

**PRACTICAL 1a**

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

**DESC:**

NxtRobot() - Constructor for class ch.aplu.robotsim.NxtRobot

Gear() - Constructor for class ch.aplu.robotsim.Gear

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

addPart(Part) - Method in class ch.aplu.robotsim.LegoRobot

Assembles the given part into the robot.

setSpeed(int) - Method in class ch.aplu.robotsim.Gear

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

forward() - Method in class ch.aplu.robotsim.Gear

Starts the forward movement.

left() - Method in class ch.aplu.robotsim.Gear

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

right() - Method in class ch.aplu.robotsim.Gear

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);

}

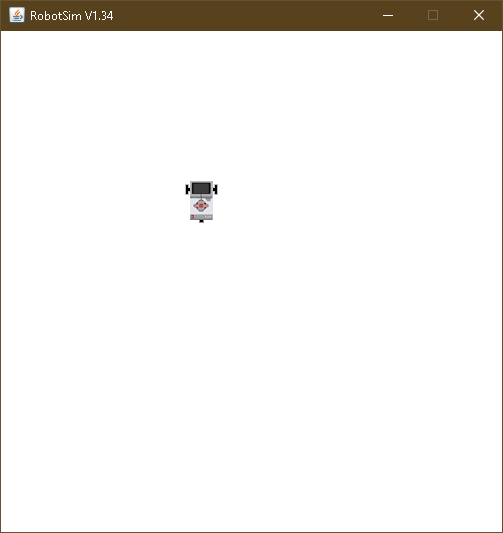
public static void main (String[] args) {

new Prac\_1a();

}

}

**OUTPUT:**

****

**PRACTICAL 1b**

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

**DESC:**

TurtleRobot() - Constructor for class ch.aplu.robotsim.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);

}

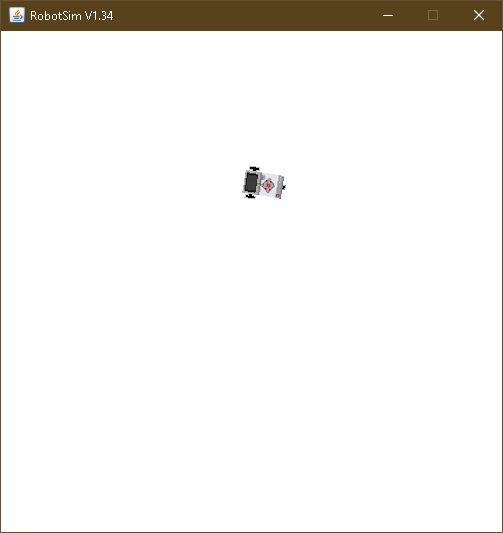
public static void main (String[] args) {

new Prac\_1b();

}

}

**OUTPUT:**

****

**PRACTICAL 2**

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

**DESC:**

Motor - Class in ch.aplu.robotsim

Class that represents one of the NXT motors.

Motor(MotorPort) - Constructor for class ch.aplu.robotsim.Motor

Creates a motor instance that is plugged into given port.

Tools() - Constructor for class ch.aplu.robotsim.Tools

delay(int) - Static method in class ch.aplu.robotsim.Tools

Suspends execution of the current thread for the given amount of time (unless the game grid window is disposed).

stop() - Method in class ch.aplu.robotsim.Motor

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();

}

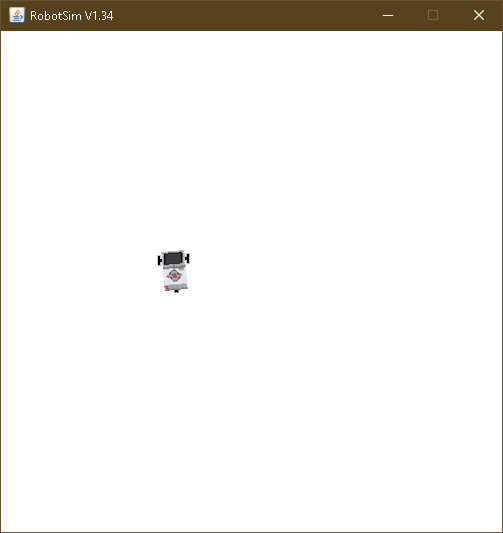
public static void main(String args[]){

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);

}

}

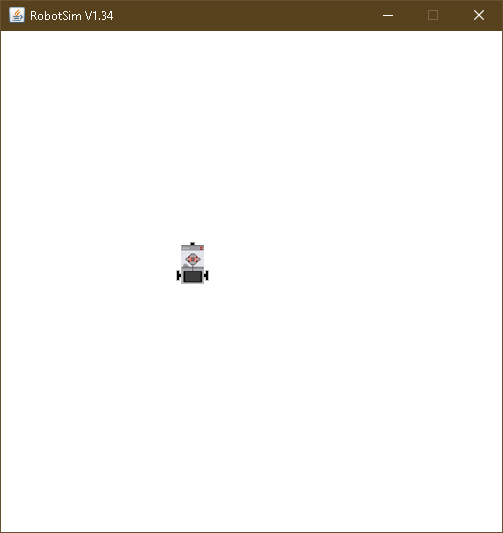
public static void main (String[] args) {

new Prac\_3();

}

}

**OUTPUT:**

****

**PRACTICAL 4**

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

**DESC:**

RobotContext() - Constructor for class ch.aplu.robotsim.RobotContext

Creates a RobotContext instance.

setStartPosition(int, int) - Static method in class ch.aplu.robotsim.RobotContext

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

useBackground(String) - Static method in class ch.aplu.robotsim.RobotContext

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

LegoRobot() - Constructor for class ch.aplu.robotsim.LegoRobot

Creates a robot with its playground using defaults from RobotContext.

LightSensor(SensorPort) - Constructor for class ch.aplu.robotsim.LightSensor

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

getValue() - Method in class ch.aplu.robotsim.LightSensor

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

leftArc(double) - Method in class ch.aplu.robotsim.Gear

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

rightArc(double) - Method in class ch.aplu.robotsim.Gear

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);

}

}

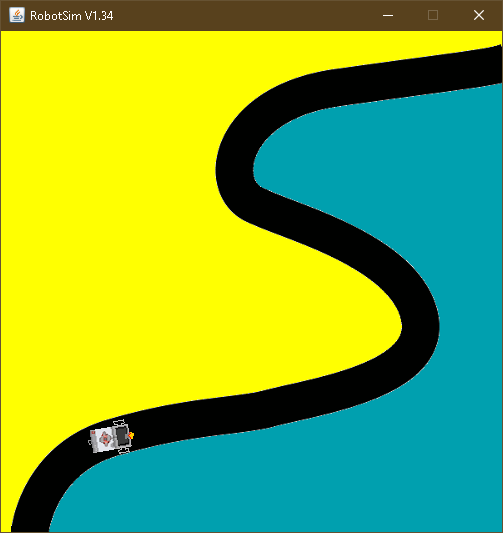
public static void main (String args[]){

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();

}

}

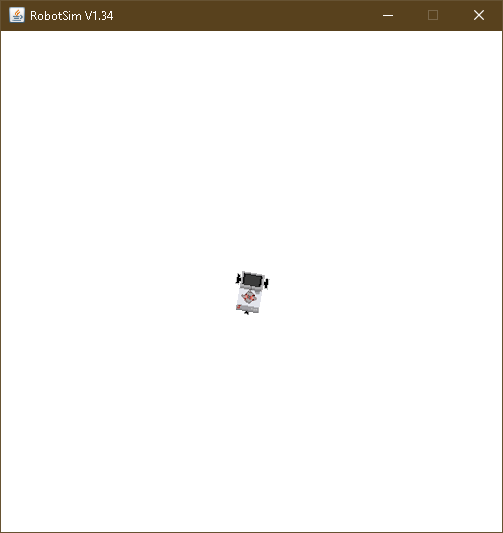
public static void main(String arg[]) {

new Prac\_5();

}

}

**OUTPUT:**

****

**PRACTICAL 6**

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

**DESC:**

NxtContext() - Constructor for class ch.aplu.robotsim.NxtContext

setStartDirection(double) - Static method in class ch.aplu.robotsim.RobotContext

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);

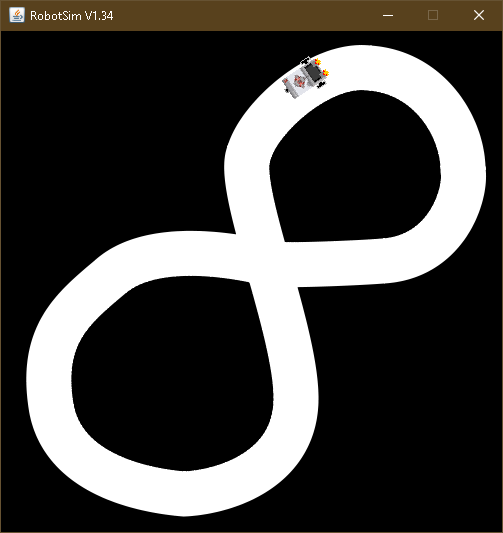
NxtContext.setStartDirection(-90);

NxtContext.useBackground("sprites/path.gif");

}

}

**OUTPUT:**

****

**PRACTICAL 7**

**AIM:** WRITE A PROGRAM TO RESIST OBSTACLES.

**DESC:**

TouchSensor(SensorPort) - Constructor for class ch.aplu.robotsim.TouchSensor

Creates a sensor instance connected to the given port.

isPressed() - Method in class ch.aplu.robotsim.TouchSensor

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

backward() - Method in class ch.aplu.robotsim.TurtleRobot

Starts moving backward and returns immediately.

useObstacle(Obstacle) - Static method in class ch.aplu.robotsim.RobotContext

Defines the given obstacle to be used as touch obstacle.

channel - Static variable in class ch.aplu.robotsim.RobotContext

**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);

}

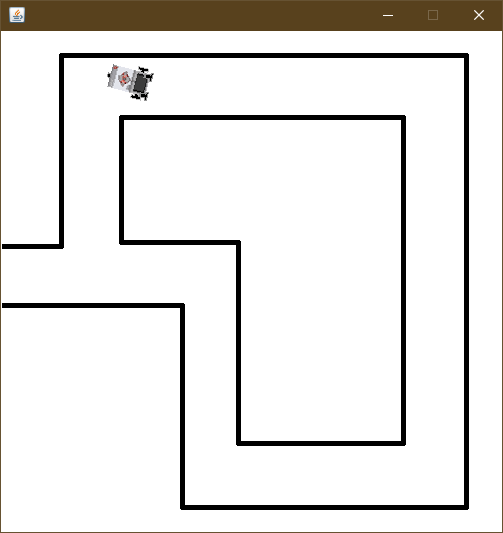
public static void main(String args[]){

new Prac\_7();

}

}

**OUTPUT:**

****

**PRACTICAL 8**

**AIM:** ULTRASONIC SENSOR.

**DESC:**

UltrasonicSensor(SensorPort) - Constructor for class ch.aplu.robotsim.UltrasonicSensor

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

setBeamAreaColor(Color) - Method in class ch.aplu.robotsim.UltrasonicSensor

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

setProximityCircleColor(Color) - Method in class ch.aplu.robotsim.UltrasonicSensor

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

getDistance() - Method in class ch.aplu.robotsim.UltrasonicSensor

Returns the distance to the nearest target object.

useTarget(String, Point[], int, int) - Static method in class ch.aplu.robotsim.RobotContext

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);

}

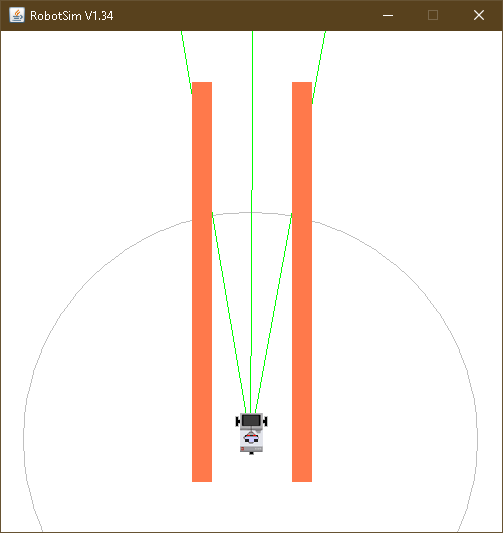
public static void main(String[] args) {

new Prac\_8();

}

}

**OUTPUT:**

****

**ASSIGNMENT 1A**

**AIM:** WRITE A PROGRAM TO CREATE A ROBOT TO PERFORM SQUARE MOTION WITHOUT USING GEAR.

**CODE:**

import ch.aplu.robotsim.\*;

public class Assignment1A {

Assignment1A(){

TurtleRobot t = new TurtleRobot();

t.setTurtleSpeed(100);

while (true){

t.forward(200);

t.left(90);

}

}

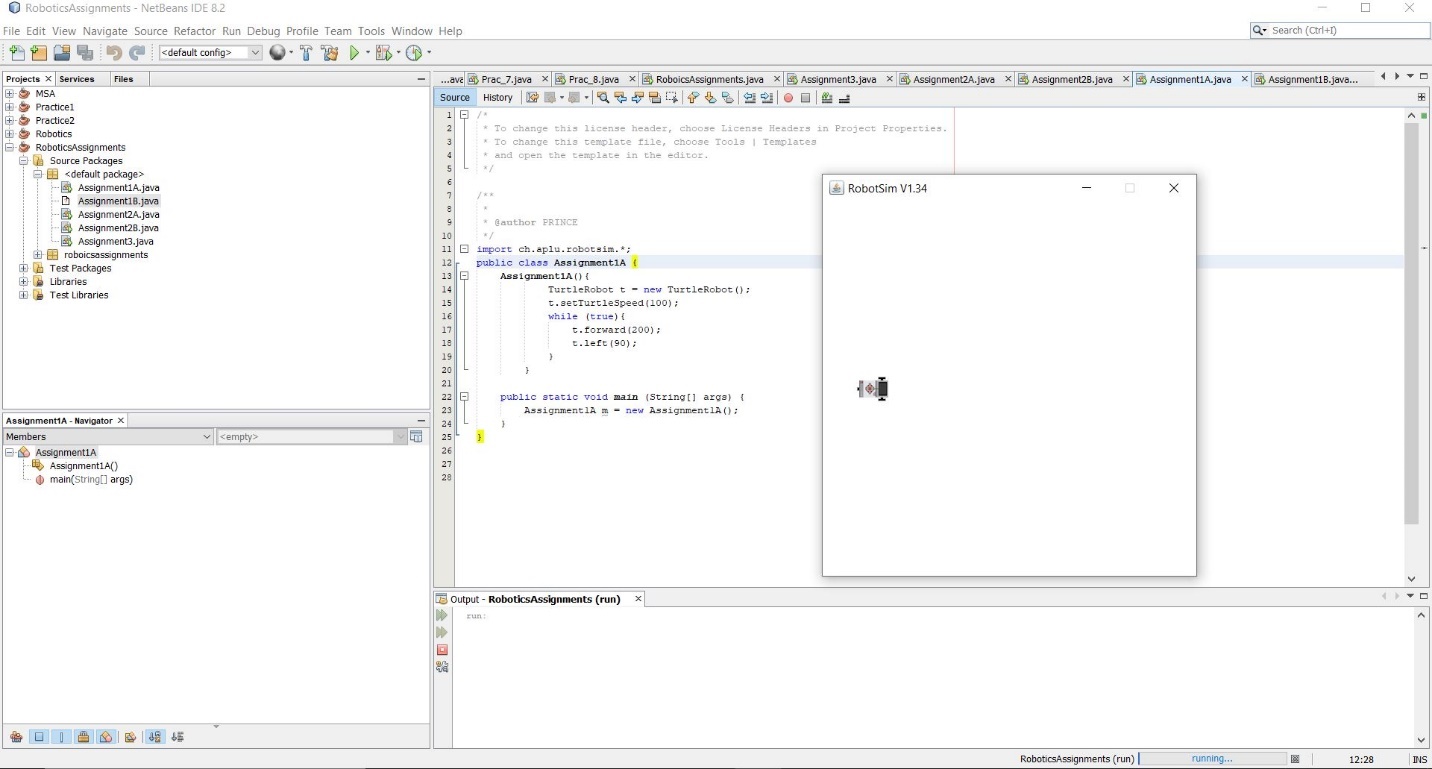
public static void main (String[] args) {

new Assignment1A();

}

}

**OUTPUT:**

****

**ASSIGNMENT 1B**

**AIM:** WRITE A PROGRAM TO CREATE A ROBOT TO PERFORM CIRCULAR MOTION WITHOUT USING GEAR.

**CODE:**

import ch.aplu.robotsim.\*;

public class Assignment1B {

Assignment1B(){

TurtleRobot t = new TurtleRobot();

t.setTurtleSpeed(100);

while (true){

t.forward(2);

t.left(2);

}

}

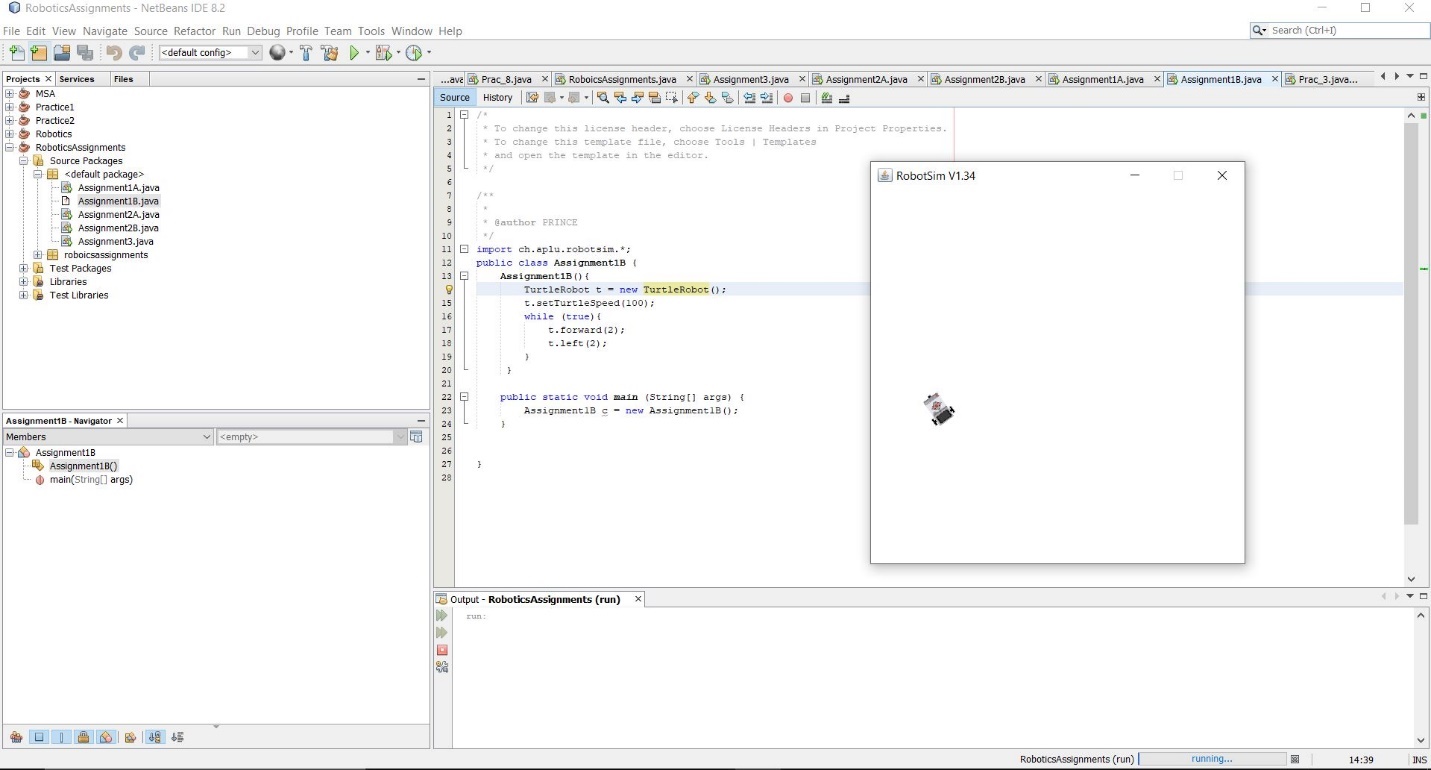
public static void main (String[] args) {

new Assignment1B();

}

}

**OUTPUT:**

****

**ASSIGNMENT 2A**

**AIM:** CREATE A ROBOT TO PERFORM RECTANGULAR MOTION USING GEARS.

**CODE:**

import ch.aplu.robotsim.\*;

public class Assignment2A {

Assignment2A(){

NxtRobot robot = new NxtRobot();

Gear g = new Gear();

robot.addPart(g);

g.setSpeed(100);

while (true){

g.forward(800);

g.left(280);

g.forward(1500);

g.left(280);

}

}

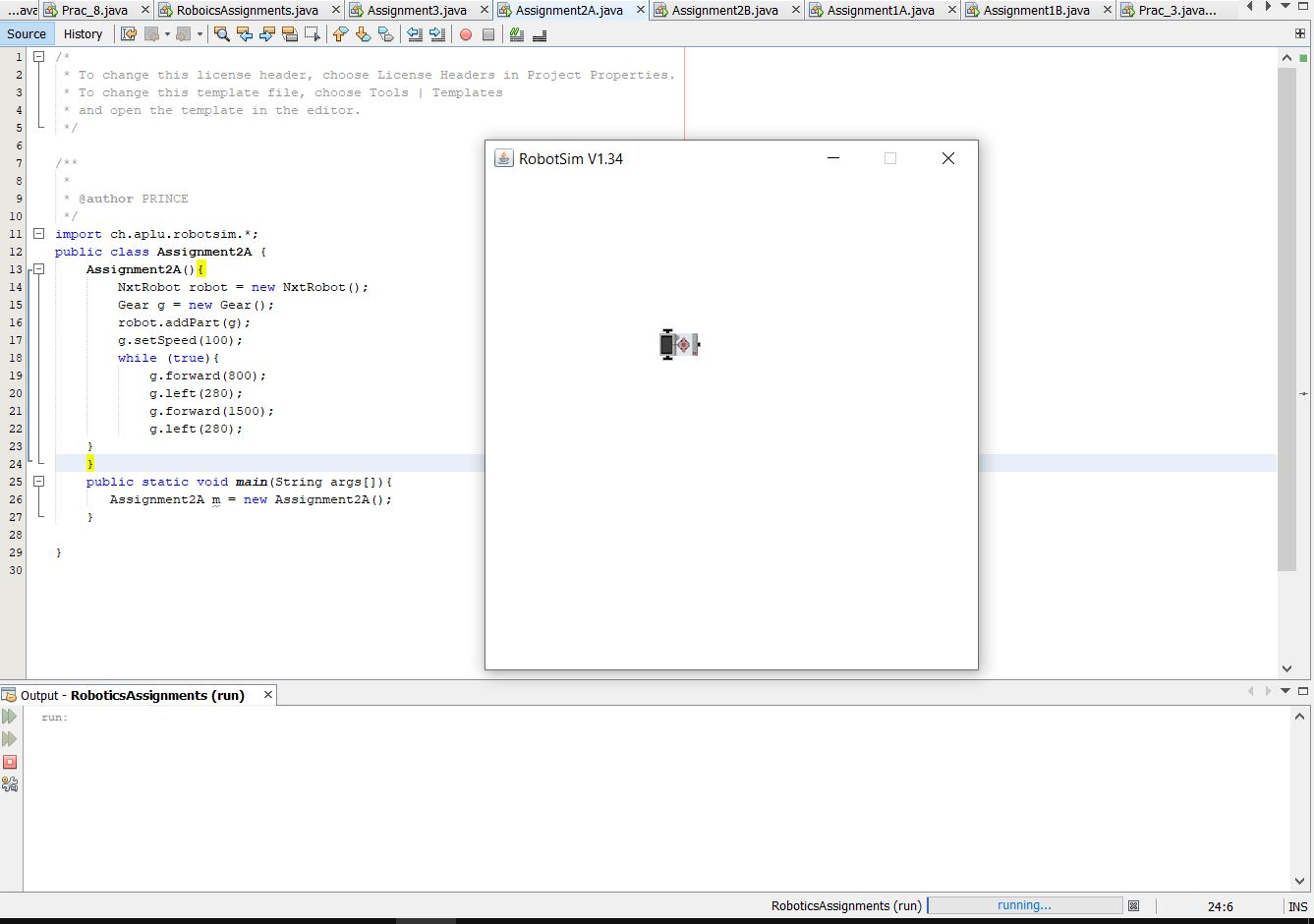
public static void main (String[] args) {

Assignment2A m = new Assignment2A();

}

}

**OUTPUT:**

****

**ASSIGNMENT 2B**

**AIM:** CREATE A ROBOT TO PERFORM CIRCULAR MOTION USING GEARS.

**CODE:**

import ch.aplu.robotsim.\*;

public class Assignment2B {

Assignment2B (){

NxtRobot robot = new NxtRobot();

Gear g = new Gear();

robot.addPart(g);

g.setSpeed(700);

while (true){

g.leftArc(0.4);

}

}

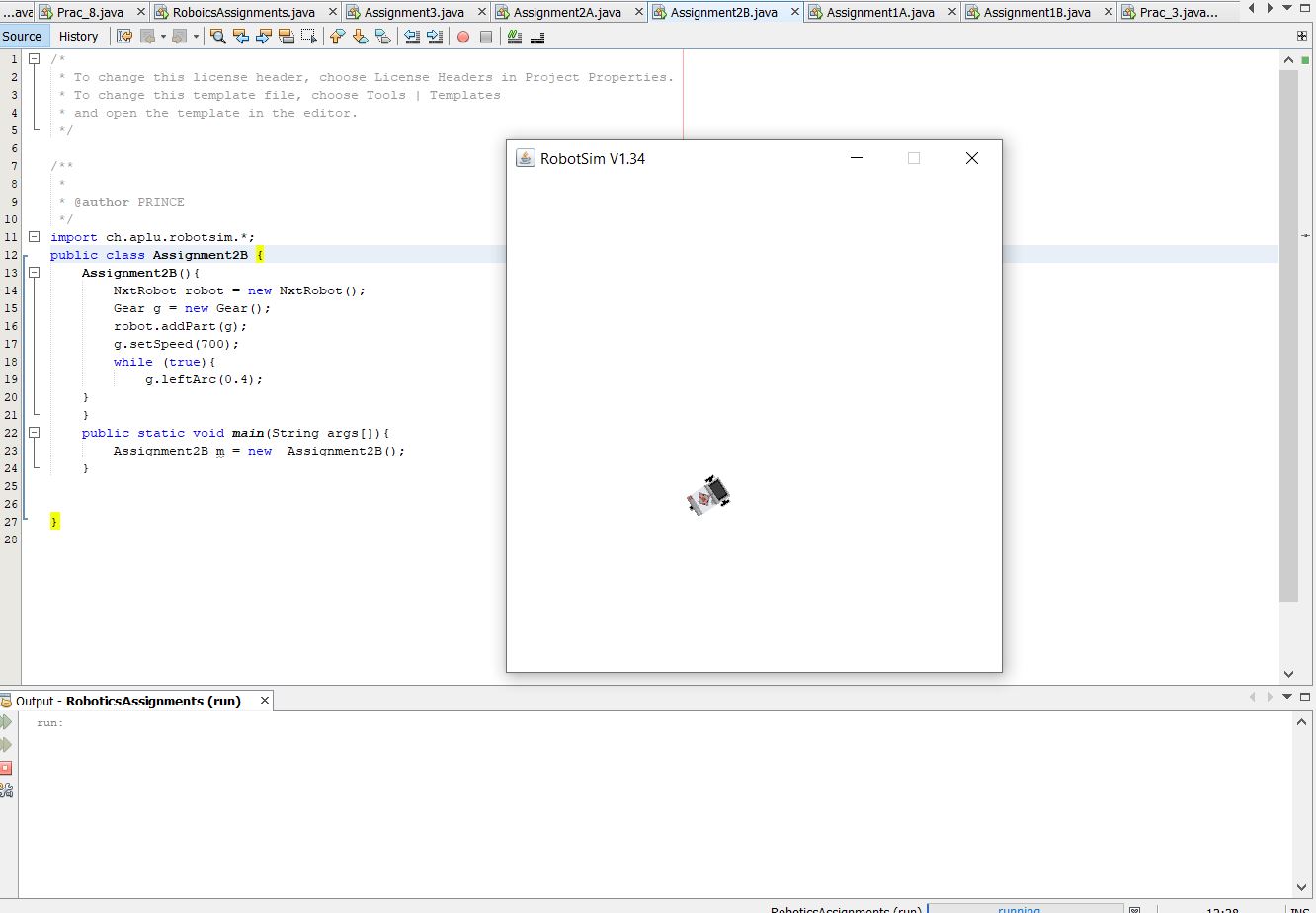
public static void main (String[] args) {

Assignment2B m = new Assignment2B();

}

}

**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 USING SWITCH CASE.

**CODE:**

import ch.aplu.robotsim.\*;

import java.util.\*;

public class Assignment3 {

Assignment3(){

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;

}

}

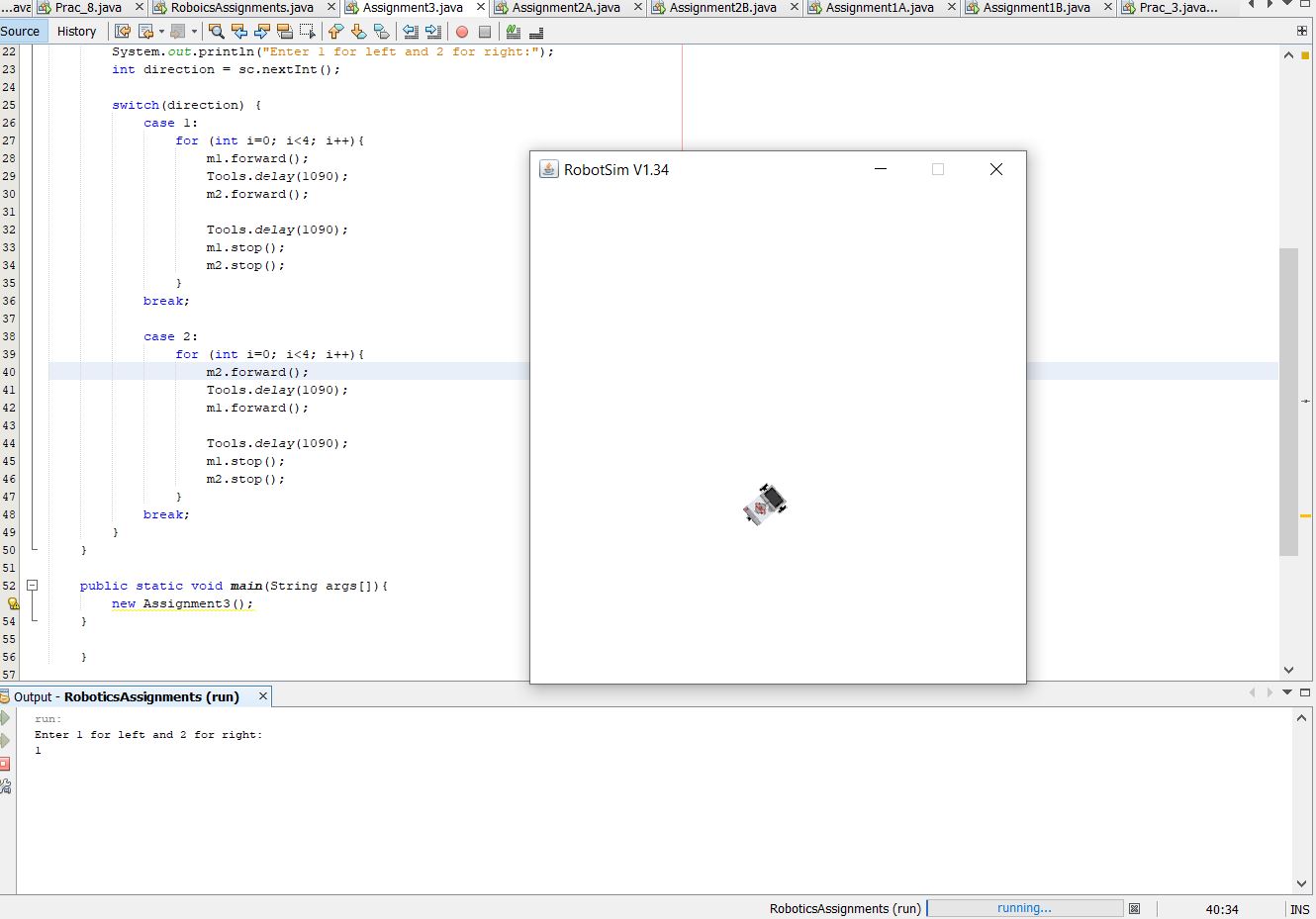
public static void main(String args[]){

new Assignment3();

}

}

**OUTPUT:**

****