Robotics Practical

**Practical 1**

**Write a program to create a robot (i) With gear (ii) Without gear and move it forward, left, right**

(i) With gear

Code:

**import** ch.aplu.robotsim.NxtRobot;

**import** ch.aplu.robotsim.Gear;

**public** **class** MoveWithGear {

**public** MoveWithGear()

{

NxtRobot robot = **new** NxtRobot();

Gear g= **new** Gear();

robot.addPart(g);

g.setSpeed(100);

g.forward(1000);

g.left(300);

g.forward(1000);

g.right(300);

g.forward(1000);

robot.exit();

}

**public** **static** **void** main(String arg[])

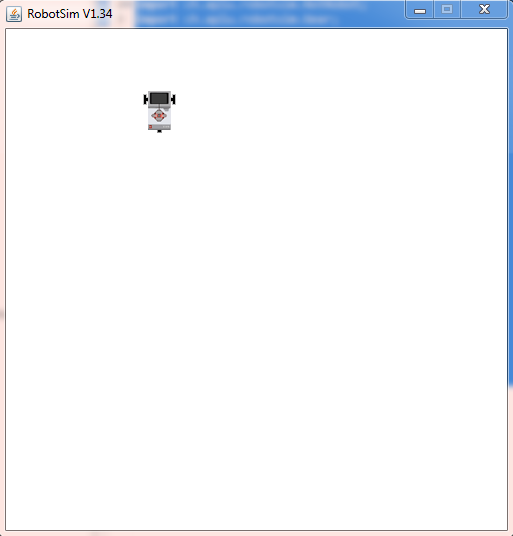
{

**new** MoveWithGear();

}

}

Output:



(ii) Without gear

Code:

**import** ch.aplu.robotsim.TurtleRobot;

**public** **class** MoveWithoutGear {

**public** MoveWithoutGear()

{

TurtleRobot robot = **new** TurtleRobot();

robot.forward(50);

robot.left(45);

robot.forward(50);

robot.right(90);

robot.forward(40);

robot.exit();

}

**public** **static** **void** main(String[] args) {

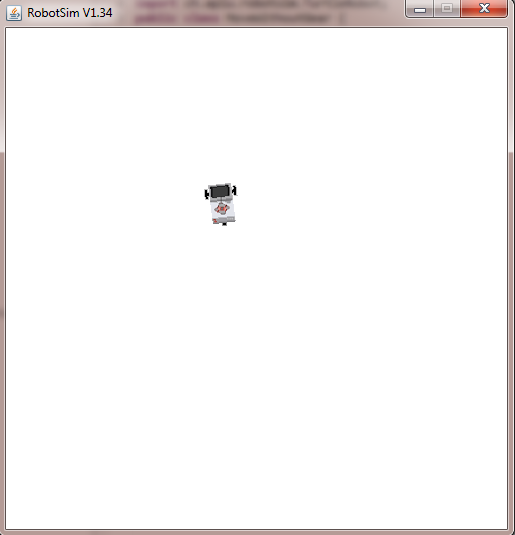
// **TODO** Auto-generated method stub

**new** MoveWithoutGear();

}

}

Output:



**Practical 2**

**Write a program to create a robot with a two motor and move it forward, left, right**

Code:

**import** ch.aplu.robotsim.NxtRobot;

**import** ch.aplu.robotsim.Motor;

**import** ch.aplu.robotsim.MotorPort;

**import** ch.aplu.robotsim.Tools;

**public** **class** MoveWithMotor {

**public** MoveWithMotor() {

NxtRobot r = **new** NxtRobot();

Motor MotA = **new** Motor(MotorPort.*A*);

Motor MotB = **new** Motor(MotorPort.*B*);

r.addPart(MotB);

r.addPart(MotA);

MotA.setSpeed(100);

MotB.setSpeed(100);

MotA.forward();

MotB.forward();

Tools.*delay*(2000);

MotA.stop();

Tools.*delay*(1000);

MotB.forward();

MotA.*delay*(2000);

r.exit();

}

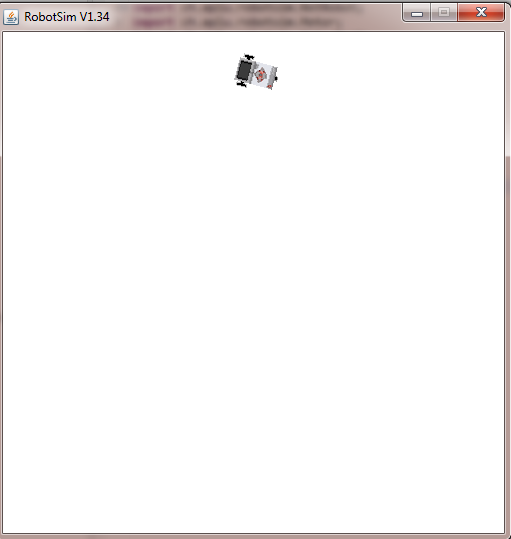
**public** **static** **void** main(String args[]) {

**new** MoveWithMotor();

}

}

Output:



**Practical 3**

**Write a program to do a square using a while loop, doing steps with a for loop, to change directions based on condition, controlling motor speed using switch case**

Code:

**import** java.util.Scanner;

**import** ch.aplu.robotsim.Gear;

**import** ch.aplu.robotsim.NxtRobot;

**import** ch.aplu.robotsim.Tools;

**public** **class** Square {

Square(){

String dir;

NxtRobot r = **new** NxtRobot();

Gear g = **new** Gear();

r.addPart(g);

g.setSpeed(100);

**while**(**true**)

{

Scanner S = **new** Scanner(System.***in***);

System.***out***.println("Enter the direction: ");

dir = S.nextLine();

**if** (dir.equals("left"))

{

g.forward(1000);

g.left(270);

g.forward(1000);

}

**else** **if**(dir.equals("right"))

{

g.forward(1000);

g.right(270);

g.forward(1000);

}

**else** {

r.exit();

}

}

}

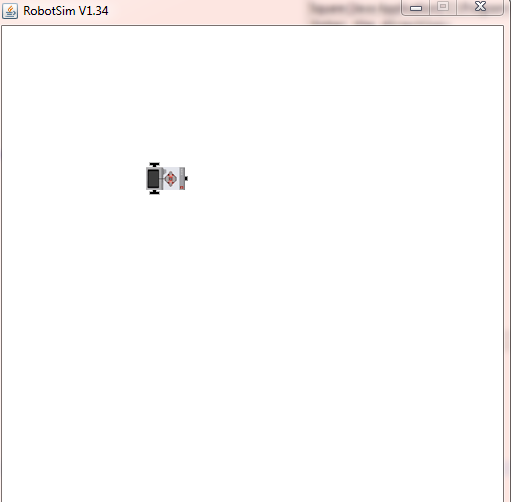
**public** **static** **void** main(String[] args) {

**new** Square();

}

}

Output:



**Practical 4**

**Write a program to create a robot with light sensors to follow a line**

Code:

import ch.aplu.robotsim.Gear;

import ch.aplu.robotsim.LegoRobot;

import ch.aplu.robotsim.LightSensor;

import ch.aplu.robotsim.SensorPort;

import ch.aplu.robotsim.RobotContext;

public class LineFollower {

public LineFollower()

{

LegoRobot r = new LegoRobot();

Gear g = new Gear();

LightSensor ls = new LightSensor(SensorPort.S3);

r.addPart(ls);

r.addPart(g);

while(true)

{

int p = ls.getValue();

if(p<100)

{

g.forward();

}

if(p>300 && p<750)

{

g.leftArc(0.08);

}

if(p>800)

{

g.rightArc(0.05);

}

System.out.println("Sensor Value: " +p);

}

}

public static void main(String arg[])

{

new LineFollower();

}

static

{

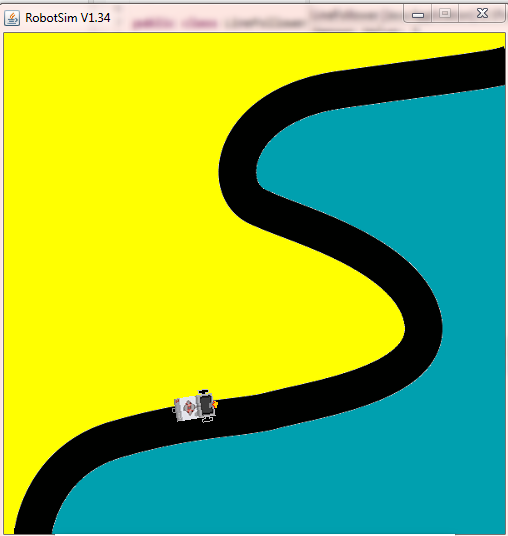
RobotContext.setStartPosition(40,490);

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

}

}

Output:



**Practical 5**

**Write a program to create a robot that does a circle using 2 motors**

Code:

import ch.aplu.robotsim.NxtRobot;

import ch.aplu.robotsim.Motor;

import ch.aplu.robotsim.MotorPort;

import ch.aplu.robotsim.Tools;

public class CircleWith2Motors {

public CircleWith2Motors() {

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();

Tools.delay(200);

A.stop();

Tools.delay(200);

A.forward();

Tools.delay(200);

A.stop();

Tools.delay(200);

A.forward();

Tools.delay(200);

A.stop();

Tools.delay(200);

A.forward();

Tools.delay(200);

A.stop();

Tools.delay(200);

A.forward();

Tools.delay(200);

A.stop();

Tools.delay(200);

A.forward();

Tools.delay(200);

A.stop();

Tools.delay(200);

A.forward();

Tools.delay(200);

A.stop();

Tools.delay(200);

A.forward();

Tools.delay(200);

A.stop();

Tools.delay(200);

A.forward();

Tools.delay(200);

A.stop();

Tools.delay(200);

A.forward();

Tools.delay(200);

A.stop();

Tools.delay(200);

A.forward();

Tools.delay(200);

A.stop();

Tools.delay(100);

A.forward();

r.exit();

}

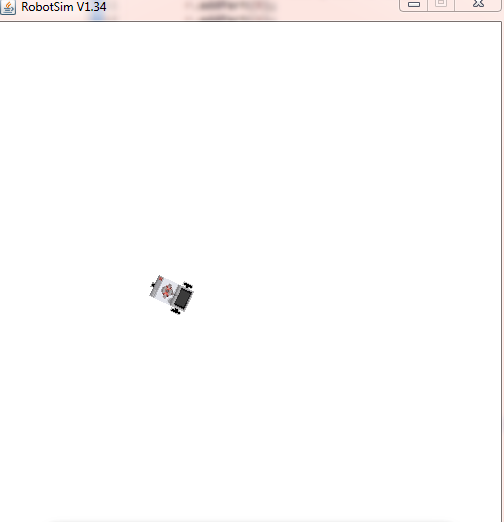
public static void main(String arg[]) {

new CircleWith2Motors();

}

}

Code:



**Practical 6**

**Write a program to create a path following robot**

Code:

import ch.aplu.robotsim.Gear;

import ch.aplu.robotsim.LegoRobot;

import ch.aplu.robotsim.LightSensor;

import ch.aplu.robotsim.RobotContext;

import ch.aplu.robotsim.SensorPort;

public class TrackFollower {

public TrackFollower() {

LegoRobot legoRobot = new LegoRobot();

Gear gear = new Gear();

LightSensor lightSensor = new LightSensor(SensorPort.S3);

legoRobot.addPart(lightSensor);

legoRobot.addPart(gear);

while (true) {

int p = lightSensor.getValue();

if (p == 1000) {

gear.forward();

}

if (p > 50 && p < 1000) {

gear.rightArc(0.08);

}

System.out.println("Sensor Val:" + p);

if (p == 0) {

gear.stop();

}

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

new TrackFollower();

}

static {

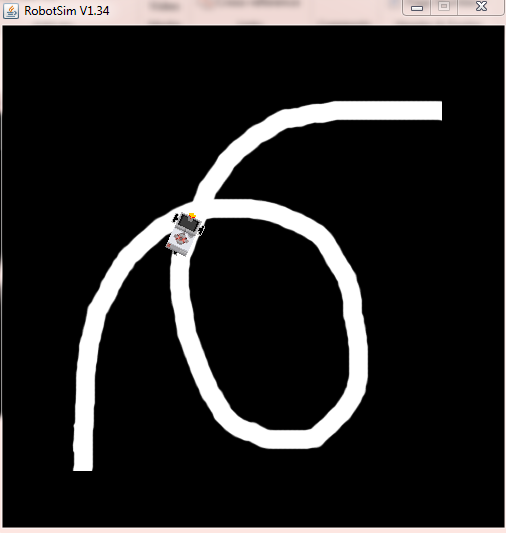
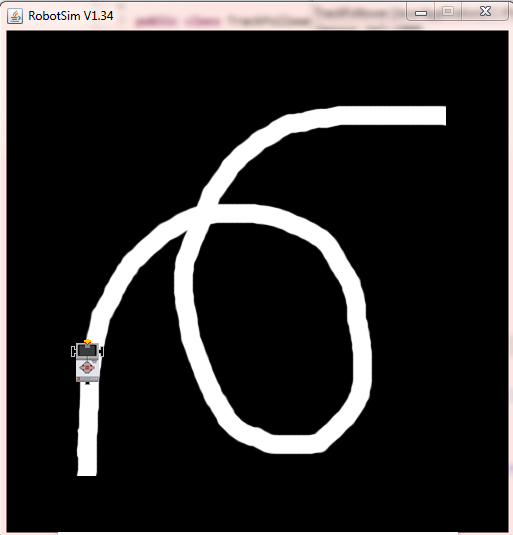
RobotContext.setStartPosition(80, 452);

RobotContext.useBackground("sprites/track.PNG");

}

}

Output:



Code:

import ch.aplu.robotsim.Gear;

import ch.aplu.robotsim.LegoRobot;

import ch.aplu.robotsim.LightSensor;

import ch.aplu.robotsim.SensorPort;

import ch.aplu.robotsim.RobotContext;

public class LineFollower {

public LineFollower()

{

LegoRobot r = new LegoRobot();

Gear g = new Gear();

LightSensor ls = new LightSensor(SensorPort.S3);

r.addPart(ls);

r.addPart(g);

while(true)

{

int p = ls.getValue();

if(p<100)

{

g.forward();

}

if(p>300 && p<750)

{

g.leftArc(0.08);

}

if(p>800)

{

g.rightArc(0.05);

}

System.out.println("Sensor Value: " +p);

}

}

public static void main(String arg[])

{

new LineFollower();

}

static

{

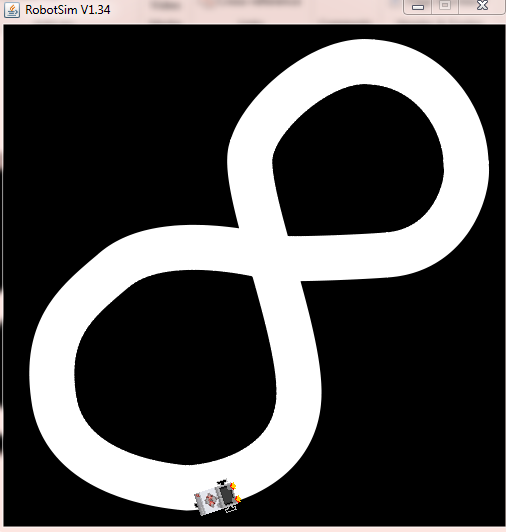
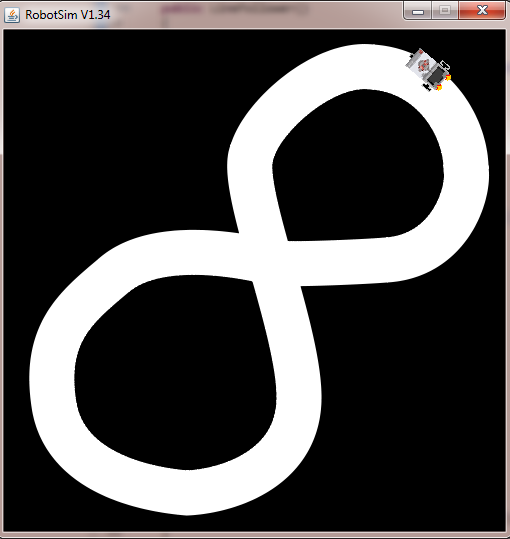
RobotContext.setStartPosition(40,490);

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

}

}

Output:



**Practical 7**

**Write a program to resist obstacles**

Code:

import ch.aplu.robotsim.Gear;

import ch.aplu.robotsim.LegoRobot;

import ch.aplu.robotsim.RobotContext;

import ch.aplu.robotsim.SensorPort;

import ch.aplu.robotsim.Tools;

import ch.aplu.robotsim.TouchSensor;

import ch.aplu.util.QuitPane;

public class ResistObstacle {

public ResistObstacle()

{

LegoRobot robot = new LegoRobot();

Gear g = new Gear();

TouchSensor ts1 = new TouchSensor(SensorPort.S1);

TouchSensor ts2 = new TouchSensor(SensorPort.S2);

robot.addPart(g);

robot.addPart(ts1);

robot.addPart(ts2);

g.forward();

while (!QuitPane.quit()) {

Boolean t1 = ts1.isPressed();

Boolean t2 = ts2.isPressed();

if (t1 && t2) {

g.backward(500);

g.left(400);

g.forward();

} else {

if (t1) {

g.backward(500);

g.left(400);

g.forward();

} else {

if (t2) {

g.backward(500);

g.right(100);

g.forward();

}

}

}

Tools.delay(20);

}

robot.exit();

}

public static void main(String[] args) {

new ResistObstacle();

}

static {

RobotContext.setLocation(10, 10);

RobotContext.setStartDirection(5);

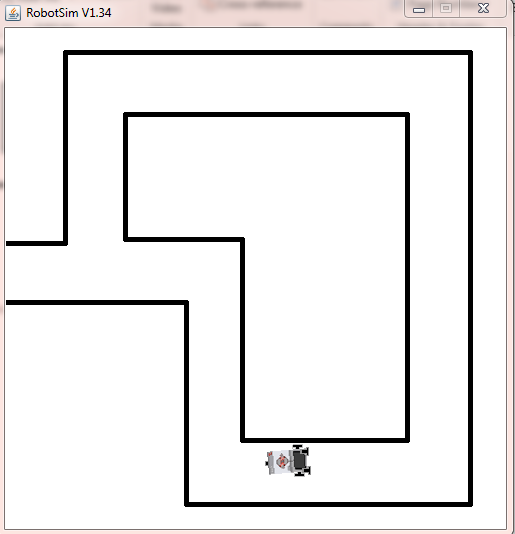
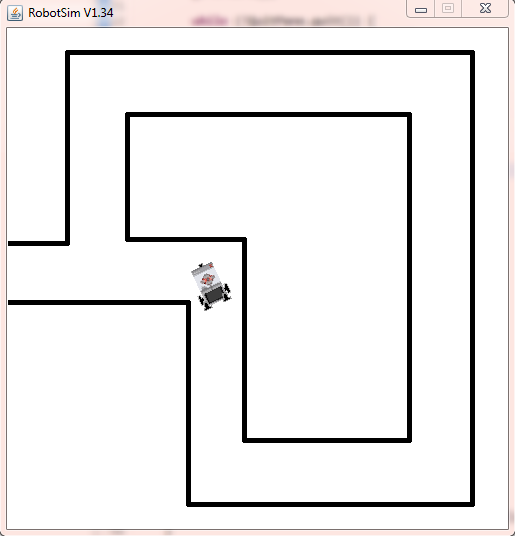
RobotContext.setStartPosition(100, 240);

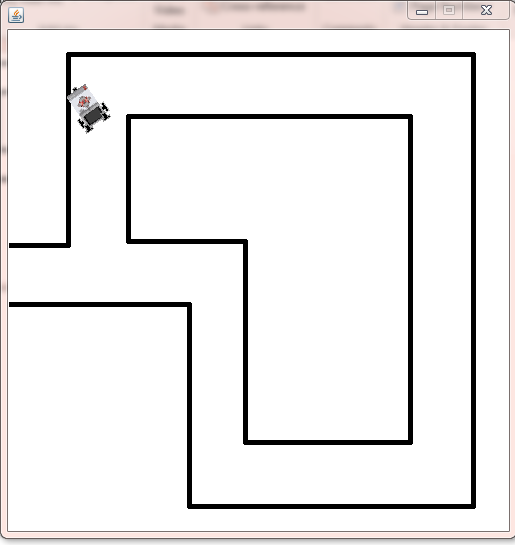
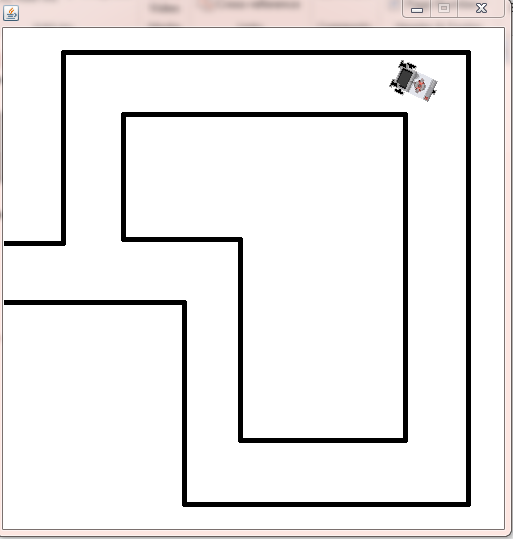
RobotContext.useObstacle(RobotContext.channel);

}

}

Output:





**Practical 8**

**Write a program to implement Torch following robot**

Code:

**import** ch.aplu.robotsim.Gear;

**import** ch.aplu.robotsim.LegoRobot;

**import** ch.aplu.robotsim.LightSensor;

**import** ch.aplu.robotsim.RobotContext;

**import** ch.aplu.robotsim.SensorPort;

**import** ch.aplu.robotsim.Tools;

**public** **class** TorchFollower {

TorchFollower()

{

LegoRobot robot = **new** LegoRobot();

LightSensor lsFR = **new** LightSensor(SensorPort.*S1*, **true**);

LightSensor lsFL = **new** LightSensor(SensorPort.*S2*, **true**);

LightSensor lsRR = **new** LightSensor(SensorPort.*S3*, **true**);

LightSensor lsRL = **new** LightSensor(SensorPort.*S4*, **true**);

Gear gear = **new** Gear();

robot.addPart(gear);

robot.addPart(lsFR);

robot.addPart(lsFL);

robot.addPart(lsRL);

robot.addPart(lsRR);

gear.setSpeed(25);

gear.forward();

**double** s = 0.02;

**while** (!robot.isEscapeHit())

{

**int** vFR = lsFR.getValue();

**int** vFL = lsFL.getValue();

**int** vRR = lsRR.getValue();

**int** vRL = lsRL.getValue();

**double** d = 1.0 \* (vFL - vFR) / (vFL + vFR);

**if** (vRL + vRR > vFL + vFR) // torch behind robot

gear.left();

**else** **if** (d > -s && d < s)

gear.forward();

**else**

{

**if** (d >= s)

gear.leftArc(0.05);

**else**

gear.rightArc(0.05);

}

Tools.*delay*(100);

}

robot.exit();

}

**public** **static** **void** main(String[] args)

{

TorchFollower t = **new** TorchFollower();

}

// ------------------ Environment --------------------------

**static**

{

RobotContext.*useTorch*(1, 150, 250, 100);

}

}

Output:

