

Name : _____ Roll No : _____

Paper IV (Robotics)
MSG (Computer Science) Semester-Ist 2022-23

INDEX				
NO	DATE	TITLE	PAGE NO	SIGN
1		Write a program to create a robot (i) With gear (ii) Without gear and move it forward, left, right.		
2		Write a program to create a robot with a two motor and move it forward, left, right.		
3		Write a program to do a square using a while loop, doing steps with a for loop.		
4		Write a program to create a robot with light sensors to follow a line.		
5		Write a program to create a robot that does a circle using 2 motors.		
6		Write a program to create a path following robot.		
7		Write a program to resist obstacles.		
8		Ultrasonic Sensor.		
9		Drag and Bot Simulator Demo.		
10		Pick-up Object using Drag and Bot Simulator.		
11		a) Write a program to create a robot to perform rectangular motion using gears. b) Write a program to create a robot to perform circular motion using gears.		
12		a) Write a program to create a robot to perform square motion without using gears. b) Write a program to create a robot to perform circular motion without using gears.		
13		Write a program to do a square using while or for loop, change direction based on condition and control motor.		

PRACTICAL 1A

AIM: WRITE A PROGRAM TO CREATE A ROBOT WITH GEAR AND MOVE IT FORWARD, LEFT, RIGHT.

Description:

1] NxtRobot() –

Class that represents a simulated NXT robot brick. Parts (e.g. motors, sensors) may be assembled into the robot to make it doing the desired job.

2] Gear() -

Creates a gear instance with right motor plugged into port A, left motor plugged into port B.

3] addPart(Part) -

Assembles the given part into the robot.

4] setSpeed(int) -

Sets the speed to the given value (arbitrary units).

5] forward() -

Starts the forward movement.

6] left() -

Starts to rotate left (center of rotation at middle of the wheel axes).

7] right() -

Starts to rotate right (center of rotation at middle of the wheel axes).

CODE:

```
import ch.aplu.robotsim.*;

public class Prac_1a {

    Prac_1a(){

        NxtRobot robot = new NxtRobot();

        Gear g = new Gear();

        robot.addPart(g);

        g.setSpeed(100);

        g.forward(500);

        g.left(250);

        g.forward(500);

        g.right(250);

        g.forward(500);

    }

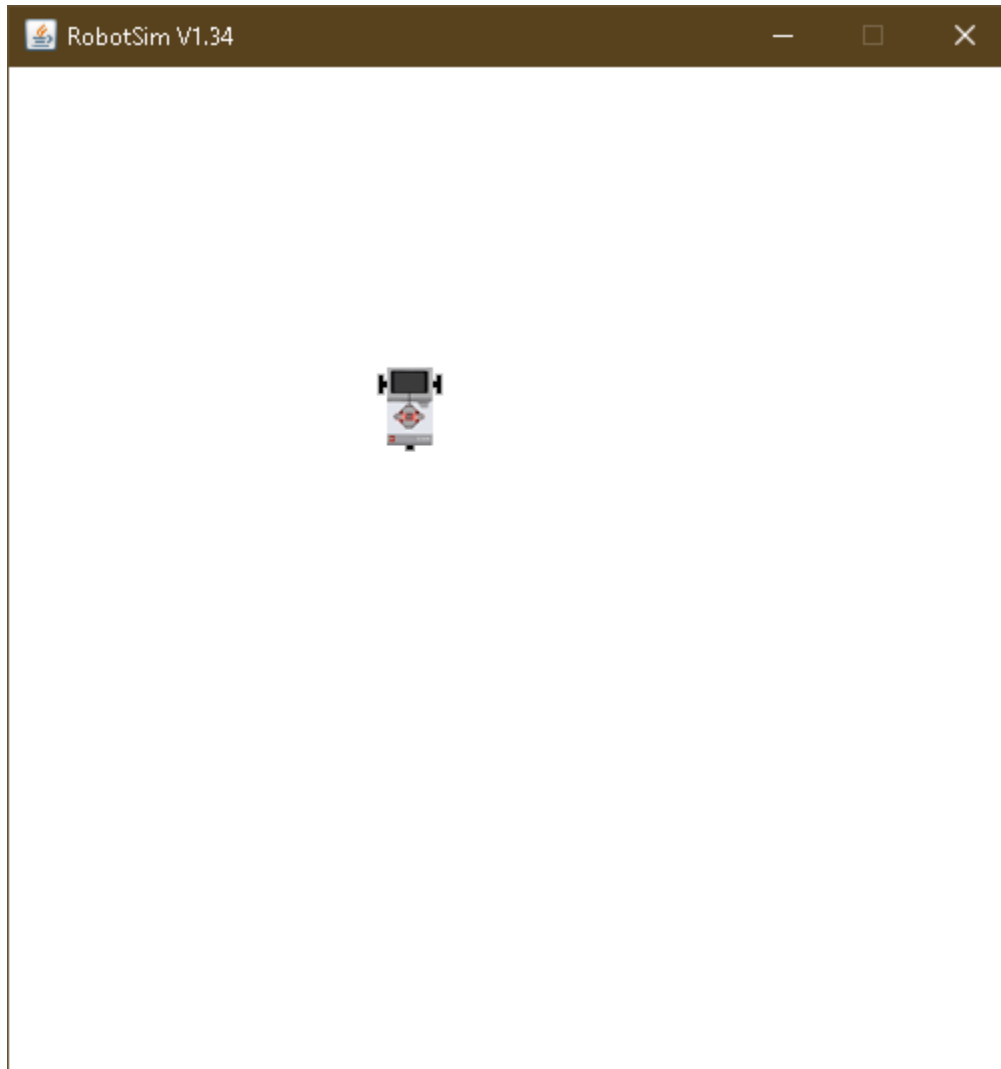
    public static void main (String[] args) {

        new Prac_1a();

    }

}
```

OUTPUT:



PRACTICAL 1B

AIM: WRITE A PROGRAM TO CREATE A ROBOT WITHOUT GEAR AND MOVE IT FORWARD, LEFT, RIGHT.

Description:

TurtleRobot() -

Creates a turtle robot instance.

CODE:

```
import ch.aplu.robotsim.*;

public class Prac_1b {

    Prac_1b(){

        TurtleRobot t = new TurtleRobot();

        t.forward(100);

        t.left(90);

        t.forward(100);

        t.right(90);

        t.forward(100);

    }

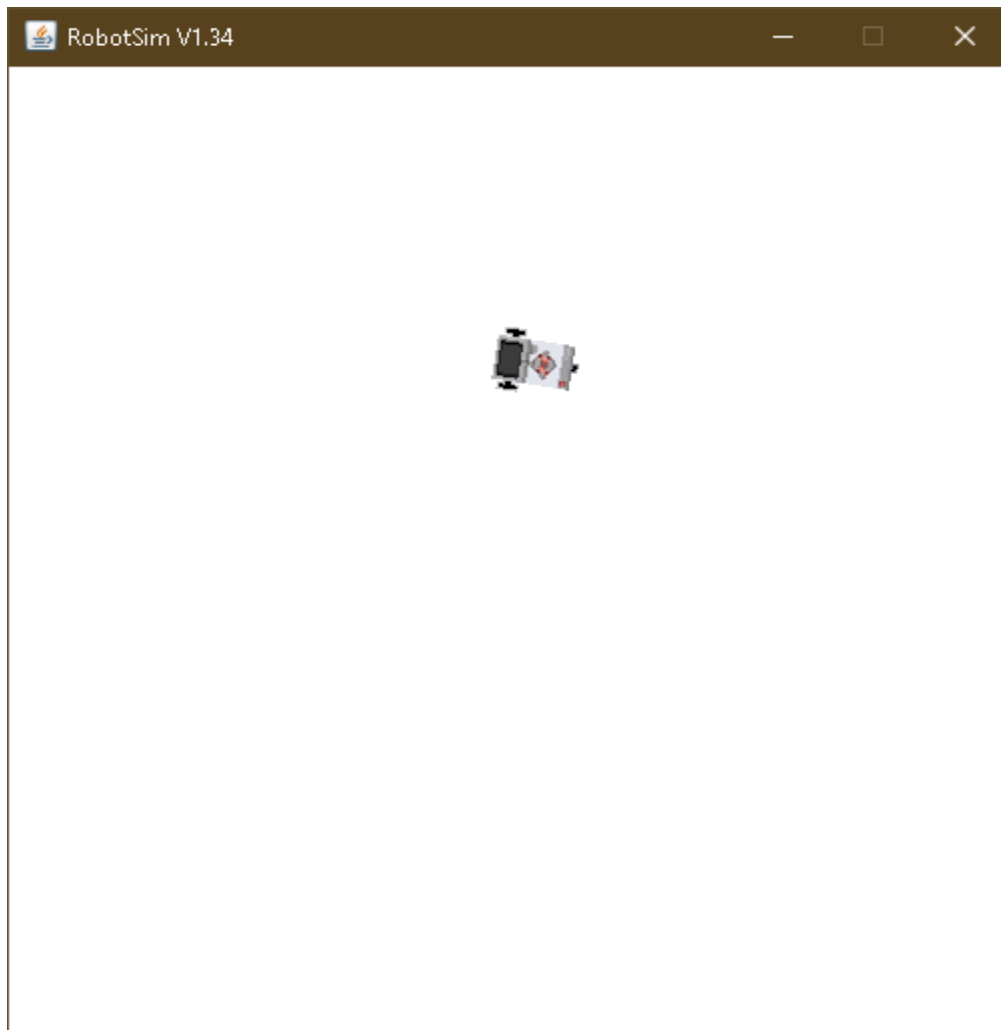
    public static void main (String[] args) {

        new Prac_1b();

    }

}
```

OUTPUT:



PRACTICAL 2

AIM: WRITE A PROGRAM TO CREATE A ROBOT WITH 2 MOTORS AND MOVE IT FORWARD, LEFT, RIGHT.

DESC:

1] Motor() -

Creates a motor instance that is plugged into given port.

2] Tools.delay() -

Suspends execution of the current thread for the given amount of time.

3] stop() –

Stops the rotation.

CODE:

```
import ch.aplu.robotsim.*;

public class Prac_2 {

    Prac_2(){

        NxtRobot r = new NxtRobot();

        Motor m1 = new Motor(MotorPort.A);

        Motor m2 = new Motor(MotorPort.B);

        r.addPart(m1);

        r.addPart(m2);

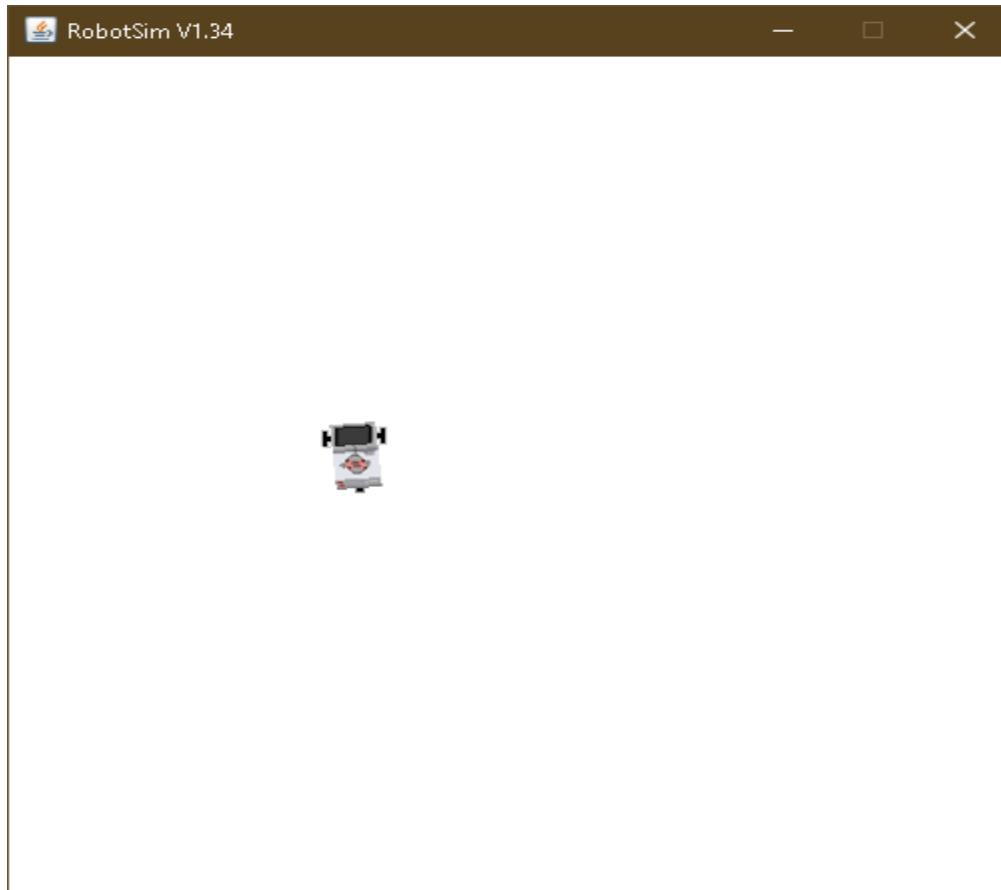
        m1.forward();

        Tools.delay(1090);

        m2.forward();
```

```
Tools.delay(1090);  
  
m1.stop();  
  
m2.forward();  
  
Tools.delay(1090);  
  
m1.forward();  
  
m1.stop();  
  
m2.stop();  
  
}  
  
public static void main(String args[]){  
  
    new Prac_2();  
  
}  
  
}
```

OUTPUT:



PRACTICAL 3

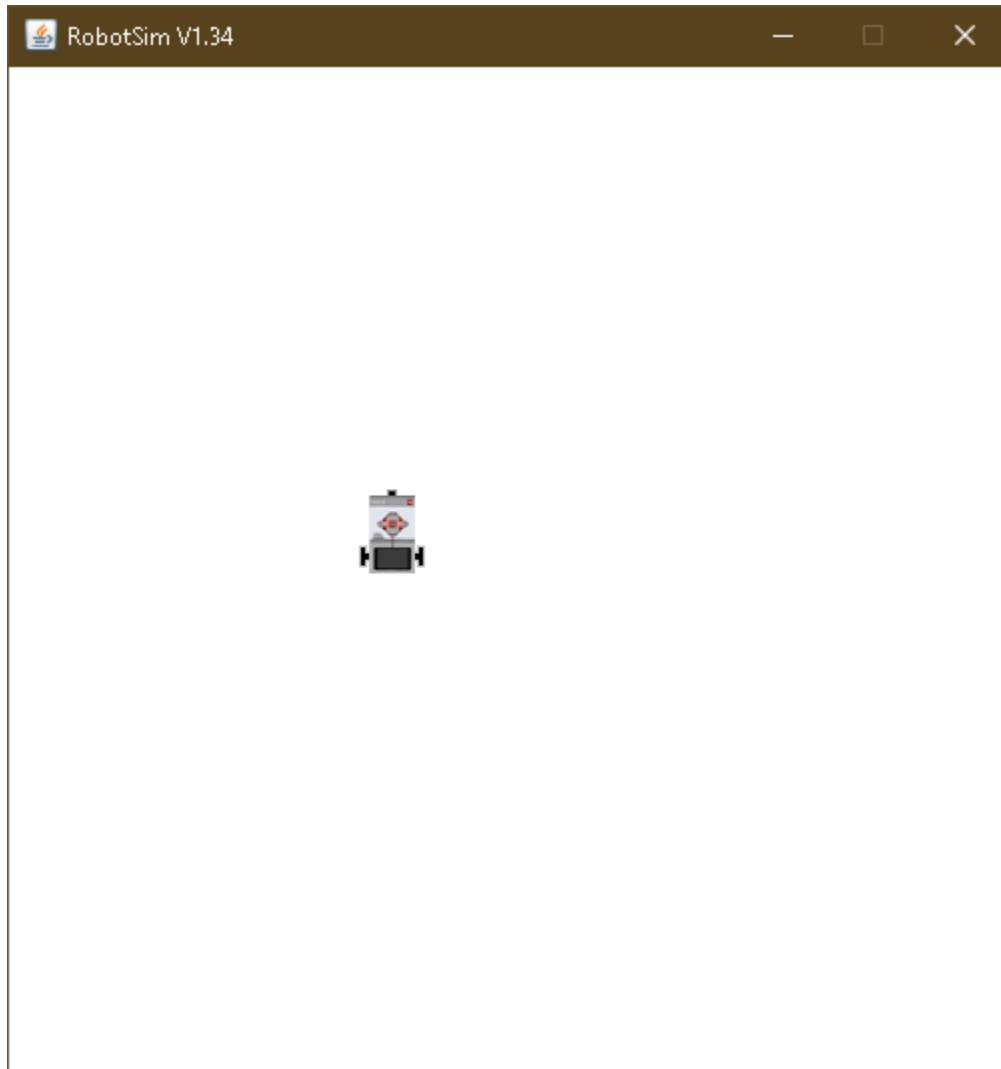
AIM: WRITE A PROGRAM TO DO A SQUARE USING A WHILE LOOP.

CODE:

```
import ch.aplu.robotsim.*;  
  
public class Prac_3 {
```

```
Prac_3(){  
  
    NxtRobot robot = new NxtRobot();  
  
    Gear g = new Gear();  
  
    robot.addPart(g);  
  
    g.setSpeed(100);  
  
    while (true){  
  
        g.forward(600);  
  
        g.left(280);  
  
    }  
}  
  
public static void main (String[] args) {  
  
    new Prac_3();  
  
}  
}
```

OUTPUT:



PRACTICAL 4

AIM: WRITE A PROGRAM TO CREATE A ROBOT WITH LIGHT SENSORS TO FOLLOW A LINE.

Description:

1] RobotContext() -

Creates a RobotContext instance.

2] `setStartPosition(int, int)` –

Sets the Nxt starting position (x-y-coordinates 0..500, origin at upper left).

3] `useBackground(String)` –

Use the given image as background (playground size 501 x 501).

4] `LegoRobot()` –

Creates a robot with its playground using defaults from `RobotContext`.

5] `LightSensor(SensorPort)` -

Creates a sensor instance pointing downwards connected to the given port.

6] `getValue()` –

For sensor ports 1, 2, 3, 4: returns the brightness of the background at the current location.

7] `leftArc()` –

Starts to move to the left on an arc with given radius.

8] `rightArc()` -

Starts to move to the right on an arc with given radius.

CODE:

```
import ch.aplu.robotsim.*;

public class Prac_4 {

    static {

        RobotContext.setStartPosition(32,495);

        RobotContext.useBackground("sprites/road.gif");

    }

    Prac_4(){

        LegoRobot r=new LegoRobot();
```

```
Gear g = new Gear();

LightSensor ls= new LightSensor(SensorPort.S3);

r.addPart(g);

r.addPart(ls);

g.forward();

g.setSpeed(50);

while(true){

    int v =ls.getValue();

    if(v < 100)

        g.forward();

    if(v > 350 && v<750)

        g.leftArc(0.005);

    if(v > 800)

        g.rightArc(0.005);

}

}

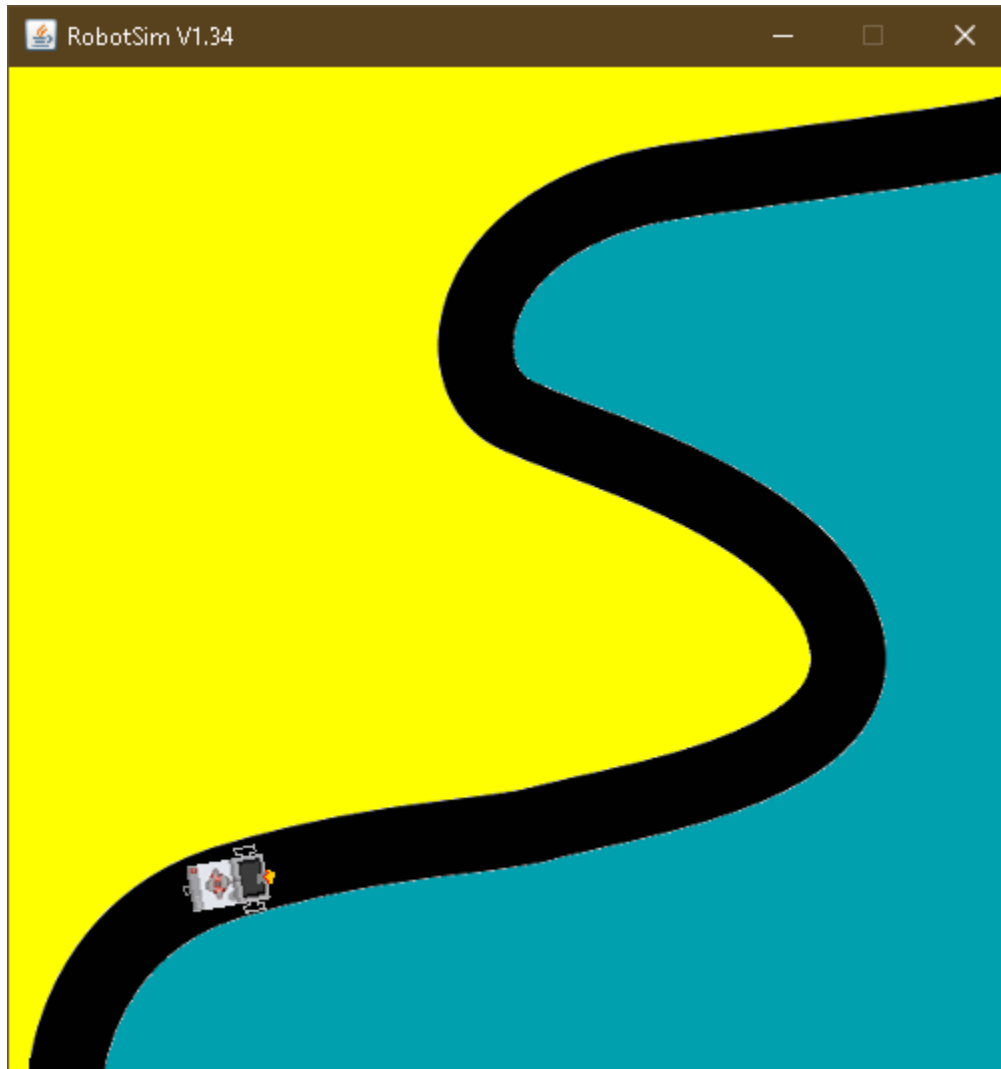
public static void main (String args[]){

    new Prac_4();

}

}
```

OUTPUT:



PRACTICAL 5

AIM: WRITE A PROGRAM TO CREATE A ROBOT THAT DOES A CIRCLE USING 2 MOTORS.

CODE:

```
import ch.aplu.robotsim.*;

public class Prac_5 {

    Prac_5() {

        NxtRobot r = new NxtRobot();

        Motor A = new Motor(MotorPort.A);

        Motor B = new Motor(MotorPort.B);

        r.addPart(B);

        r.addPart(A);

        A.setSpeed(100);

        B.setSpeed(100);

        A.forward();

        B.forward();

        while (true){

            Tools.delay(200);

            A.stop();

            Tools.delay(200);

            A.forward();

        }

    }

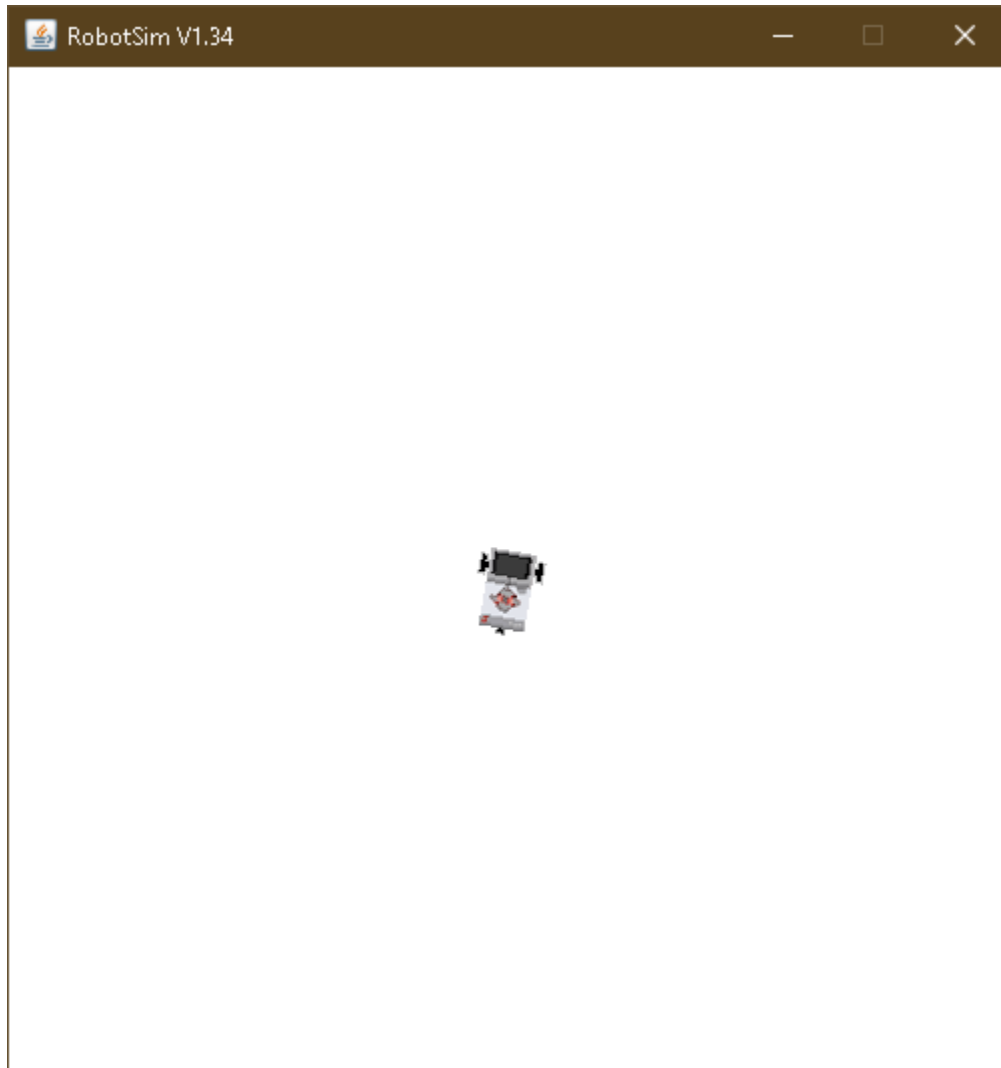
    public static void main(String arg[]) {

        new Prac_5();

    }

}
```

OUTPUT:



PRACTICAL 6

AIM: WRITE A PROGRAM TO CREATE A PATH FOLLOWING ROBOT.

Description:

1] `setStartDirection(double)` -

Sets the Nxt starting direction (zero to EAST).

CODE:

```
import ch.aplu.robotsim.*;

public class Prac_6 {

    Prac_6(){

        NxtRobot robot=new NxtRobot();

        Gear gear=new Gear();

        LightSensor ls1=new LightSensor(SensorPort.S1);

        LightSensor ls2=new LightSensor(SensorPort.S2);

        robot.addPart(gear);

        robot.addPart(ls1);

        robot.addPart(ls2);

        gear.forward();

        gear.setSpeed(100);

        while(true)

        {

            int rightValue=ls1.getValue();

            int leftValue=ls2.getValue();

            if(leftValue < 10)

                gear.rightArc(0.05);

            if(rightValue < 10)

                gear.leftArc(0.05);

            if(leftValue > 10 && rightValue > 10)

                gear.forward();

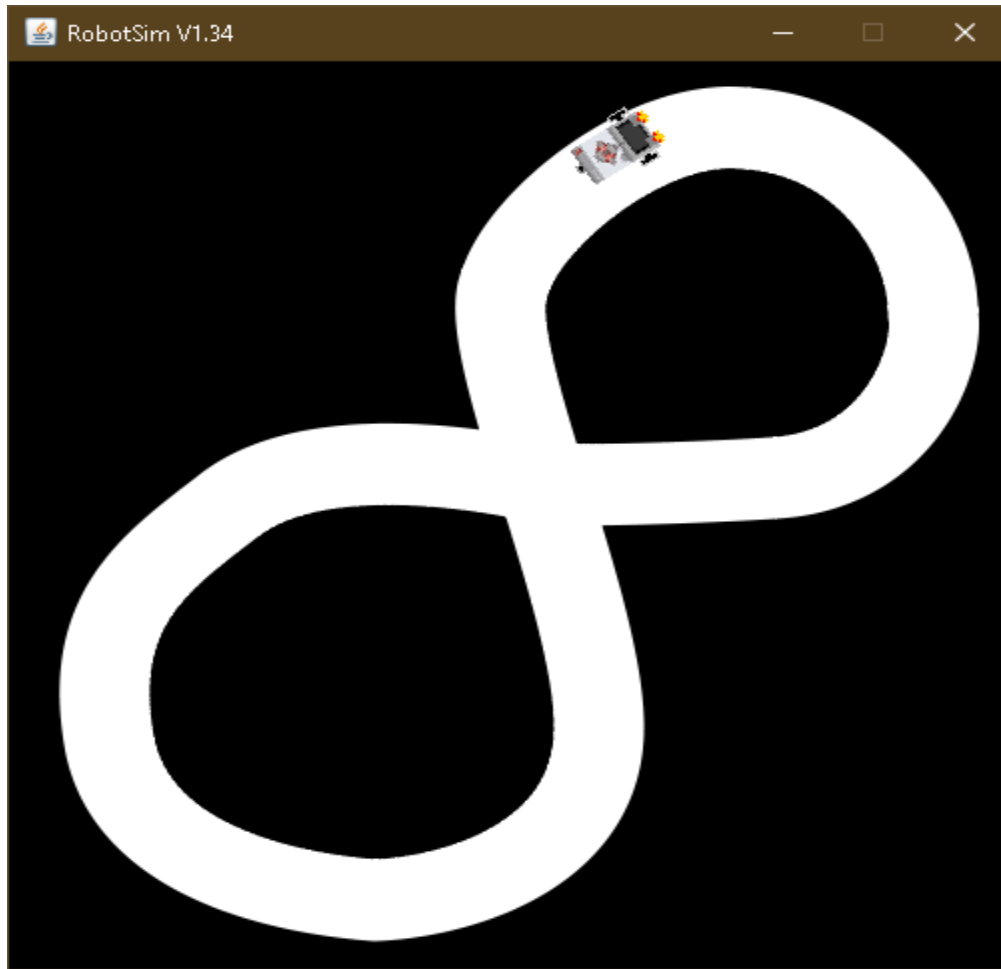
        }

    }

}
```

```
public static void main(String args[])  
{  
    new Prac_6();  
}  
  
static  
{  
    NxtContext.setStartPosition(267,232);  
    NxtContext.setStartDirection(-90);  
    NxtContext.useBackground("sprites/path.gif");  
}  
}
```

OUTPUT:



PRACTICAL 7

AIM: WRITE A PROGRAM TO RESIST OBSTACLES.

Description:

1] TouchSensor(SensorPort) -

Creates a sensor instance connected to the given port.

2] isPressed() –

Polls the touch sensor and returns true, if there is a collision with any of the collision obstacles.

3] backward() –

Starts moving backward and returns immediately.

4] useObstacle(Obstacle) –

Defines the given obstacle to be used as touch obstacle.

CODE:

```
import ch.aplu.robotsim.*;

public class Prac_7 {

    Prac_7(){

        LegoRobot r=new LegoRobot();

        Gear g = new Gear();

        TouchSensor t1= new TouchSensor(SensorPort.S1);

        TouchSensor t2 = new TouchSensor(SensorPort.S2);

        r.addPart(g);

        r.addPart(t1);

        r.addPart(t2);

        g.forward();

        g.setSpeed(50);

        while(true){

            Boolean b1 = t1.isPressed();

            Boolean b2 = t2.isPressed();

            if(b1 && b2){

                g.backward(150);
```

```
        g.right(400);

        g.forward();
    }

    if(b1){

        g.backward(150);

        g.left(200);

        g.forward();
    }

    if(b2){

        g.backward(150);

        g.right(200);

        g.forward();
    }
}

static {

    RobotContext.setStartPosition(100,250);

    RobotContext.useObstacle(RobotContext.channel);

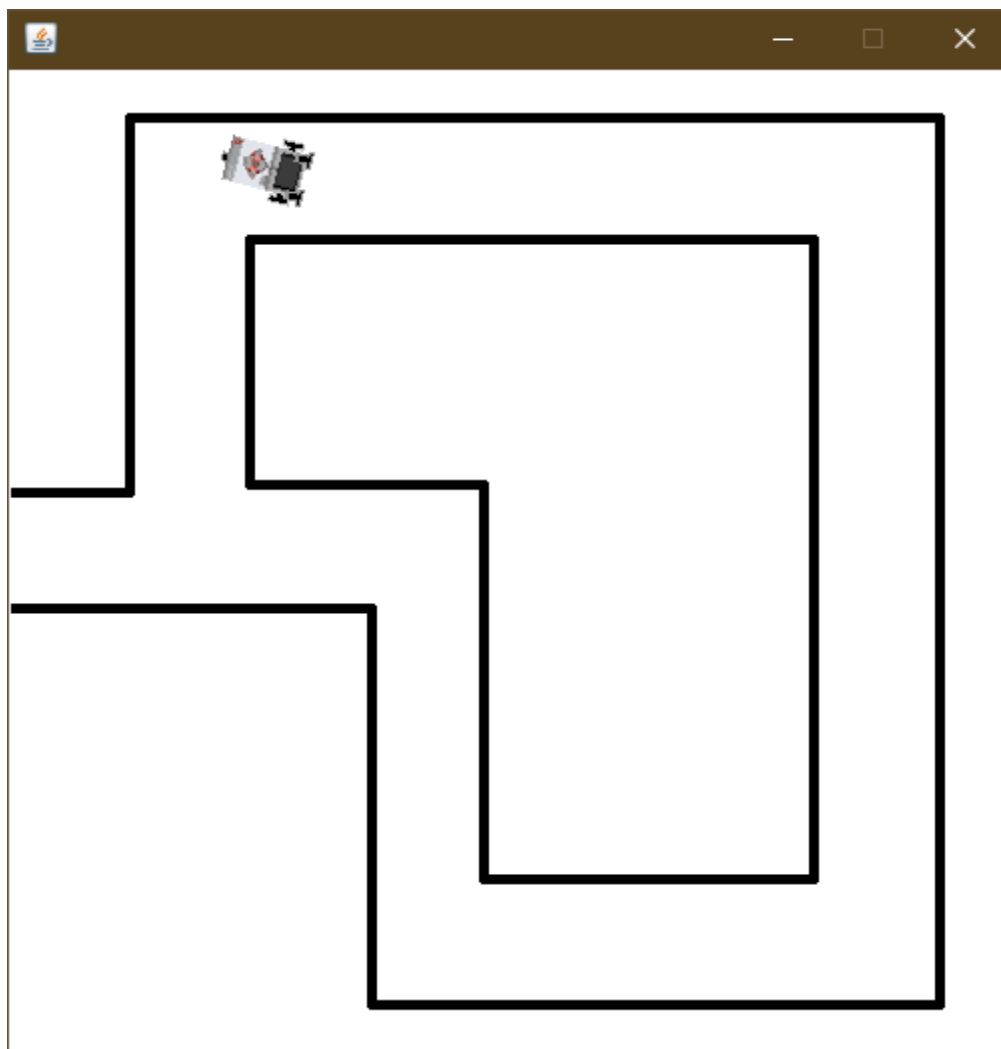
}

public static void main(String args[]){

    new Prac_7();

}
}
```

OUTPUT:



PRACTICAL 8

AIM: ULTRASONIC SENSOR.

DESC:

1] UltrasonicSensor(SensorPort) -

The port selection determines the position of the sensor and the direction of the beam axis.

2] setBeamAreaColor(Color) -

Sets the color of the beam area (two sector border lines and axis).

3] setProximityCircleColor(Color) –

Sets the color of the circle with center at sensor location and radius equals to the current distance value.

4] getDistance() –

Returns the distance to the nearest target object.

5] useTarget(String, Point[], int, int) –

Creates a target for the ultrasonic sensor using the given sprite image.

CODE:

```
import ch.aplu.robotsim.*;

import java.awt.Color;

import java.awt.Point;

public class Prac_8 {

    Prac_8() {

        LegoRobot robot = new LegoRobot();

        Gear gear = new Gear();

        robot.addPart(gear);

        UltrasonicSensor us = new UltrasonicSensor(SensorPort.S1);

        robot.addPart(us);

        us.setBeamAreaColor(Color.green);

        us.setProximityCircleColor(Color.lightGray);

        double arc = 0.5;

        gear.setSpeed(50);

        gear.rightArc(arc);

        boolean isRightArc = true;

        int oldDistance = 0;

        while (true)

        {

            Tools.delay(100);

            int distance = us.getDistance();

            if (distance == -1)

                continue;

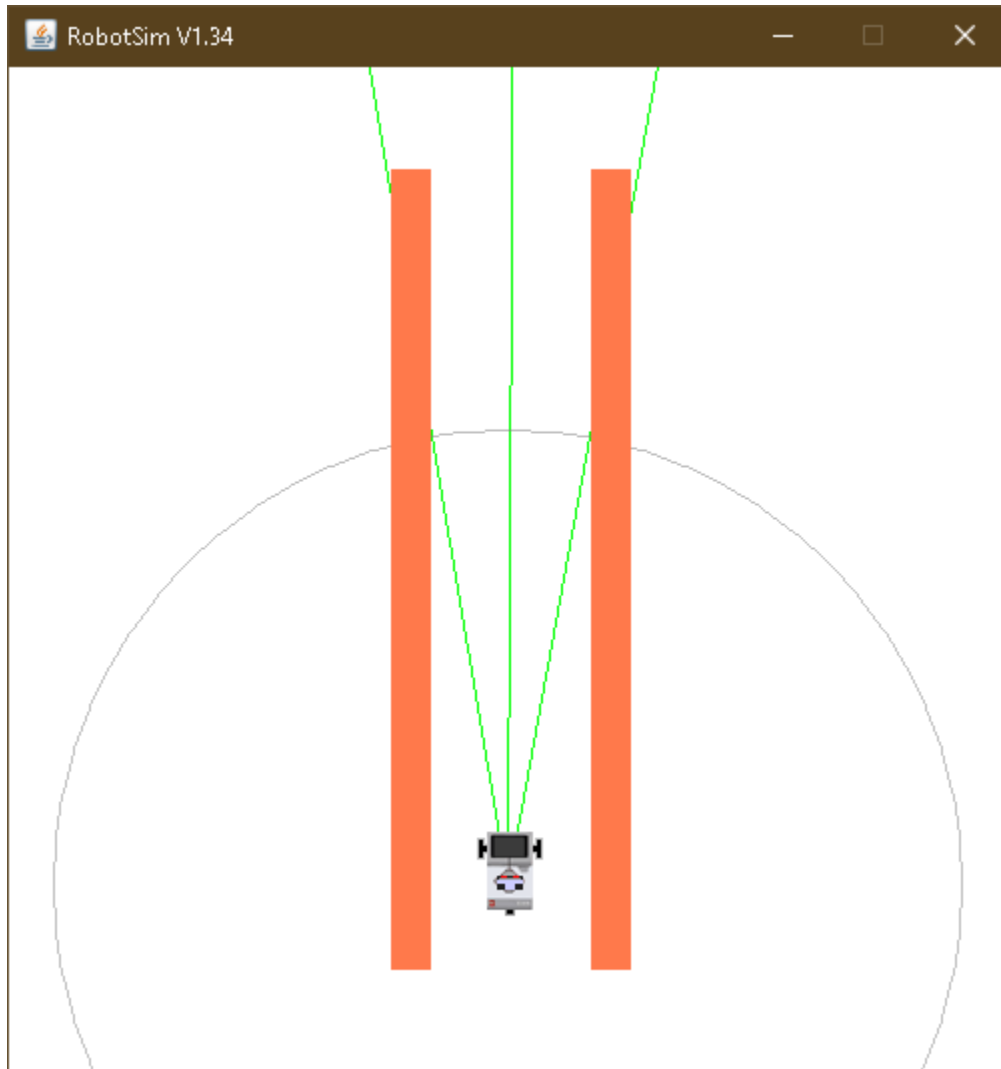
            if (distance < oldDistance)
```



```
{  
    if (isRightArc)  
    {  
        gear.leftArc(arc);  
        isRightArc = false;  
    }  
    else  
    {  
        gear.rightArc(arc);  
        isRightArc = true;  
    }  
}  
oldDistance = distance;  
}  
  
static{  
    Point[] mesh_bar =  
    {  
        new Point(10, 200), new Point(-10, 200),  
        new Point(-10, -200), new Point(10, -200)  
    };  
    RobotContext.useTarget("sprites/bar1.gif", mesh_bar, 200, 250);  
    RobotContext.useTarget("sprites/bar1.gif", mesh_bar, 300, 250);  
    RobotContext.setStartPosition(250, 460);  
}
```

```
public static void main(String[] args) {  
    new Prac_8();  
}  
}
```

OUTPUT:



Assignment 1(A):

Aim: Write a program to create a robot to perform rectangular motion using gears

Description:

1] NxtRobot() :

Class that represents a simulated NXT robot brick. Parts (e.g. motors, sensors) may be assembled into the robot to make it doing the desired job.

2] Gear() :

Creates a gear instance with right motor plugged into port A, left motor plugged into port B.

3] addPart() :

Assembles the given part into the robot.

4] setSpeed() :

Sets the speed to the given value (arbitrary units).

5] forward() :

Starts the forward movement for the given duration (in ms) and stops. Method returns at the end of the given duration.

6] left() :

Starts to rotate left (center of rotation at middle of the wheel axes). Method returns immediately, while the movement continues

Code:

```
import ch.aplu.robotsim.NxtRobot;
import ch.aplu.robotsim.Gear;
public class assignment1A {
    public assignment1A() {
```

```
NxtRobot r = new NxtRobot ();  
  
Gear g = new Gear();  
  
r.addPart (g);  
  
g.setSpeed (100);  
  
while (true){  
    g.forward (800);  
    g.left (280);  
}  
}  
  
public static void main (String [] args){  
    new assignment1A ();  
}  
}
```

Output:



Assignment 1(B):

Aim: Write a program to create a robot to perform circular motion using gears

Description:

1] rightArc() :

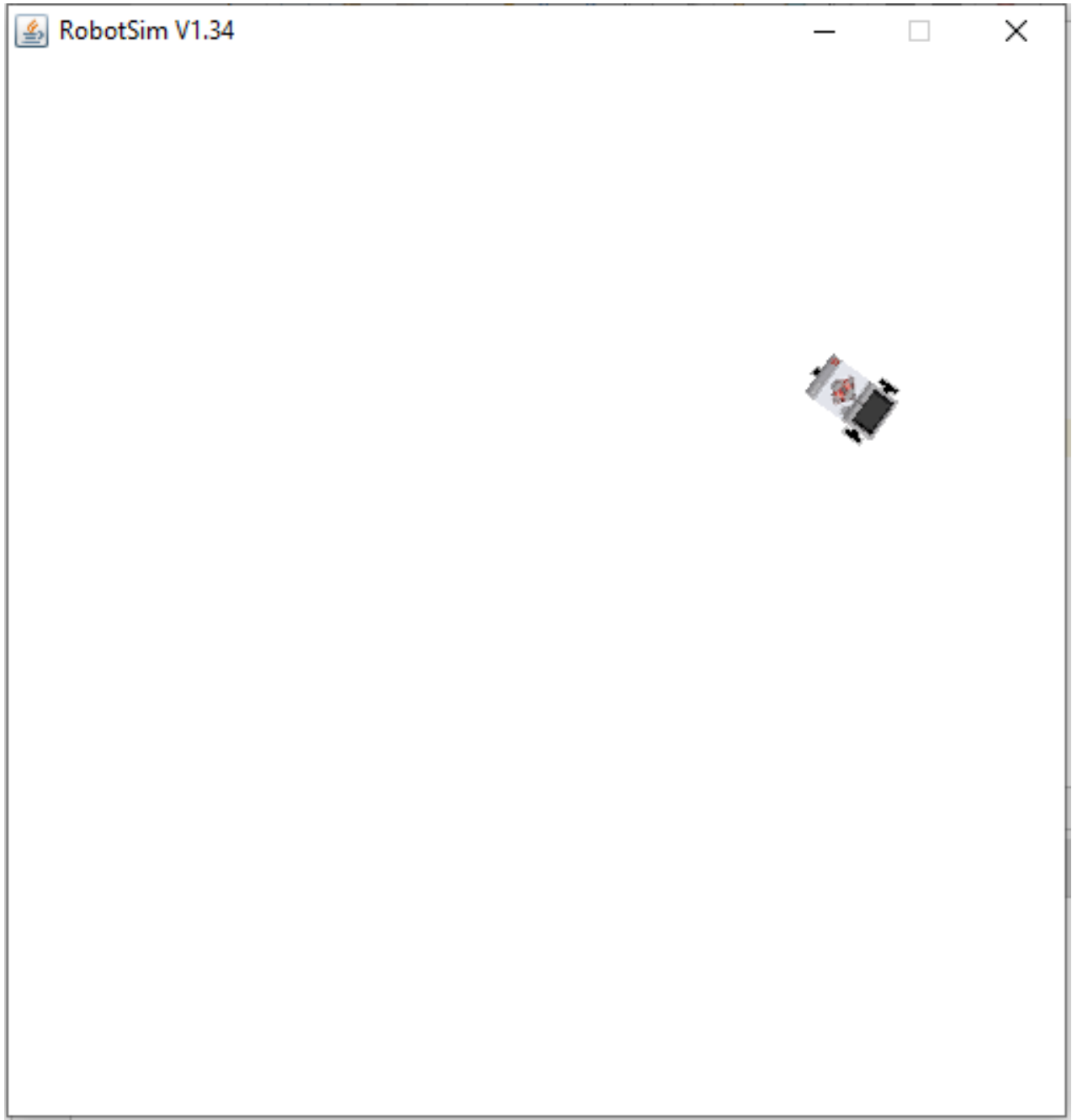
Starts to move to the right on arc with given radius. Method returns immediately, while the movement continues.

.

Code:

```
import ch.aplu.robotsim.NxtRobot;
import ch.aplu.robotsim.Gear;
public class assignment1B {
    public assignment1B () {
        NxtRobot r = new NxtRobot ();
        Gear g = new Gear ();
        r.addPart (g);
        g.setSpeed (100);
        while (true) {
            g.rightArc (0.5);
        }
    }
    public static void main (String [] args){
        new assignment1B ();
    }
}
```


Output



Assignment 2 (A):

Aim: Write a program to create robot to perform a square motion without using gear.

Code:

```
import ch.aplu.robotsim.*;

public class Assignment_2a {

    Assignment_2a () {

        TurtleRobot t = new TurtleRobot ();

        t.setTurtleSpeed (100);

        while (true){

            t.forward(200);

            t.left (90);

        }

    }

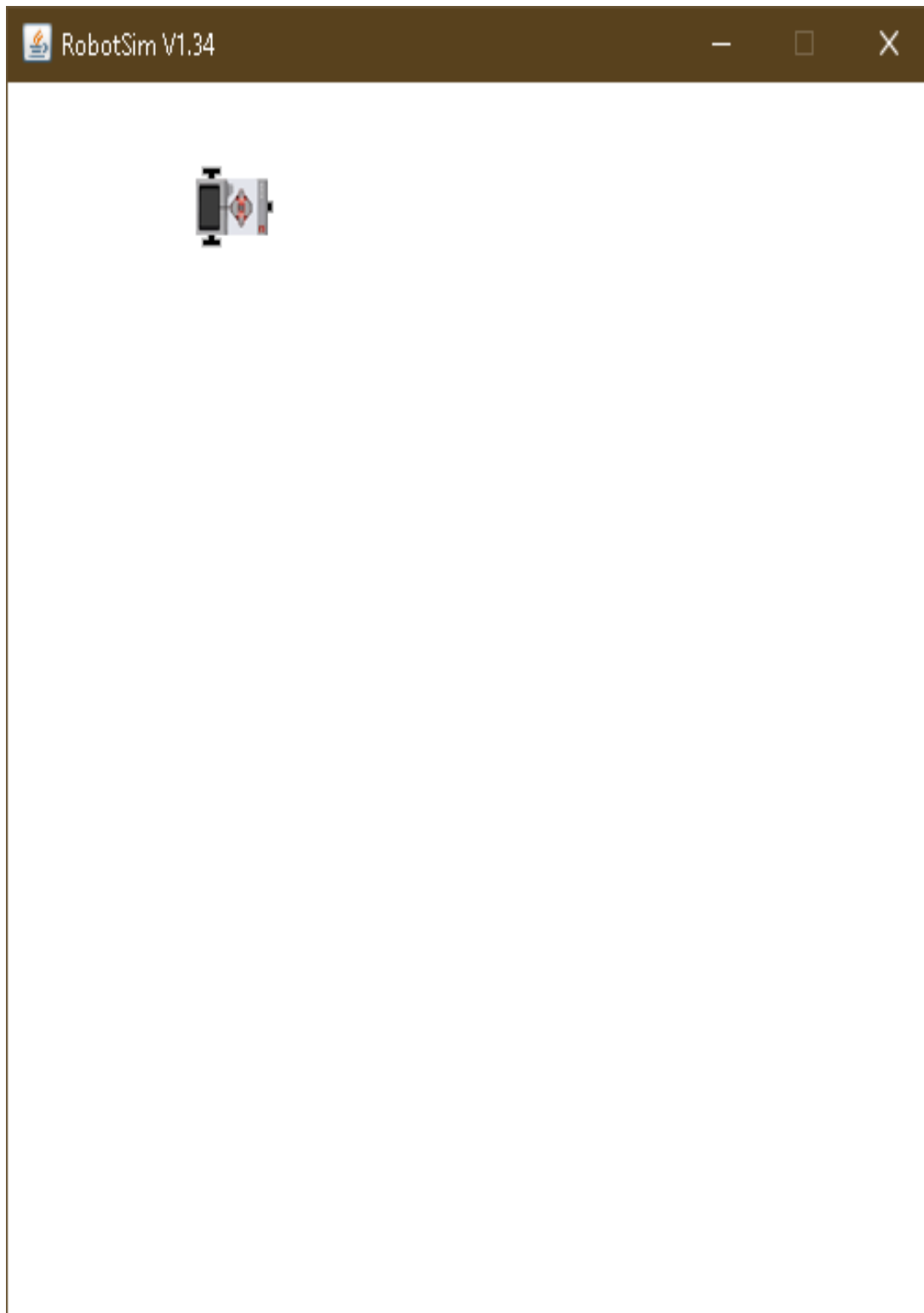
    public static void main (String [] args) {

        new Assignment_1a ();

    }

}
```

Output:



Assignment 2 (B):

Aim: Write a program to create robot to perform a circular motion without using gear.

Code:

```
import ch.aplu.robotsim.*;

public class Assignment_2b {

    Assignment_2b () {

        TurtleRobot t = new TurtleRobot ();

        t.setTurtleSpeed (100);

        while (true) {

            t.forward (2);

            t.left (2);

        }

    }

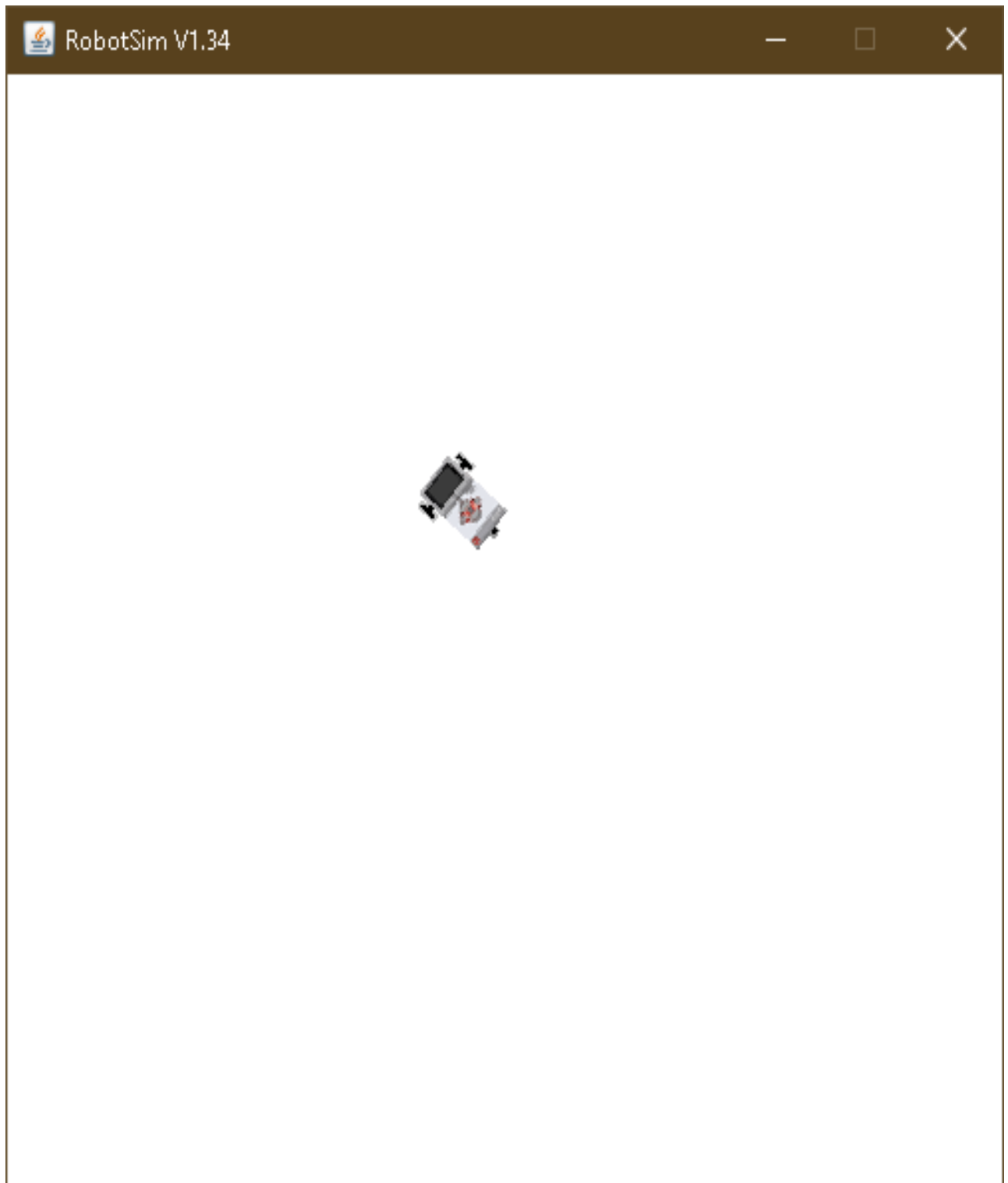
    public static void main (String [] args) {

        new Assignment_1b ();

    }

}
```

Output:



Assignment 3:

Aim: Write a program to do a square using while or for loop, change direction based on condition and control motor movement

Description:

1] Motor() :

Creates a motor instance that is plugged into given port.

2] Tools.delay() :

Suspends execution of the current thread for the given amount of time.

Code:

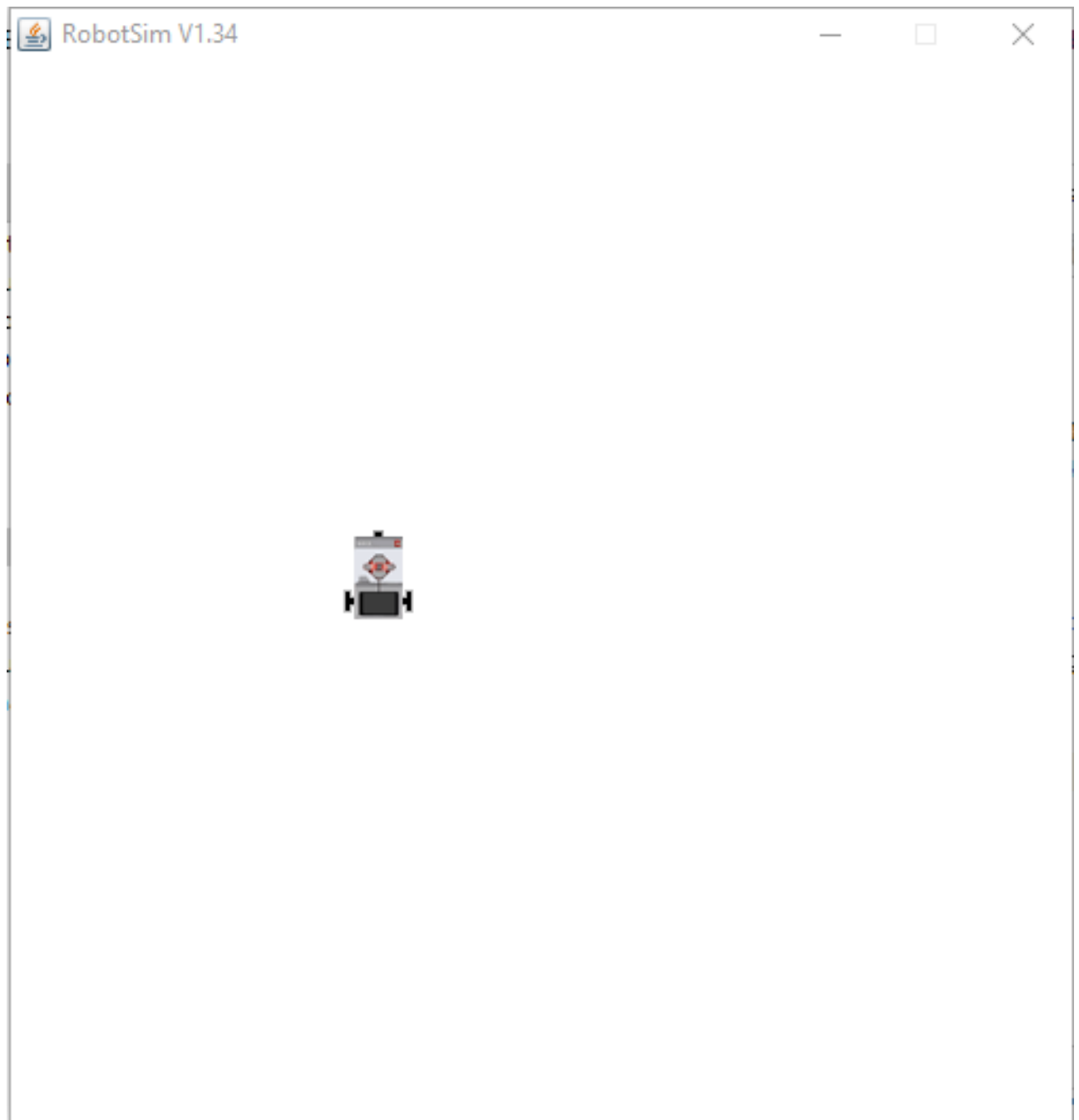
```
import ch.aplu.robotsim.*;
import java.util.*;

public class assignment2 {
    assignment2 () {
        Scanner sc = new Scanner (System.in);
        NxtRobot r = new NxtRobot ();
        Motor m1 = new Motor (MotorPort.A);
        Motor m2 = new Motor (MotorPort.B);
        r.addPart (m1);
        r.addPart (m2);
        System.out.println ("Enter 1 for left and 2 for right :");
        int direction = sc.nextInt ();
        switch (direction) {
            case 1:
                for (int i=0; i<4; i++){
                    m1.forward ();
                    Tools.delay (1090);
                    m2.forward ();

                    Tools.delay (1090);
                    m1.stop ();
```

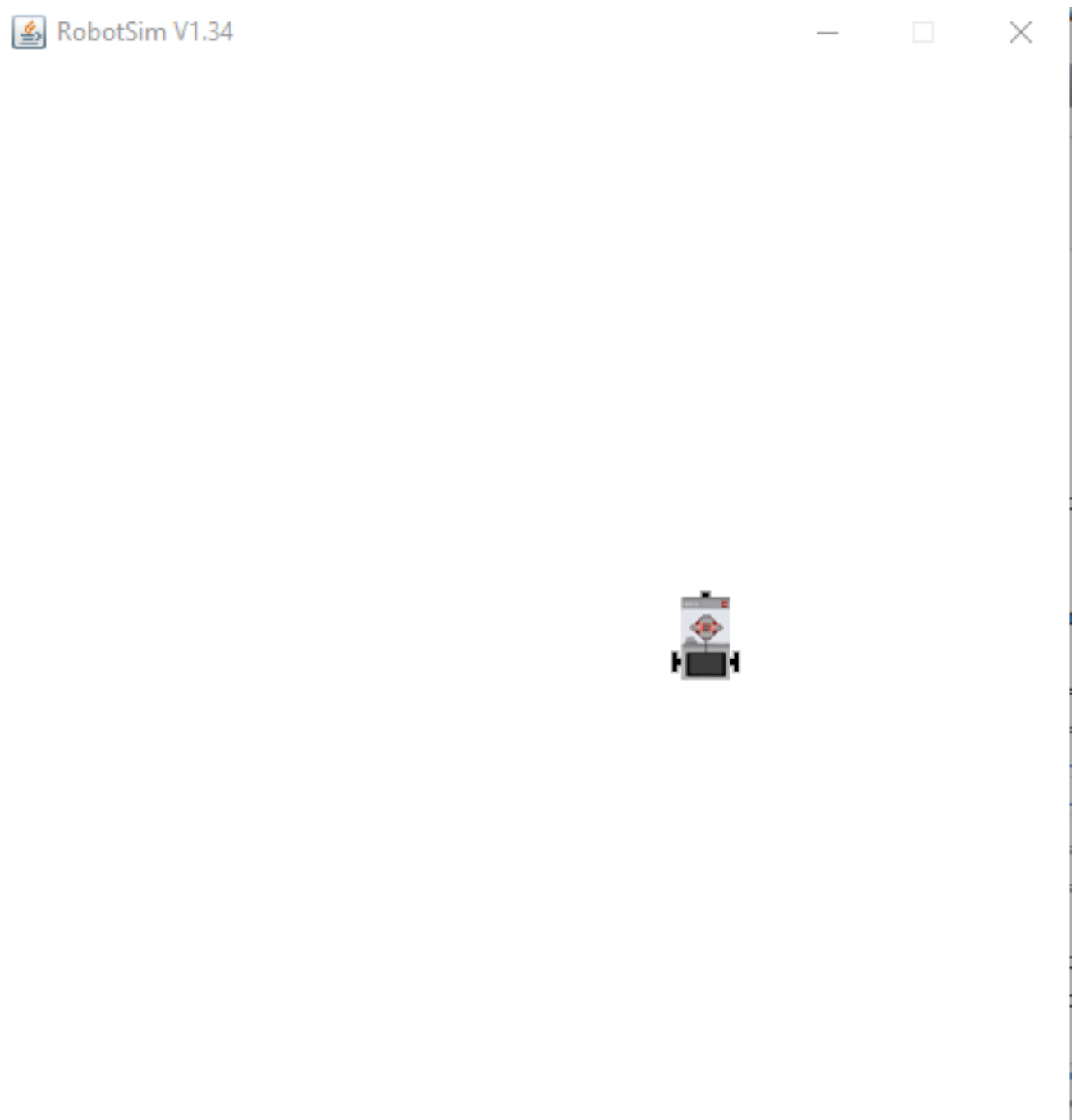
```
        m2.stop ();
    }
    break;
case 2:
    for (int i=0; i<4; i++){
        m2.forward ();
        Tools.delay (1090);
        m1.forward ();
        Tools.delay (1090);
        m1.stop ();
        m2.stop ();
    }
    break;
}
}
public static void main (String args[]){
    new assignment2 ();
}
}
```

Output:




```
Output - RoboticsAssignment (run) ×
run:
Enter 1 for left and 2 for right:
2
|
```

Activate Windows
Go to Settings to activate Windows.



Eram Khan
Roll no - 524

