0};
                try {
                        swiftbot.fillUnderlights(colourToLightUp);
                        Thread.sleep(1000);
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
                swiftbot.disableUnderlights();

        }


        //press buttoned X
        public static void buttonX() {
                swiftbot.enableButton(Button.X, () -> {
                        System.out.println("Button X has been pressed");
                        swiftbot.disableButton(Button.X);
                        System.exit(0);
                });
        }

        //what interface shows when invalid button pressed
        static void inter1error() {
                System.out.println("
**************************************************");
                System.out.println("  Welcome to search for light programme");
                System.out.println("
**************************************************");
                System.out.println();
                System.out.println("Press button`A'to start the programme");
                System.out.println("Press  button`X'to terminate the programme");
                System.out.println();
                System.out.println("Invalid input!Please press one of the following:
");
                System.out.println();
                System.out.println("Press button`A'to start the programme");
                System.out.println("Press  button`X'to terminate the programme");
                System.out.println();
                System.out.println("Thank you for your input the Swiftbot will start
the execution now");
        }

    }
```