Brunel University London

SOFTWARE IMPLEMENTATION

DETECT OBJECT

TAHMID AL SIFAT

BSC IN COMPUTER SCIENCE (1ST YEAR)

BRUNEL UNIVERSITY LONDON

Table of Contents

Implementation Summary

Functional Requirements

1. SwiftBot Mode Selection :

a) The program asks the user to scan a QR code using the bot's camera.
b) The user scans a QR code to select one of three modes .
c) Curious SwiftBot,
d) Scaredy SwiftBot and
e) Dubious SwiftBot
f) The program should display the chosen mode after scanning the QR code and execute the mode.
g) The bot should wander around at the beginning of every mode.
h) The bot's ultrasound sensors will be activated so that it starts measuring the distance from each object or obstacle it encounters.

2. Wandering Around :

a) The bot should set its underlights as blue.
b) For Curious mode , the bot should check first if any object is at the buffer zone if not , then it will keep wandering around.
c) For Scaredy mode , the bot should check first if any object is within the 50 cm distance , if not , then it will keep wandering around.
d) The bot should move forward for 2 seconds at 40% Speed.
e) The bot should move right for 3 seconds at 80% Speed.
f) The bot should move forward again for 3 seconds at 40% Speed.
g) The bot should move right again for 3 seconds at 90% Speed.
h) The bot will keep moving forward at 35% speed.

3. The Curious SwiftBot Mode :

a) The bot detects an object using ultrasound sensors by calculating the distance of the object from the bot, the bot keeps storing the distance intakes.
b) The bot should maintain a 29 to 31 cm buffer zone between the SwiftBot and the object . The object has to be exactly at the zone .
c) If the distance from the object is more than 31 cm, then the bot should move forward to make the distance within buffer zone while undelights are set as Green.
d) If the distance from the object is less than 29 cm, then the bot should move backwards to make the distance within buffer zone while underlights are set as Green.
e) When the distance of the object is exactly at the buffer zone , blink underlights as green.
f) Upon establishing a buffer zone the bot should not move and will remain stationary.
g) The program counts the object.
h) The bot should take a picture of the object, save the image in a file path in a Swiftbot folder.
i) The program should start a 5 seconds timer in the backend.
j) The bot should wait for 5 seconds and check if the object has moved.
k) The bot should check the movement of the object by calculating if the distance has changed or not.
l) The bot should adjust the position to maintain the buffer zone if the object has moved.
m) After 5 seconds of no object detection or movement, the bot should wait for 1 more second and start moving again in a slightly different direction.
n) The same process will be repeated unless the user wants to quit.

4. The Scaredy SwiftBot Mode :

a) The bot detects an object using ultrasound sensors by calculating the object's distance from the bot, the bot keeps storing the distance intakes.
b) The bot should detect an object if it is within a 50-cm distance.
c) The bot should click an image of the object and save the image in a file path in the user's device as well as count it.
d) After detecting an object, the bot should blink the underlights as red.
e) The bot should set the underlights as Red again.
f) The bot moves backwards.
g) The bot turns 180 degree.
h) The bot moves forward again for 3 seconds.
i) After the turning, the bot should remain stationary.
j) The bot turn off the underlights .
k) The bot should check if the object has moved by checking if the distance is more than 50 cm.
l) If the object has moved, then the bot should change direction until it detects another object.
m) If no new object is detected for 5 seconds, the bot should wait for 1 more second, change direction and start moving again.
n) The same process will be repeated unless the user wants to quit.

5. Dubious SwiftBot Mode :

a) The program selects a random number between 1 and 2
b) If it is 1, then the program should execute the Curious Mode
c) If it is 2, then the program should execute Scaredy Mode
d) The mode ends after executing one of the mentioned modes

6. Termination Process :

a) If the user presses the "X" button on the SwiftBot, the termination process begins.
b) The program should calculate the total duration of the operation.
c) The program should create a log file to save the execution log.
d) The program should display and provide the user with two options.
e) If the user wants to view the execution log, then the user should press the "Y" button on the SwiftBot.
f) The execution log should contain the name of the mode it ran, the duration of the operation, the number of objects it encountered, the file path where the images are saved and the file path where the log file is saved.
g) If the user does not want to view the execution log, then the user should press the "X" button again on the SwiftBot.
h) If the X button is pressed again, then the program should display only the file path where the log file is saved.
i) The program ends.

7. Command-Line Interface

The program should be controlled via a simple, interactive CLI that prints outputs and takes input from the user.

Non-Functional Requirements

1. The program should validate the scanned QR codes.
2. The program should reject invalid QR codes and display an error message instructing the user to scan the QR code again.
3. The program should also validate the button pressings on SwiftBot, if the wrong buttons are pressed, it should prompt the user to press them again.
4. There should be display messages for the user when SwiftBot is :
    a) Wandering around.
    b) Adjusting position.
    c) Changing direction.
    d) Detects an object.
    e) Begins termination of the program.
    f) Ending the program.

Additional Functionalities

1. To scan the QR code, the program will ask the user to press the "A" Button on the SwiftBot to initiate the scan.
2. A 20 seconds timer will start upon pressing the A button.
3. If the program can not find any QR code within that 20 seconds the program will prompt the user to press A button again .
4. The program should display a message saying " Scanning in progress".
5. If any other button rather than A is pressed for QR scan , the program should validate it .
6. While scanning the QR code, the underlight should be set to Purple.
7. After detecting an object, the program should ask the user with a display message if the user wants to terminate the program.
8. The program will ask the user to press the "X" button on the SwiftBot to proceed to termination. After the user presses X to terminate, the program displays "Termination in progress" before exiting.
9. The program should show a Welcome message at the beginning of the program: "Welcome to the Object Detect operation by SwiftBot :D ".
10. If the user presses "Y" inside the termination process, the underlights should blink as Orange.
11. If the user presses "X" inside the termination process, the underlights should blink as Yellow.

My implementation included most of the requirement specifications that were mentioned in the software design . Some changes have been made . In Functional Requirement for wandering around , 2(b), 2(c) were added so that if the bot detects any object within the required zone during the modes , the bot should stop wandering around . Also in 2(d-h) , I changed some of the speed and duration of the movements for better execution. In 3(b) , I modified the buffer zone for Curious mode as 29 to 31 cm instead of exactly 30cm , so that the user experience of detecting objects get better as exact 30cm can be hard to achieve sometimes. Also for Scaredy mode , in 4(l) , I added that if the object has moved , the bot should change direction until it finds another object , as in the first place it was not clear enough regarding what should it do in this case . I removed NF-3 and AF-5 from main implementation as validating button pressing with Swiftbot's methods makes the code complicated and would create mess for a task that is not much significant. Finally , I added AF-2 and 3 , basically a 20s timer for scanning the QR code . The timer makes the QR scanning experience more professional and create a logical validation as the program does not have to wait forever for the QR code .

Changes to the Algorithm and User Interface Design

My software implementation largely followed the structure of my design flowcharts, but along the way, I made a few small adjustments to improve efficiency and functionality. First, I refined the Wander Around sub-process by integrating specific conditions for both Curious Mode and Scaredy Mode right from the start. I also fine-tuned the movement speed and duration, ensuring that the bot moves in a way that feels more fluid and natural. For Scaredy Mode, I introduced a key improvement—rather than simply resuming its wandering behavior when an object moves, the bot now instantly changes direction. This small but significant tweak makes the bot's reactions more responsive, engaging, and lifelike, giving it a sense of awareness and personality. These changes enhance the overall user experience, making interactions with the bot feel more immersive and dynamic.

I successfully implemented the User Interface Design as planned in my original design proposal. I also made the ASCII arts properly. However, I made a few minor adjustments, such as slight changes to certain symbols. For example, instead of the following : [2]

```
>>> Taking an image of the object <<<
```

```
<>  Current distance is 30 cm  <>
```

I replaced these with following for better appearance in CLI :

```
|><| Taking an image of the object  |><|
```

```
====  Current distance is 42.22 cm  ====
```

Overall, the UI design remains true to the original plan and functions smoothly without any issues. The interface is both intuitive and responsive, ensuring a seamless user experience while maintaining the intended aesthetics and functionality.

Testing

Test Case ID	Test Case Description	Requirement number	Test Data	Expected Behavior	Observed Behavior	Status	Comments
1	The program shows welcome message and asks the user to scan a QR code with A button press	FR-1(a) AF- 1,9	Null	The mentioned messages should be shown with applied user interface design	The program displays the messages without any error	Pass	No delay spotted
2	The program detects the A button press on SwiftBot and starts 20 seconds timer	AF – 2,4	"A" button press on the bot	The 20s timer will start with display messages	The timer starts and works properly with display messages	Pass	
3	The user shows an image of the QR code from mobile or paper or any other medium	FR-1(b)	Pre generated QR code images	The bot will be able to take picture of the QR code and decode it accordingly	The bot recognises the QR image and decodes it	Pass	User has to place the QR code properly in front of the bot's camera
4	If the QR code contains the text Curious/ Scaredy/Dubious the program will display and execute curious mode	FR-1(c),1(d), 1(e) ,1(f)	"Curious" " Scaredy" "Dubious"	The program should display the mentioned mode and execute it	The program displays the mode and executes it precisely	Pass	Some adjustment of position might require while showing QR to bot
5	If the QR code is empty or contains any other texts rather than those 3 modes program will validate inputs	NFR-1,2	Faull Null "(scry)" "Curis " "667" "#dubious"	The program should reject the inputs and tell the user to scan again	The program rejects the inputs with display and asks the user to scan again	Pass	The precision is accurate
6	If the Bot can not find any QR within the 20s timer, the program will ask user to press A again	AF-3	Null	After 20s , the program should ask the user to press A button to scan again	Program asks the user to press A button again for scanning	Pass	

Test Case ID	Test Case Description	Requirement number	Test Data	Expected Behavior	Observed Behavior	Status	Comments
7	While scanning the QR code, the underlight should be set to Purple	AF-6	A button press	The Swiftbot should set its undelights as purple	The undelights ON as purple colour	Pass	
8	If wrong buttons are pressed the program will validate that	NFR-3 AF -5	"B" button	The prog-ram will reject and ask user to scan again	Could not validate different button press	Fail	
9	The bot should wander around at the beginning of every mode	FR- 1(g)	Showing a valid QR image	The bot should start wandering around at the start of the mode	The bot execute wandering around at every beginning	Pass	Wander around works as separate method
10	Ultrasound sensors will be activated so that it starts measuring the distance from each object or obstacle it encounters.	FR – 1(h), 3(a), 4(a)	Obstacles placed in front of the bot	The bot should continuously measure and display distance from every obstacle it encounters	The bot measures & displays distance from every obstacle it encounters at no interval	Pass	Sensors work better when Objects/obstacle are solid
11	The bot's underlights will be set as blue during wandering around	FR- 2(a)	Null	Blue underlightsbe seen during wandering around	Blue underlights working fine during wandering around	Pass	
12	The Swiftbot's consecutive wandering around movements as forward , right,forward,right,forward at desired duration, speed	FR-2(d), 2(e), 2(f), 2(g), 2(h)	Null	Bot should (move direction, speed, seconds) : (Forward , 40% , 2); (Right , 80%, 3); (Forward , 40%,3); (Right , 90% , 3) ; (Forward , 35%,2)	The bot's movement in wandering around precisely follows the direction , speed and duration data mentioned	Pass	The bot's speed may vary depending on the type of floor (friction) it is operating on
13	In the modes , if any object is in the required zone , bot stops wandering around or keep going otherwise	FR-2(b), 2(c)	Object placed within the zones	Upon detecting the object within buffer/ 50 cm zone , the bot should stop wandering around	The bot stops wandering around to execute the main modes	Pass	[1]

Test Case ID	Test Case Description	Requirement number	Test Data	Expected Behavior	Observed Behavior	Status	Comments
14	The bot shall always maintain a buffer zone of 29 to 31 cm from the object	FR – 3(b), 3(f), 3(l)	Object is placed within 29 to 31 cm	The bot should maintain the buffer zone , and adjust position accordingly	The bot maintains the buffer zone and adjusts position otherwise	Pass	
15	The bot shall move forward/backward if the object is at out of the buffer range with green underlights on	FR- 3(c) 3(d)	Object is moved outside zone	With green underlights on , the bot should calculate the distance to maintain the buffer zone and move forward/ backward accordingly	The bot properly calculates the distance required to establish the buffer zone and moves front/back accordingly	Pass	The calculation is based on the speed test data which has slight errors depending on friction and time
16	The tasks the program does after a successful Object detection at the buffer zone	FR- 3(e), 3(f), 3(g), 3(h)	Object placed at the buffer zone	Program should : blink the underlights as green , remain stationary , count the object , take a picture of the object and save the image at designated folder	The program precisely does all of the mentioned tasks sequentially without any error or delays	Pass	The picture quality may depend on the light flux
17	5 seconds timer in the background	FR- 3(i), 4(m)	Null	The 5s timer should start and work	The timer starts and works perfectly	Pass	This timer is not displayed
18	If the object is moved , the bot should detect it	FR- 3(k), 4(k)	Object displaced	The bot should detect movement by distance calculating	The bot properly detects movement of the object	Pass	
19	After the timer ends , if no movement detected , the bot waits 1s and change direction	FR – 3(m), 4(m)	Null	The bot should wait one more second and execute change direction	The bot waits and execute change direction method on time	Pass	

Test Case ID	Test Case Description	Requirement number	Test Data	Expected Behavior	Observed Behavior	Status	Comments
20	The mode will be kept on loop unless the user wants to quit the program	FR-3(n) 4(n)	Null	The mode program should run spontaneously until user wishes to quit	The mode program keeps on running by loop until user wishes to quit	Pass	Termination depends on user's choice , the mode does not end by self
21	After detecting an object, the program should ask the user with a display message if the user wants to terminate the program within that 5s timer	AF-7,8	An object is detected	Program should display a message giving user choice to quit by pressing X button	The program displays the message and provides choice after detecting a object	Pass	User gets 5 seconds time to press the X button for termination
22	Object Detection within 50 cm zone	FR-4(b)	Object is within 50cm	The bot should detect an object if it is within 50cm	Object detected successfully	Pass	
23	After detecting an object , the bot takes an image and saves the image in designated folder and blink underlights as red & counts the object	FR-4(c), 4(d)	Null	The bot should take a still picture , save it at the folder , blink the underlights as red and count the object	The bot properly completes all the mentioned tasks at no delay	Pass	The folder needs to be refreshed to view the latest saved images
24	After detecting an object , the bot performs the scaredy movement including opposite turn and red underlights	FR – 4(e), 4(f), 4(g), 4(h), 4(i), 4(j)	Null	The bot should : Set Underlights as Red, move back and turn 180 degree , move forward for 3 seconds , stop & turn off lights	The bot does all the tasks properly in subsequent manner	Pass	The movement depends on the friction of the floor/ground
25	"X" button press to initiate termination	FR-6(a)	X button press	The program should display termination page and execute it	Termination page displayed and executed	Pass	

Test Case ID	Test Case Description	Requirement number	Test Data	Expected Behavior	Observed Behavior	Status	Comments
26	In dubious mode , the program should randomly select between Curious or Scaredy mode and execute it	FR-5(a), 5(b), 5(c), 5(d)	QR image containing “dubious”	The program should randomly select a number between 1 and 2 , to execute one of the modes.	Random generation of mode , selection and execution have been successful.	Pass	The selection is totally random
27	The program calculates the total duration of operation	FR-6(b)	Device time	(End time – start time) should be done to calculate	Total duration of operation calculated in seconds	Pass	Device's bios time should be correct
28	The program writes a new file each time when a mode runs	FR-6(c)	Null	The new file should be created and execution data be saved	File created and execution data saved	Pass	The folder needs to be refre - shed everytime
29	User will be provided with two options , Y or X button press	FR-6(d), 6(e), 6(g)	X button press	Either press Y or X , Y for execution log , X for just log file path	The options are given and buttons initiated properly	Pass	
30	The data that Execution log file will contain and display if user presses Y as well as blink undelights as orange	FR-6(f), AF-10	Y button press	The log file should save and display : Mode, Duration of operation , Number of objects and image file path and orange blink as button press	The log file contains all the mentioned data and displays it upon Y button press , the undelights blinks properly as orange	Pass	The user interface of this part is appealing
31	Printing of log file path with Yellow undelights blink as X button press	FR-6(h) AF-11	X button press	The log file path should be displayed and lights blink as yellow	The file path displayed with yellow blink upon button press	Pass	
32	The program ends	FR-6(i)	X / Y button press	The program should exit without error	Exits without any error	Pass	The program Smoothly exits
33	Command Line Interface as control screen	FR – 7	Null	Program should be executed in CLI	Operates perfectly via CLI	Pass	

Main Class

```
package object_detect_program;


import java.awt.image.BufferedImage;
import java.io.FileWriter;
import java.io.IOException;
import swiftbot.SwiftBotAPI;
import swiftbot.Button;
import java.text.SimpleDateFormat;
import java.util.Date;


// This is the main class from where the program is structured


public class ObjectDetect_Main {


// Using static variables for global accessibility ( can be accessed from other classes ) and better
convenience


static SwiftBotAPI swiftBot; // Declaring SwiftBot API for global usability
static String selectedMode = ""; // variable that will display Selected Mode
static int objectCount = 0; // for counting the number of objects encountered
static long startTime = System.currentTimeMillis(); // To calculate the duration and initiate timer
static String imageFilePath = "/data/home/pi/object_detect_program/Images_objectD/"; // inside the
swiftBot
static boolean terminateCheck = false; // flag to store termination press


// start of main method


/* Here, throws InterruptedException in the main method is used to handle potential
interruptions during thread operations like Thread.sleep() without catching the exception locally */


public static void main(String[] args) throws InterruptedException
{
try {


swiftBot = new SwiftBotAPI(); // declaring the SwiftBot API as class object


}
```

```
catch (Exception e)
{
/*
* Outputs a warning if I2C is disabled. For a better error handling
* This only needs to be turned on once
*/
System.out.println("\n Error handling SwiftBot API ");
System.out.println("\nI2C disabled!");
System.out.println("Run the following command:");
System.out.println("sudo raspi-config nonint do_i2c 0\n");
System.exit(5);
}

while(!terminateCheck)
{
// an infinite loop to keep the program running until terminated otherwise
// Enable X button globally for termination , so that user can initiate termination
// any time during the program
swiftBot.disableButton(Button.X); // disabling the Button before performing the task
swiftBot.enableButton(Button.X, () -> {
swiftBot.disableButton(Button.X);
System.out.println("\n| >>>>> X button pressed. Initiating termination... <<<<< | \n");
terminateCheck=true;
terminateProgram(); // Call termination process


});


displayWelcomeScreen(); // The welcome user interface will be displayed
swiftBot.disableButton(Button.A);
// calling the method QR scan
QRscan();


} // while loop ends


} // main method ends

// Method for scanning the QR code


public static void QRscan() throws InterruptedException {


// ANSI escape codes to make the UI better
final String RED = "\033[31m"; // Red text
final String CYAN = "\033[36m"; // Cyan text
final String BG_CYAN = "\033[46m"; // Cyan background
final String BG_RED = "\033[41m"; // Red background
final String BG_BLUE = "\033[44m"; // Blue background
final String RESET = "\033[0m"; // Reset to default color
```

```
System.out.println(BG_BLUE+" "+RESET);
System.out.println("------------------------------------------------------------------------------");
System.out.println("--- Please press 'A' button on the bot to scan a QR code using Bot's camera ---");
System.out.println("------------------------------------------------------------------------------");
System.out.println(BG_BLUE+" "+RESET);


// Using a flag to wait for the A button press , so that it can be updated
boolean[] aButtonPressed = { false };

swiftBot.disableButton(Button.A); // Disabling Button A first to avoid multiple callbacks
swiftBot.enableButton(Button.A, () -> {
swiftBot.disableButton(Button.A); // Disable after press
aButtonPressed[0] = true; // Set flag to true
});


// Wait for A button press
while (!aButtonPressed[0]) { // This loop will run if A button is not pressed
Thread.sleep(100); // Small delay to avoid busy-waiting and smooth operation
}


// Proceed with QR scanning


System.out.println(CYAN+"\n<><><><><><><><><><><><><><><>"+RESET);
System.out.println(CYAN+"<><>"+RESET+" Scanning in progress"+CYAN+" . . ."+RESET);
System.out.println(CYAN+"<><><><><><><><><><><><><><><>\n"+RESET);


long startTime = System.currentTimeMillis();
long endTime = startTime + 20000; // 20 seconds in milliseconds


System.out.println(BG_CYAN + " +++ Starting 20s timer to scan a QR code +++ " + RESET+ "\n");

try {
while (System.currentTimeMillis() <= endTime) { // This loop will run till 20 seconds


BufferedImage img = swiftBot.getQRImage();
String QRin = swiftBot.decodeQRImage(img); // storing the text inside QRin


selectedMode=QRin; // transferring it to selected mode so that we can use it in termination


int[] rgb = { 128, 0, 128 }; // purple
swiftBot.fillUnderlights(rgb);


if (QRin.isEmpty())
{ // for QR validation 1
System.out.println(BG_RED + "Invalid QR code , please scan again !" + RESET + "\n");
}
```

```
else {
System.out.println(BG_CYAN+" QR code found and decoded successfully "+RESET+ "\n");
swiftBot.disableUnderlights(); // turning the lights off after decoding

System.out.println("\n");


// The mode selection will be done as well as QR validation 2 with switch statement


switch (QRin.toLowerCase()) // making the text all lower case to avoid case errors
{
case "curious":
System.out.println(CYAN+"----------------------------------------"+RESET);
System.out.println(CYAN+"| >>>>>>" +RESET+" Executing Curious Mode"+CYAN+" . . . |"+RESET);
System.out.println(CYAN+"----------------------------------------\n"+RESET);
new CuriousMode(swiftBot, imageFilePath).run();
// calling the class Curious mode to execute the mode , the class takes input
// as the api and imageFilePath
break;
// the switching stops


case "scaredy":
System.out.println(CYAN+"----------------------------------------"+RESET);
System.out.println(CYAN+"| >>>>>>" +RESET+" Executing Scaredy Mode"+CYAN+" . . . |"+RESET);
System.out.println(CYAN+"----------------------------------------\n"+RESET);
new ScaredyMode(swiftBot, imageFilePath).run();
break;


case "dubious":
System.out.println(CYAN+"----------------------------------------"+RESET);
System.out.println(CYAN+"| >>>>>>" +RESET+" Executing Dubious Mode"+CYAN+" . . . |"+RESET);
System.out.println(CYAN+"----------------------------------------\n"+RESET);
new DubiousMode(swiftBot, imageFilePath).run();
break;
// if the QR does not match any of the 3 modes , then we call the method again


default:
System.out.println(BG_RED + "Invalid QR code , please scan again !" + RESET + "\n");
QRscan(); // Recursive call to scan again
} // switch ends


break; // the Loop breaks
} // else ends


if (System.currentTimeMillis() > endTime) // 20s timer ends
{
System.out.println(BG_RED + "\\n Times Up! Please Scan Again" + RESET + "\n");
}

} // Main while loop for scanning QR code ends
} // try block ends
```

```
catch (Exception e)
{ // Handling any other error during the QR scan method
System.out.println(RED + "\n Error occured during QR code scan , please run the program again " +
RESET + "\n");
e.printStackTrace(); // using it to print detailed information about an exception, including the stack
trace
System.exit(5); // The program ends
} // catch block ends


} // QRScan Method ends


// Method for the termination process


public static void terminateProgram()
{
final String RED = "\033[31m"; // Red text
final String GREEN = "\033[32m"; // Green text
final String BLUE = "\033[34m"; // Blue text
final String YELLOW = "\033[33m"; // Yellow text
final String RESET = "\033[0m"; // Reset to default color


try {
long endTime = System.currentTimeMillis();
double duration = (endTime - startTime) / 1000.0 ;
// creating a unique date so that the log files can be unique named
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMdd_HHmmss");
String uniqueDate = dateFormat.format(new Date());
String AbslogFilePath = "Object_Detect_Log_Swiftbot_"+uniqueDate+".txt";
String logFilePath = "/data/home/pi/object_detect_program/ "+AbslogFilePath; // inside SwiftBot
// Calculating the total Duration in seconds


// Creating log file , logFilePath as given at the beginning
FileWriter writer = new FileWriter(logFilePath); // Initiating writing log file


writer.write("Mode: " + selectedMode + "\n"); // from main method
writer.write("Duration: " + duration + " seconds\n");
writer.write("Objects Encountered: " + objectCount + "\n"); // from the classes
writer.write("Image File Path: " + imageFilePath + "\n");
writer.write("Log File Path: " + logFilePath + "\n");
writer.close();

// Termination UI ASCII art
System.out.println("\n");
System.out.println(" _____ _ _ _ ");
System.out.println("|_ _| (_) | | (_) ");
System.out.println(" | | ___ _ __ _ __ ___ _ _ __ __ _| |_ _ ___ _ __ ");
System.out.println(" | |/ _ \\ '__| '_ ` _ \\| | '_ \\ / _` | __| |/ _ \\| '_ \\ ");
System.out.println(" | | __/ | | | | | | | | (_| | | | | (_) | | | |");
System.out.println(" \\_/\\___|_| |_| |_| |_|_|_| |_|\\__,_|\\__|_|\\___/|_| |_|");
System.out.println("\n");
```

```
// Asking the user Termination options
System.out.println(RED + ">>> "+RESET+" Choose how do you want to terminate "+RED+ "<<<\n" + RESET);
System.out.println(RED + "--------------------------------------------------------------------------------
---" + RESET);
System.out.println(RED +"|"+RESET+" Press " + YELLOW + "'Y'" + RESET + " button to view the execution log
"+RED+" |"+RESET );
System.out.println(RED +"|"+RESET+" Press " + GREEN + "'X'" + RESET + " button to view only the log file path
"+RED+" |"+RESET );
System.out.println(RED + "--------------------------------------------------------------------------------
---" + RESET+"\n");


// Disable buttons before enabling
swiftBot.disableButton(Button.Y);
swiftBot.disableButton(Button.X);


swiftBot.enableButton(Button.Y, () -> { // when user presses Y
swiftBot.disableButton(Button.Y);


swiftBot.fillUnderlights(new int[] { 255, 165, 0 }); // Orange light
// Execution log display
System.out.println(BLUE + "--------------------------------------------------------------------------------
------------" + RESET);
System.out.println(BLUE + "| Mode of operation : " + GREEN + selectedMode + BLUE + " | " + RESET);
System.out.println(BLUE + "| Duration of operation: " + GREEN + duration + " Seconds" + BLUE + " | " +
RESET);
System.out.println(BLUE + "| Objects Encountered : " + GREEN + objectCount + BLUE + " | " + RESET);
System.out.println(BLUE + "| Image file path : " + GREEN + imageFilePath + BLUE + " | " + RESET);
System.out.println(BLUE + "--------------------------------------------------------------------------------
------------\n" + RESET);
swiftBot.disableUnderlights();
thankText();
System.exit(0);
});


swiftBot.enableButton(Button.X, () -> { // when user presses X
swiftBot.disableButton(Button.X);


swiftBot.fillUnderlights(new int[] { 255, 255, 0 }); // Yellow light
// Log file path display
System.out.println(BLUE + "--------------------------------------------------------------------------------
------------------------------------------------" + RESET);
System.out.println(BLUE + "| >>> " + GREEN + "Execution log file path : " + logFilePath + BLUE + " |" + RESET);
System.out.println(BLUE + "--------------------------------------------------------------------------------
------------------------------------------------\n" + RESET);
swiftBot.disableUnderlights();
thankText();
System.exit(0);
});


} // try block ends
```

```
catch (IOException e)
{
e.printStackTrace();
System.out.println(RED + "Error handling the termination, please try again\n" + RESET);
} // catch block ends
} // Termination ends
// Method to adjust position based on distance
// From the speed testing I did , at 35% speed of Swiftbot
// The velocity is 11.333 cm/s , so we are using the velocity formula
// distance = velocity x time to adjust position ( Maintaining the buffer zone for curious mode
public static void AdjustPosition(double distance) {
try
{
System.out.println(" \n >>> Adjusting position <<< ");
swiftBot.fillUnderlights(new int[] { 0, 255, 0 }); // green
if(distance>100) // when the object is too far , the bot moves forward
{
swiftBot.move( 50, 50, 2000 );
}
else if (distance > 31) // if the bot is far from buffer zone
{
double s = distance-30;
double t = s/(11.33333); // using the formula of velocity from test data
int time = (int) (t*1000); // converting the double to int as move method requires integer as time
swiftBot.move( 35, 35, time ); // move forward to make it inside buffer
}
else if (distance < 29) // if the bot is inside buffer zone
{
double s = 30-distance;
double t = s/(11.33333);
int time = (int) (t*1000);
swiftBot.move( -35, -35, time ); // move backward to make it inside buffer
}
swiftBot.disableUnderlights();
} // try block ends
catch (Exception e)
{
System.out.println("Error adjusting position");
}
} // Adjust position ends


// Method to display Welcome UI
public static void displayWelcomeScreen() {
System.out.println("\n");
// Object Detect ASCII art - with your preferred Detect style
System.out.println(" ____ _ _ _ _____ _ _ ");
System.out.println(" / __ \\| | (_) | | | __ \\ | | | | ");
System.out.println(" | | | | |__ _ ___ ___| |_ | | | _____| |_ ___ ___| |_ ");
System.out.println(" | | | | '_ \\| |/ _ \\/ __| __| | | |/ _ \\ __/ _ \\/ __| __|");
System.out.println(" | |__| | | | |_) | | __/ (__| |_ | |__/ __/ || __/ (__| |_ ");
System.out.println(" \\____/|_|__/| |\\___|\\___|\\__| |_____/\\___|\\__\\___|\\___|\\__|");
System.out.println(" _/ | ");
System.out.println(" |__/ ");
System.out.println("\n");
```

```
// Welcome message in blue box
final String CYAN = "\033[36m"; // Cyan text
final String BG_CYAN = "\033[46m"; // CYAN TEXT
final String RESET = "\033[0m";// background reset

System.out.println(BG_CYAN+" *** Welcome to the Object Detect Program by SwiftBot :D *** "+RESET);
System.out.println("\n");

// Button layout ASCII art using standard characters
System.out.println(" ------ ");
System.out.println(" _/ \\_ ");
System.out.println(" |Y B| ");
System.out.println(" _| X A |_ ");
System.out.println(" / \\ ");
System.out.println("| | ");
System.out.println(" \\ / ");
System.out.println(" |_ _| ");
System.out.println(" | | ");
System.out.println(" \\__ __/ ");
System.out.println(" ------ \n");

// Button labels in boxes using standard ASCII
System.out.println(" +--------+ +--------+");
System.out.println(" | Y | | B |");
System.out.println(" +--------+ +--------+");
System.out.println(" +--------+ +--------+");
System.out.println(" | X | | A |");
System.out.println(" +--------+ +--------+\n");


// Bottom description
System.out.println(CYAN+"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"+RESET);
System.out.println("++ The outline ( top view ) shows the approximate positions of ++");
System.out.println("+++ the four Buttons X , Y , A , B of the bot +++");
System.out.println(CYAN+"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"+RESET);


System.out.println("\n\n");
} // Welcome screen ends

// Method to display Thank you ASCII art
public static void thankText()
{

System.out.println("\n");
System.out.println(" _____ _ _ __ __ ");
System.out.println("|_ _| | | | \\ \\ / / ");
System.out.println(" | | | |__ __ _ _ __ | | __ \\ V /___ _ _ ");
System.out.println(" | | | '_ \\ / _` | '_ \\| |/ / \\ // _ \\| | | |");
System.out.println(" | | | | | (_| | | | | < | | (_) | |_| |");
System.out.println(" \\_/ |_| |_|\\__,_|_| |_|_|\\_\\ \\_/\\___/ \\__,_|");
System.out.println("\n");

} // Thank text ends

} // main class ends
```

Mode Super Class (Parent Class for the modes)

```
package object_detect_program;

import java.util.Random;
import java.text.SimpleDateFormat;
import java.util.Date;
import swiftbot.SwiftBotAPI;

// this is the parent/ super class [4]
abstract public class Mode {
// inheritance being used in Childs / sub classes
// we are inheriting swiftBot api , imageFilePath , changeDirection method in the child classes
// so that we do not need to repeat the code structure inside the classses over and over again
protected SwiftBotAPI swiftBot;
protected String imageFilePath;

public Mode(SwiftBotAPI swiftBot, String imageFilePath)
{
this.swiftBot = swiftBot;
this.imageFilePath = imageFilePath;
}

// creating a unique date so that the image files can be unique named
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMdd_HHmmss");
String uniqueDate = dateFormat.format(new Date());
abstract void run(); // this method is a template for the child classes
public void ChangeDirection()
{
try {
// It will randomly pick between Right or Left movement


Random random = new Random();
int num = random.nextInt(2) + 1;
if (num == 1)
{
swiftBot.move(80, 10, 3000); // Move right
swiftBot.move(35, 35, 2000); // Move forward
}
else {
swiftBot.move(10, 80, 3000); // Move left
swiftBot.move(35, 35, 2000); // Move forward
}
} // Try ends
catch (Exception e) {
System.out.println("Error changing direction");
}
} // Change direction ends
}
```

Curious mode Sub-Class

```
package object_detect_program;


import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
import swiftbot.SwiftBotAPI;
import swiftbot.Button;
import swiftbot.ImageSize;


// this is the child class of Mode for Curious mode that are being called


public class CuriousMode extends Mode {


public CuriousMode(SwiftBotAPI swiftBot, String imageFilePath) {
super(swiftBot, imageFilePath);
}


private static final String CYAN = "\u001B[36m"; // Cyan text
private static final String RED = "\u001B[31m"; // Red text
private static final String GREEN = "\033[32m";
private static final String BG_CYAN = "\033[46m"; // CBG YAN
private static final String BG_BLUE = "\u001B[44m"; // Blue background
private static final String RESET = "\u001B[0m"; // Reset color


// using private as access modifier , as a part of better encapsulation
                    [3]
public void run() { // method to run the functions of curious mode as the concept of Polymorphism
try {
displayWelcome();
WanderAround(); // calling the methods from the class


while (!ObjectDetect_Main.terminateCheck)
{
double distance = swiftBot.useUltrasound(); // keep storing and displaying distance
System.out.println(CYAN + "\n ==== "+RESET+" Current distance is " + distance + " cm "+CYAN +" ==== " +
RESET);


/* checking the bufferZone , instead of exactly 30 cm , for better execution of the
program , its modified to between 29 and 31 */
if (distance >= 29 && distance <= 31) // BufferZone
{
// if the object is inside the buffer zone , it should declare :
System.out.println(CYAN + "\n >>>> "+RESET+" Object Detected ! "+CYAN +" <<<<" + RESET);
```

```
for (int i = 0; i < 3; i++)
{ // For loop used for proper Blinking of the underlights as Green
swiftBot.fillUnderlights(new int[] { 0, 255, 0 });
Thread.sleep(300);
swiftBot.disableUnderlights();
}

ObjectDetect_Main.objectCount++; // Counting the object

System.out.println(CYAN + "\n |><| "+RESET+" Taking an image of the object "+CYAN +" |><|" + RESET+ "\n");
BufferedImage image = swiftBot.takeStill(ImageSize.SQUARE_720x720);
ImageIO.write(image, "png", new File(imageFilePath + "object_" + ObjectDetect_Main.objectCount+"_"
+uniqueDate + ".png"));

System.out.println(CYAN+"---------------------------------------"+RESET);
System.out.println("| Do you want to quit ? |");
System.out.println("| Please press "+ GREEN +"'X'"+RESET+" button to exit |");
System.out.println(CYAN+"---------------------------------------\n"+RESET);
long startTime = System.currentTimeMillis();
boolean[] quitPressed = { false }; // boolean to store button press by the user

swiftBot.disableButton(Button.X);
swiftBot.enableButton(Button.X, () -> {
swiftBot.disableButton(Button.X);
quitPressed[0] = true; // means button is pressed
ObjectDetect_Main.terminateCheck = true;
});

while (System.currentTimeMillis() - startTime < 5000) // 5 seconds waiting for termination
{
// if the button is pressed then the program goes to termination
if (quitPressed[0])
{ // as user pressed X button , calling terminatation from main class
ObjectDetect_Main.terminateProgram();
return;
}
Thread.sleep(100); // to smoothen the program
} // termination while ends

Thread.sleep(1000); // waiting one more second
System.out.println(CYAN + "\n>>> "+ RESET+"Changing the direction "+ CYAN +" <<<" + RESET);
ChangeDirection(); // method calling for Changing direction for new object
} // Buffer If ends

else if (distance != swiftBot.useUltrasound())
// if the object is moved then it should adjust position
{
ObjectDetect_Main.AdjustPosition(distance); // calling the method adjust position
}
} // main while loop ends
} // try block ends
catch (Exception e)
{
e.printStackTrace();
System.out.println(RED + "\n Error initiating the Curious Mode " + RESET);
} // catch ends
} // run ends
```

```
// method for wandering around of curious mode
public void WanderAround() {
try {
int[] rgb = { 0, 0, 255 }; // Blue
swiftBot.fillUnderlights(rgb); // underlights as blue
System.out.println(BG_BLUE + ">>> Hi ! I am wandering around for an object <<<" + RESET);
double distance = swiftBot.useUltrasound();
swiftBot.move(40, 40, 3000);
// Forward for 3 seconds at 40% speed

if (distance >= 29 && distance <= 31) // if the object is already inside bufferzone

{
swiftBot.disableUnderlights();
swiftBot.move(0, 0, 1000); // stops
}
else {
// Forward for 2 seconds at 40% speed
swiftBot.move(40, 40, 2000);

// Right for 3 seconds at good speed
swiftBot.move(80, 10, 3000);

// Forward for 3 seconds at 40% speed
swiftBot.move(40, 40, 3000);

// Right for 3 seconds at good speed
swiftBot.move(90, 10, 3000);

// Keep moving forward 2s at 35% speed
swiftBot.move(35, 35, 2000);
swiftBot.disableUnderlights();
// Blue light turned off after the Wandering around

}
}
catch (Exception e) {
System.out.println(RED + "Error in wandering movement" + RESET);
}
} // Wander around ends
private void displayWelcome()
{
System.out.println("\n");
System.out.println(" ____ _ __ __ _ ");
System.out.println("/ __| (_) | \\/ | | | ");
System.out.println("| | | _ _ _ __ _ ___ _ _ ___| \\ / | ___ __| | ___ ");
System.out.println("| | | | | | '__| |/ _ \\| | | / __| |\\/| |/ _ \\ / _` |/ _ \\");
System.out.println("| |___| |_| | | | | (_) | |_| \\__ \\ | | | (_) | (_| | __/");
System.out.println("\\____/\\__,_|_| |_|\\___/ \\__,_|___/_| |_|\\___/ \\__,_|\\___|");
System.out.println("\n");
System.out.println(BG_CYAN + "*** Welcome to the Curious Mode ***" + RESET);
System.out.println("\n");
} // display welcome ends
} // curious class ends
```

Scaredy mode Sub-Class

```
package object_detect_program;

import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
import swiftbot.SwiftBotAPI;
import swiftbot.Button;
import swiftbot.ImageSize;
//this is the child class of Mode for Curious mode that are being called

public class ScaredyMode extends Mode {
public ScaredyMode(SwiftBotAPI swiftBot, String imageFilePath) {
super(swiftBot, imageFilePath);
}
private static final String YELLOW = "\u001B[33m"; // Yellow/gold text
private static final String RED = "\u001B[31m"; // Red text
private static final String GREEN = "\u001B[32m"; // Green text for 'X' in quit message
private static final String BG_YELLOW = "\u001B[43m"; // Yellow background
private static final String BG_BLUE = "\u001B[44m"; // Blue background
private static final String RESET = "\u001B[0m"; // Reset color

public void run() {
try {
// same beginning as Curious mode
displayWelcome();
WanderAround();

while (!ObjectDetect_Main.terminateCheck)
{
double distance = swiftBot.useUltrasound();
System.out.println(YELLOW + "\n==== "+RESET+" Current distance is " + distance + " cm "+YELLOW +" ==== " +
RESET);

if (distance <= 50) // Checking if the object is within 50 cm zone
{
System.out.println(YELLOW + "\n>>>>"+RESET+" Object Detected ! "+YELLOW +" <<<<" + RESET);

swiftBot.fillUnderlights(new int[] { 255, 0, 0 }); // Red light
Thread.sleep(400);
swiftBot.disableUnderlights();

ObjectDetect_Main.objectCount++;

System.out.println(YELLOW + "\n|><|"+RESET+" Taking an image of the object "+YELLOW +" |><|" + RESET+
"\n");
BufferedImage image = swiftBot.takeStill(ImageSize.SQUARE_720x720);
ImageIO.write(image, "png", new File(imageFilePath + "\\object_" + ObjectDetect_Main.objectCount+"_"
+uniqueDate + ".png"));

swiftBot.fillUnderlights(new int[] { 255, 0, 0 }); // Red light
// here the swiftbot is turning 180 degree and move forward after detecting one object
swiftBot.move(-40, -40, 1500);
swiftBot.move(100, -100, 800); // 180 degree turn
swiftBot.move(40, 40, 3000);
swiftBot.disableUnderlights();
```

```
System.out.println(YELLOW+"----------------------------------------"+RESET);
System.out.println("| Do you want to quit ? |");
System.out.println("| Please press "+ GREEN +"'X'"+RESET+" button to exit |");
System.out.println(YELLOW+"----------------------------------------\n"+RESET);

long startTime = System.currentTimeMillis();
boolean[] quitPressed = { false };

swiftBot.disableButton(Button.X);
swiftBot.enableButton(Button.X, () -> {
quitPressed[0] = true;
ObjectDetect_Main.terminateCheck = true;
});

while (System.currentTimeMillis() - startTime < 5000)
{
if (quitPressed[0])
{
System.out.println(RED + "\n>>> Termination in progress . . ." + RESET);
ObjectDetect_Main.terminateProgram();
return;
}
Thread.sleep(100);
}

swiftBot.disableButton(Button.X);
} // 50 cm zone checking ends
else if (swiftBot.useUltrasound() > 50) // if the object has moved or out of range
{
System.out.println(YELLOW + "\n>>> "+ RESET+"Changing the direction "+ YELLOW +" <<<" + RESET);
ChangeDirection();
}
} // while loop ends
}
catch (Exception e)
{
e.printStackTrace();
System.out.println(RED + "\n Error initiating the Scaredy Mode " + RESET);
}
} // run method ends

// Method for Wandering Around
public void WanderAround() {
try {
int[] rgb = { 0, 0, 255 }; // Blue
swiftBot.fillUnderlights(rgb);
System.out.println(BG_BLUE + ">>> Hi ! I am wandering around for an object <<<" + RESET);
double distance = swiftBot.useUltrasound();
swiftBot.move(40, 40, 3000);
// Forward for 3 seconds at 40% speed


if (distance <= 50)


{
swiftBot.disableUnderlights();
swiftBot.move(0, 0, 1000);
}
```

```
else {
// Forward for 2 seconds at 40% speed
swiftBot.move(40, 40, 2000);


// Right for 3 seconds at good speed
swiftBot.move(80, 10, 3000);


// Forward for 3 seconds at 40% speed
swiftBot.move(40, 40, 3000);


// Right for 3 seconds at good speed
swiftBot.move(90, 10, 3000);


// Keep moving forward 2s at 35% speed
swiftBot.move(35, 35, 2000);


swiftBot.disableUnderlights();
// Blue light turned off after the Wandering around


}
} // try ends
catch (Exception e) {


System.out.println(RED + "Error in wandering movement" + RESET);
}

} // Wander around ends

private void displayWelcome() {
System.out.println("\n");
System.out.println(" _____ _ __ __ _ ");
System.out.println("/ ___| | | | \\/ | | | ");
System.out.println("\\ `--. ___ __ _ _ __ ___ __| |_ _ | \\ / | ___ __| | ___ ");
System.out.println(" `--. \\/ __/ _` | '__/ _ \\ / _` | | | | | |\\/| |/ _ \\ / _` |/ _ \\");
System.out.println("/\\__/ / (_| (_| | | | (_) | (_| | |_| | | | | | (_) | (_| | __/");
System.out.println("\\____/ \\___\\__,_|_| \\___/ \\__,_|\\__, | |_| |_|\\___/ \\__,_|\\___|");
System.out.println(" __/ | ");
System.out.println(" |___/ ");
System.out.println("\n");
System.out.println(BG_YELLOW + "*** Welcome to the Scaredy Mode ***" + RESET);
System.out.println("\n");
}
} // main class ends
```

Dubious mode Sub-Class

```
package object_detect_program;
import java.util.Random;
import swiftbot.SwiftBotAPI;
public class DubiousMode extends Mode {

public DubiousMode(SwiftBotAPI swiftBot, String imageFilePath) {
super(swiftBot, imageFilePath);  }

private static final String BLUE = "\u001B[34m"; // Blue text
private static final String RED = "\033[31m"; // Red text
private static final String CYAN = "\u001B[36m"; // Cyan text for arrows
private static final String YELLOW = "\u001B[33m"; // Yellow
private static final String BG_BLUE = "\u001B[44m"; // Blue background
private static final String RESET = "\u001B[0m"; // Reset color
public void run() {
displayWelcome();
try {
// Mode selection message
System.out.println(BLUE + "---------------------------------" + RESET);
System.out.println(BLUE + "| |" + RESET);
System.out.println(YELLOW + "| Randomly selecting between |" + RESET);
System.out.println(YELLOW + "| Curious Mode " + BLUE + "or" + YELLOW + " Scaredy Mode |" + RESET);
System.out.println(BLUE + "| |" + RESET);
System.out.println(BLUE + "---------------------------------" + RESET);
System.out.println("\n");

Random rand = new Random();
int mode = rand.nextInt(2) + 1; // Randomly select 1 or 2
if (mode == 1) {
System.out.println(BLUE + "-------------------------------------------------" + RESET);
System.out.println(BLUE + "|" + CYAN + " >>>>>> Executing Curious Mode . . ." + BLUE + " |" + RESET);
System.out.println(BLUE + "-------------------------------------------------" + RESET);
new CuriousMode(swiftBot,imageFilePath).run();
} else {
System.out.println(BLUE + "-------------------------------------------------" + RESET);
System.out.println(BLUE + "|" + CYAN + " >>>>>> Executing Scaredy Mode . . ." + BLUE + " |" + RESET);
System.out.println(BLUE + "-------------------------------------------------" + RESET);
new ScaredyMode(swiftBot,imageFilePath).run();
}
}
catch (Exception e)
{
System.out.println(RED + "\n Error initiating the Dubious Mode " + RESET);
}
} // run ends
private void displayWelcome()
{
System.out.println("\n");
System.out.println(" _____ _ _ __ __ _ ");
System.out.println("| __ \\ | | (_) | \\/ | | | ");
System.out.println("| | | |_ _| |__ _ ___ _ _ ___| \\ / | ___ __| | ___ ");
System.out.println("| | | | | | '_ \\| |/ _ \\| | / __| |\\/| |/ _ \\ / _` |/ _ \\");
System.out.println("| |__| | | |_| | | |_) | | (_) | |_| \\__ \\ | | | (_) | (_| | __/");
System.out.println("|_____/ \\__,_|_|.__/|_|\\___/ \\__,_|___/_| |_|\\___/ \\__,_|\\___|");
System.out.println("\n");
System.out.println(BG_BLUE + "*** Welcome to the Dubious Mode ***" + RESET);
System.out.println("\n");
}
}
```