MSc. Computer Science Semester-III

PAPER IV (Robotics)

INDEX

NO	DATE	TITLE		SIGN
1	26-06-23	a. Moving a robot left, right, and forward using gears.	b. Moving a robot in a square motion.	
2	26-06-23	Robot performing circular motion.		
3	01-07-23	Moving a robot with motors.		
4	03-07-23	a. Circular motion using arch function with gear.	b. Circular motion using arch function without gear.	
5	03-07-23	a. Line-following robot.	b. Line-following robot on a different image.	
6	08-07-23	a. Pathfinding robot.	b. Pathfinding robot on a different image.	
7	10-07-23	Obstacle resistance using touch sensor.		
8	11-07-23	Write a program to create a robot with ultrasonic sensor.		
9	08-08-23	Torch follower.		
10	08-08-23	Shadow follower.		

Practical 1a: Moving a robot left, right, and forward using gears.

AIM: Write a program to move a robot left, right, and forward using gears

DESCRIPTION:

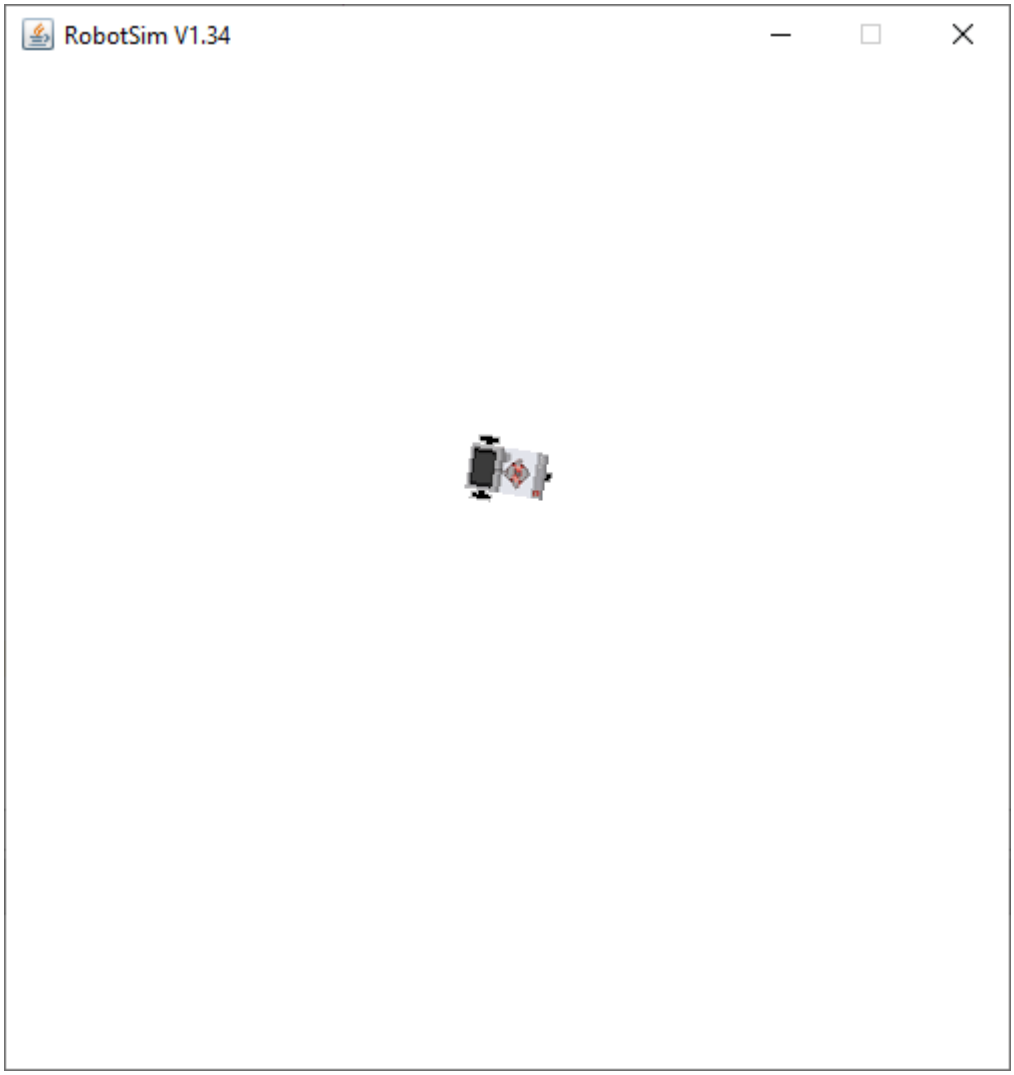
1. Gear(): This function creates an instance of a gear, with the right motor plugged into port A and the left motor plugged into port B. It seems to initialize the motors' configuration.
2. forward(int duration): This function starts a forward movement for the specified duration in milliseconds and then stops. It controls the robot's movement in the forward direction.
3. left(int duration): This function initiates a left rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the left.
4. right(int duration): Similar to the left function, this function starts a right rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the right.
5. setSpeed(int speed): This function sets the speed of the robot to the given value in arbitrary units. It allows you to control how fast the robot moves.
6. stop(): This function stops the movement of the robot. It halts any ongoing motion.
7. NxtRobot(): This function creates an instance of a turtle robot. The name "turtle robot" suggests that it might be designed for movement and potentially drawing, similar to the concept of a "turtle graphics" robot.

SOURCE CODE:

```
import ch.aplu.robotsim.*;
public class MoveWithGear
{
    MoveWithGear()
    {
        NxtRobot robot = new NxtRobot();
        Gear gear = new Gear();
        robot.addPart(gear);

        gear.forward(1000);
        gear.setSpeed(30);

        gear.left(800);
        gear.forward(1000);
        gear.right(800);
        gear.forward(1000);
        robot.exit();
    }
    public static void main(String args[])
```

```
    {
        MoveWithGear m = new MoveWithGear();
    }
}
```

OUTPUT:



Practical 1b: Moving a robot in a square motion.

AIM: Write a program to make a robot move in a square pattern.

DESCRIPTION:

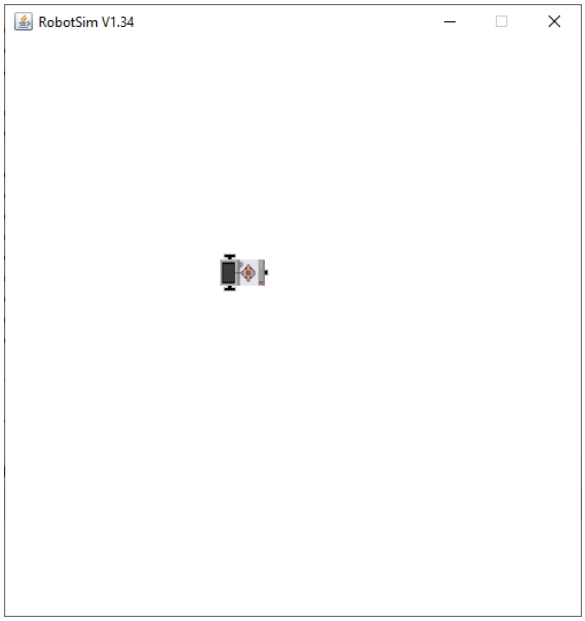
1. Gear(): This function creates an instance of a gear, with the right motor plugged into port A and the left motor plugged into port B. It seems to initialize the motors' configuration.

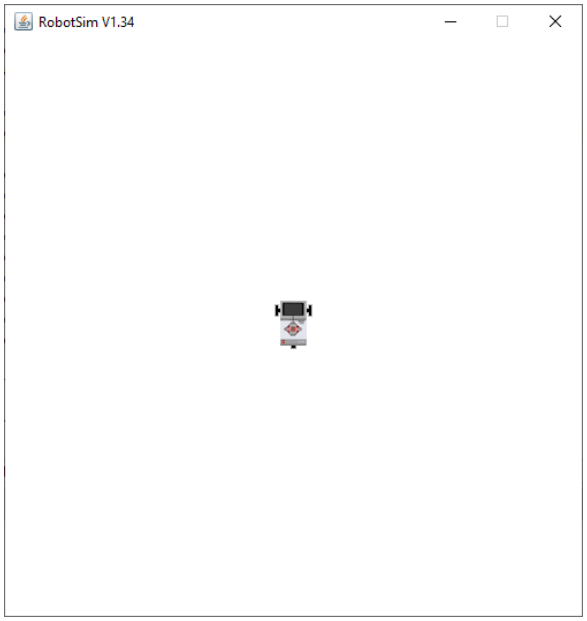
2. forward(int duration): This function starts a forward movement for the specified duration in milliseconds and then stops. It controls the robot's movement in the forward direction.
3. left(int duration): This function initiates a left rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the left.
4. right(int duration): Similar to the left function, this function starts a right rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the right.
5. setSpeed(int speed): This function sets the speed of the robot to the given value in arbitrary units. It allows you to control how fast the robot moves.
6. stop(): This function stops the movement of the robot. It halts any ongoing motion.
7. NxtRobot(): This function creates an instance of a turtle robot. The name "turtle robot" suggests that it might be designed for movement and potentially drawing, similar to the concept of a "turtle graphics" robot.

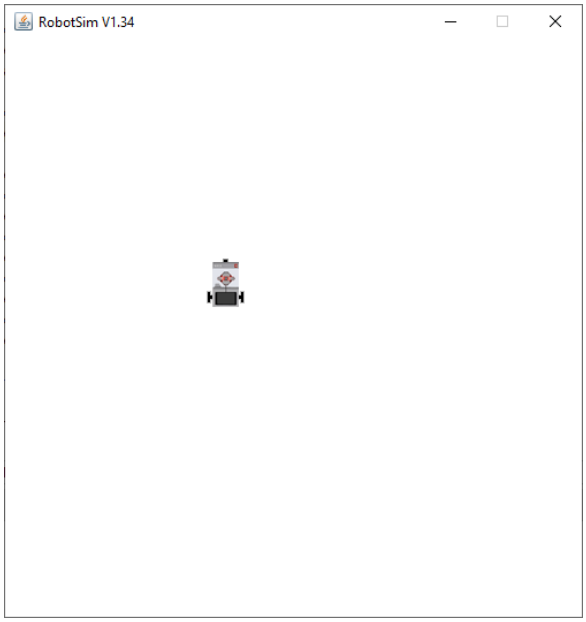
SOURCE CODE:

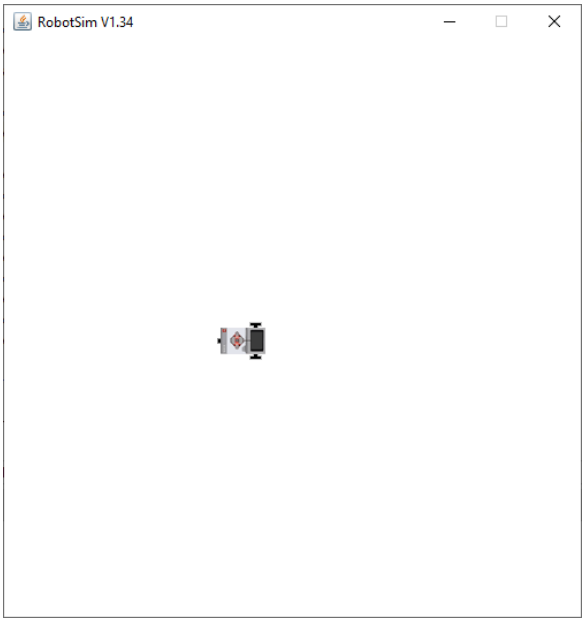
```
import ch.aplu.robotsim.*;
class MoveWithGear
{
    MoveWithGear()
    {
        NxtRobot robot = new NxtRobot();
        Gear gear = new Gear();
        robot.addPart(gear);

        gear.forward(1000);
        gear.setSpeed(30);

        gear.left(900);
        gear.forward(2000);
        gear.left(900);
        gear.forward(2000);
        gear.left(900);
        gear.forward(2000);
        gear.left(900);
        gear.forward(2000);
        robot.exit();
    }
    public static void main(String args[])
    {
        MoveWithGear m = new MoveWithGear();
    }
}
```

OUTPUT:


RobotSim V1.34


RobotSim V1.34


RobotSim V1.34


RobotSim V1.34

Practical 2: Robot performing circular motion.

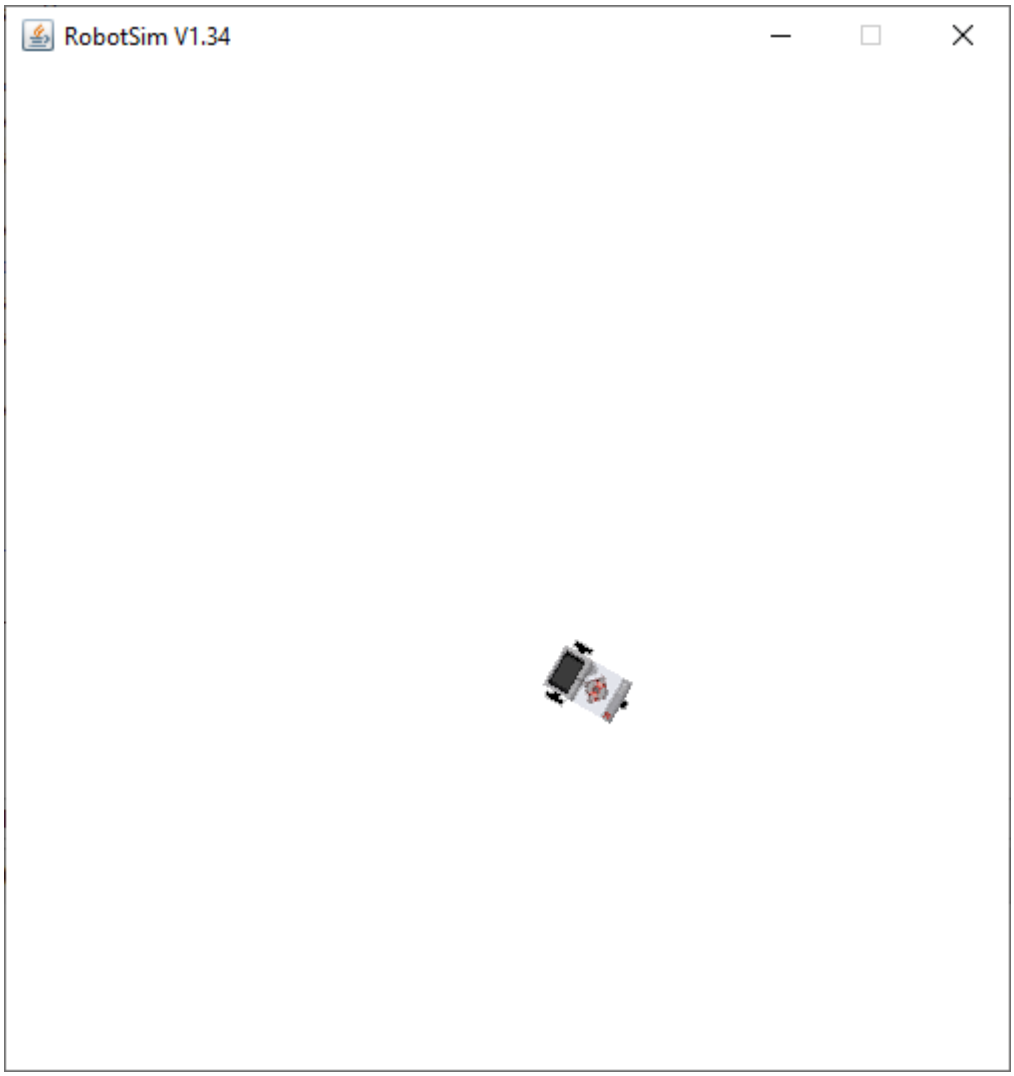
AIM: Write a program to make a robot move in a circular motion.

DESCRIPTION:

SOURCE CODE:

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

public class MoveWithGear
{
    MoveWithGear()
    {
        NxtRobot robot = new NxtRobot();
        Gear gear = new Gear();
        robot.addPart(gear);

        gear.setSpeed(200);
        for (int i = 0; i<100;i++){
            gear.forward(150);
            gear.right(30);
        }
        robot.exit();


    }
    public static void main(String args[])
    {
        MoveWithGear m = new MoveWithGear();
    }
}
```

OUTPUT:



Practical 3: Moving a robot with motors.

AIM: Write a program for moving a robot with motors.

DESCRIPTION:

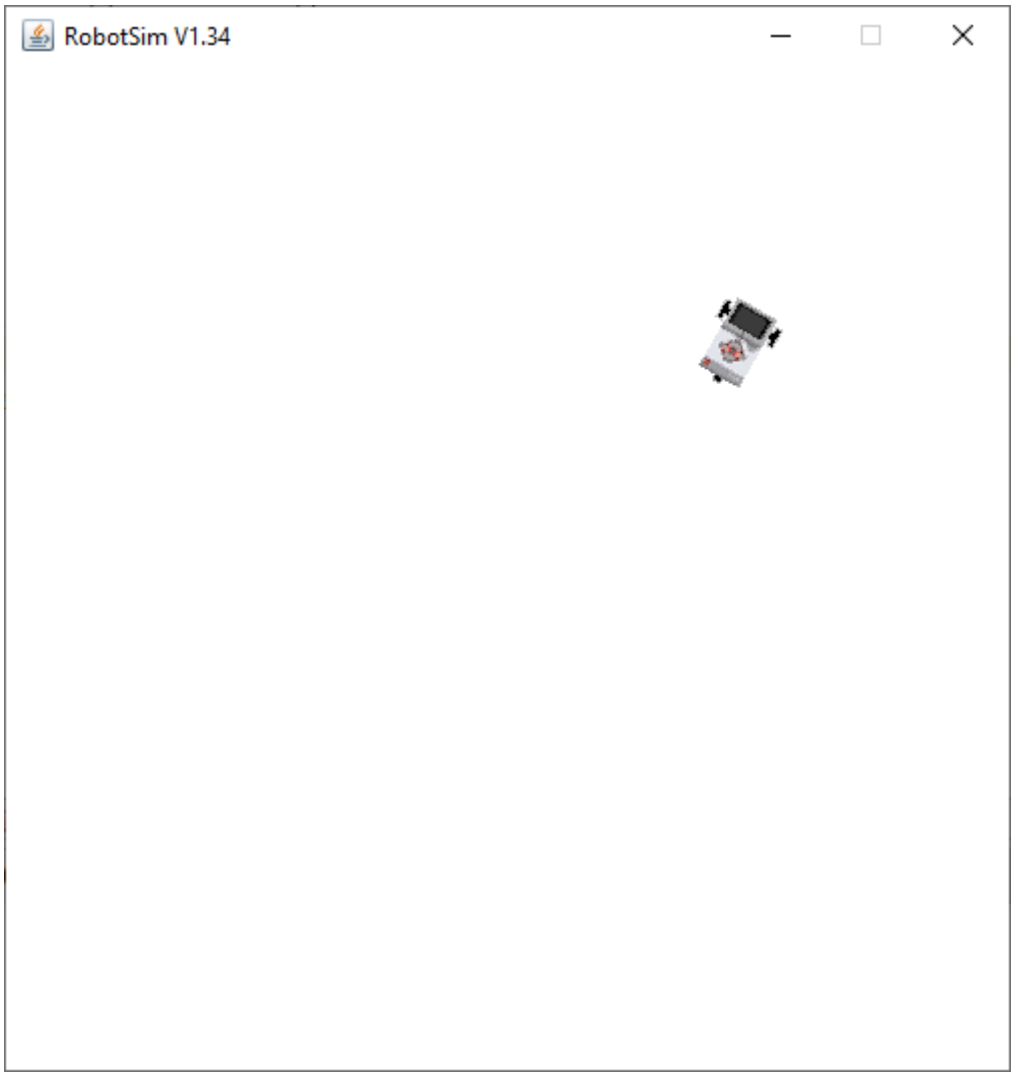
1. Gear(): This function creates an instance of a gear, with the right motor plugged into port A and the left motor plugged into port B. It seems to initialize the motors' configuration.
2. forward(int duration): This function starts a forward movement for the specified duration in milliseconds and then stops. It controls the robot's movement in the forward direction.
3. left(int duration): This function initiates a left rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the left.
4. right(int duration): Similar to the left function, this function starts a right rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the right.
5. setSpeed(int speed): This function sets the speed of the robot to the given value in arbitrary units. It allows you to control how fast the robot moves.
6. stop(): This function stops the movement of the robot. It halts any ongoing motion.
7. NxtRobot(): This function creates an instance of a turtle robot. The name "turtle robot" suggests that it might be designed for movement and potentially drawing, similar to the concept of a "turtle graphics" robot.

SOURCE CODE:

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

public class MoveWithMotors
{
    public MoveWithMotors()
    {
        NxtRobot robot=new NxtRobot();
        Motor motA=new Motor(MotorPort.A);
        Motor motB=new Motor(MotorPort.B);
        robot.addPart(motA);
        robot.addPart(motB);
        motA.forward();
        motB.forward();
        Tools.delay(2000);

        motA.stop();
        Tools.delay(1050);
        motA.forward();
        Tools.delay(2000);

        motB.stop();
```

```
        Tools.delay(1050);
        motB.forward();
        Tools.delay(2000);

        robot.exit();
    }
    public static void main(String args[])
    {
        new MoveWithMotors();
    }
}
```

OUTPUT:



Practical 4a: Circular motion using arch function with gear.

AIM: Write a program to perform circular motion using an arch function with gears.

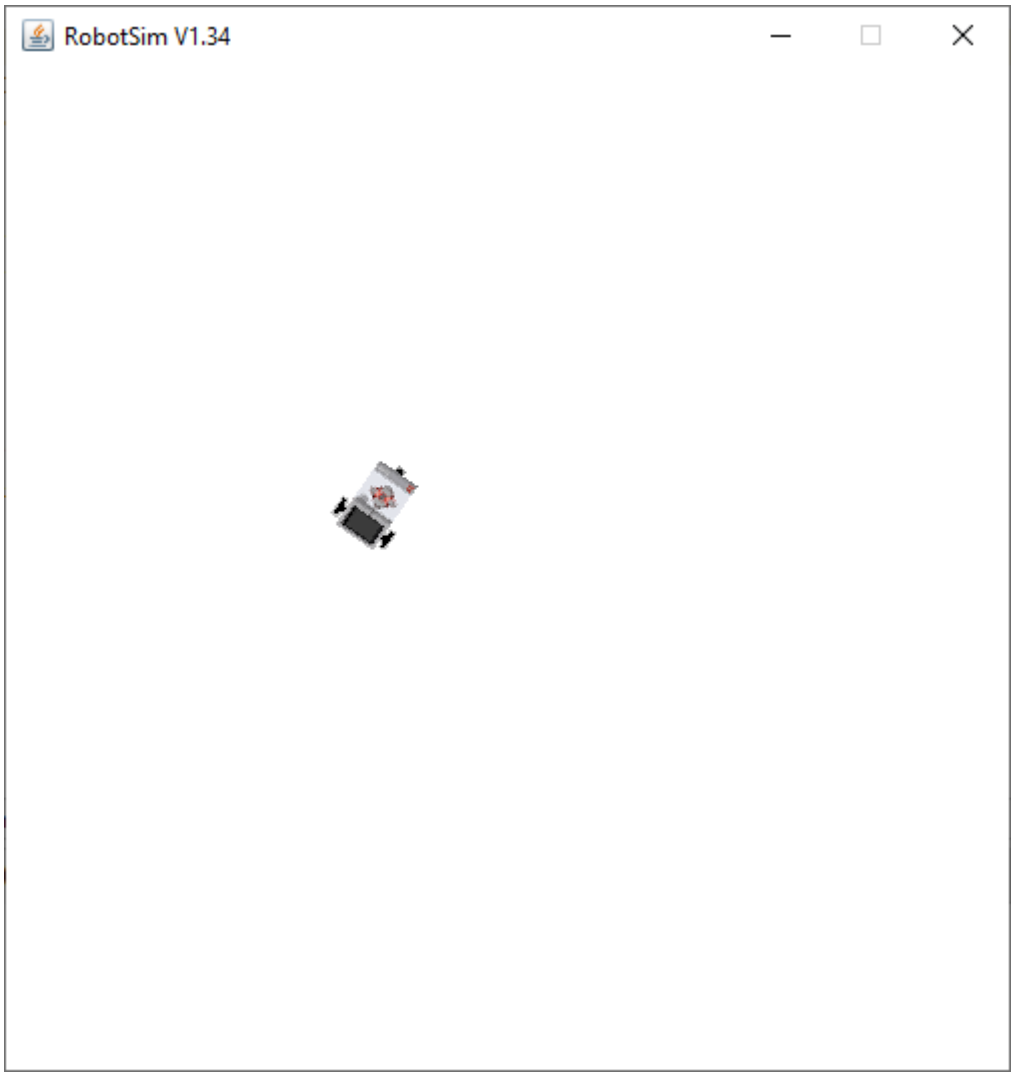
DESCRIPTION:

1. Gear(): This function creates an instance of a gear, with the right motor plugged into port A and the left motor plugged into port B. It seems to initialize the motors' configuration.
2. forward(int duration): This function starts a forward movement for the specified duration in milliseconds and then stops. It controls the robot's movement in the forward direction.
3. left(int duration): This function initiates a left rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the left.
4. right(int duration): Similar to the left function, this function starts a right rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the right.
5. setSpeed(int speed): This function sets the speed of the robot to the given value in arbitrary units. It allows you to control how fast the robot moves.
6. stop(): This function stops the movement of the robot. It halts any ongoing motion.
7. NxtRobot(): This function creates an instance of a turtle robot. The name "turtle robot" suggests that it might be designed for movement and potentially drawing, similar to the concept of a "turtle graphics" robot.

SOURCE CODE:

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

 public class Circlem {
    public Circlem()
    {

        NxtRobot robot = new NxtRobot();
        Gear gear = new Gear();
        robot.addPart(gear);
        gear.setSpeed(60);
        gear.leftArc(0.2, 7000);
        gear.rightArc(0.2);
        Tools.delay(5000);
        robot.exit();
    }

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

OUTPUT:



Practical 4b: Circular motion using arch function without gear.

AIM: Write a program to perform circular motion using an arch function without gears.

DESCRIPTION:

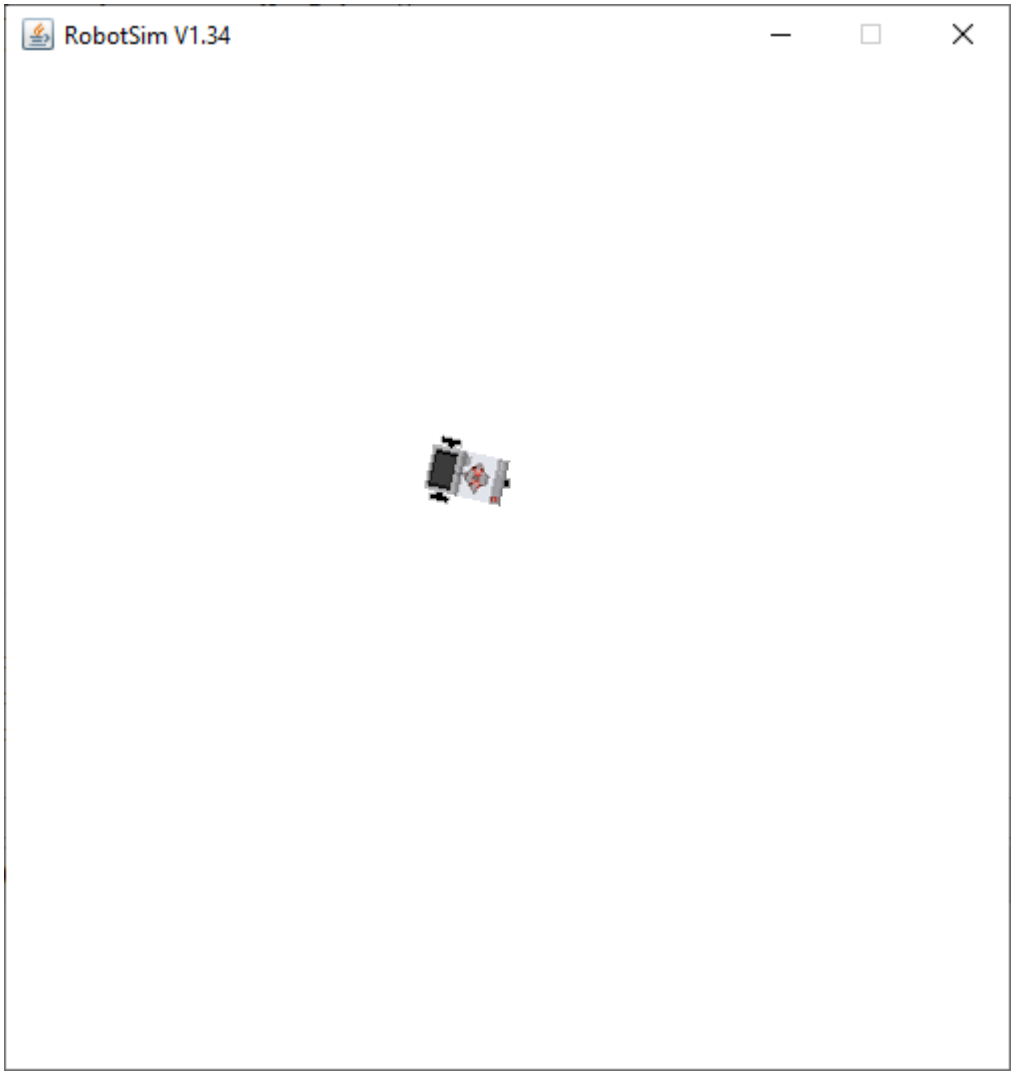
1. forward(int duration): This function starts a forward movement for the specified duration in milliseconds and then stops. It controls the robot's movement in the forward direction.
2. left(int duration): This function initiates a left rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the left.
3. right(int duration): Similar to the left function, this function starts a right rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the right.
4. setSpeed(int speed): This function sets the speed of the robot to the given value in arbitrary units. It allows you to control how fast the robot moves.
5. stop(): This function stops the movement of the robot. It halts any ongoing motion.
6. NxtRobot(): This function creates an instance of a turtle robot. The name "turtle robot" suggests that it might be designed for movement and potentially drawing, similar to the concept of a "turtle graphics" robot.

SOURCE CODE:

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

public class CircularGear {
    CircularGear() {
        NxtRobot robot = new NxtRobot();
        Gear gear = new Gear();
        robot.addPart(gear);
        gear.forward(200);
        gear.setSpeed(20);
        gear.leftArc(0.2, 7000);
        gear.forward(200);
        gear.leftArc(0.2, 7000);
        gear.forward(200);
        gear.leftArc(0.2, 7000);
        gear.forward(200);
        gear.leftArc(0.2, 7000);
        gear.forward(200);
        robot.exit();
    }

    public static void main(String[] args) {
        CircularGear m = new CircularGear();
        NxtContext.setStartPosition(250, 200);
```

```
        NxtContext.setStartDirection(90);
        NxtContext.setStartDirection(90);
    }
}
```

OUTPUT:



Practical 5a: Line-following robot.

AIM: Write a program for a line-following robot.

DESCRIPTION:

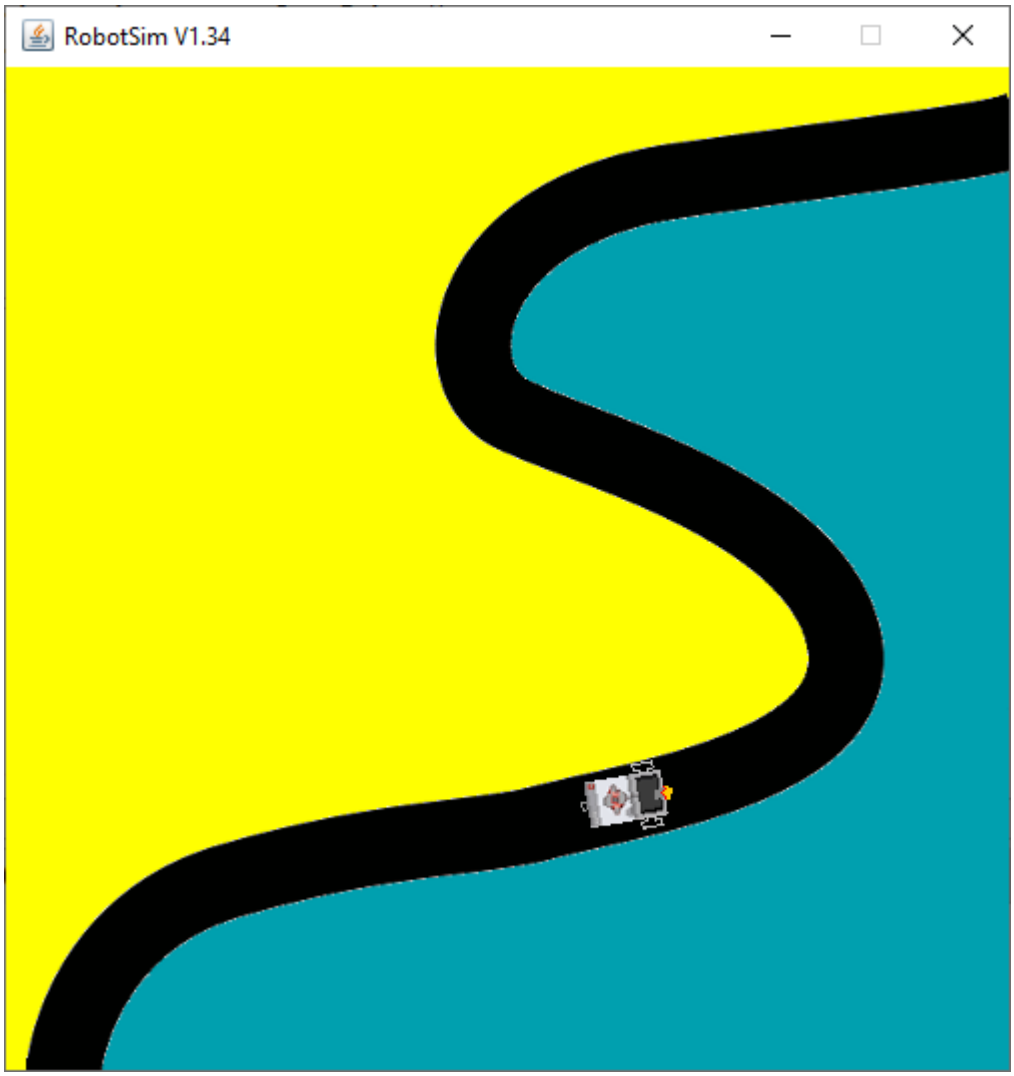
1. Gear(): This function creates an instance of a gear, with the right motor plugged into port A and the left motor plugged into port B. It seems to initialize the motors' configuration.
2. forward(int duration): This function starts a forward movement for the specified duration in milliseconds and then stops. It controls the robot's movement in the forward direction.
3. left(int duration): This function initiates a left rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the left.
4. right(int duration): Similar to the left function, this function starts a right rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the right.
5. setSpeed(int speed): This function sets the speed of the robot to the given value in arbitrary units. It allows you to control how fast the robot moves.
6. stop(): This function stops the movement of the robot. It halts any ongoing motion.
7. NxtRobot(): This function creates an instance of a turtle robot. The name "turtle robot" suggests that it might be designed for movement and potentially drawing, similar to the concept of a "turtle graphics" robot.

SOURCE CODE:

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

public class LineFollower {
    public LineFollower() {
        LegoRobot robot = new LegoRobot();
        Gear gear = new Gear();
        LightSensor ls = new LightSensor(SensorPort.S3);
        robot.addPart(gear);
        robot.addPart(ls);
        gear.setSpeed(20);

        while (true) {
            int v = ls.getValue();
            if (v < 100) {
                gear.forward();
            }
            if (v > 300 && v < 750) {
                gear.leftArc(0.05);
            }
            if (v > 800) {
                gear.rightArc(0.05);
```

```
            }
        }
    }

    public static void main(String args[]) {
        RobotContext.setStartPosition(50, 490);
        RobotContext.setStartDirection(-90);
        RobotContext.useBackground("sprites/road.gif");
        new LineFollower();
    }
}
```

OUTPUT:



Practical 5b: Line-following robot on a different image.

AIM: Write a program for a line-following robot, but for a different image.

DESCRIPTION:

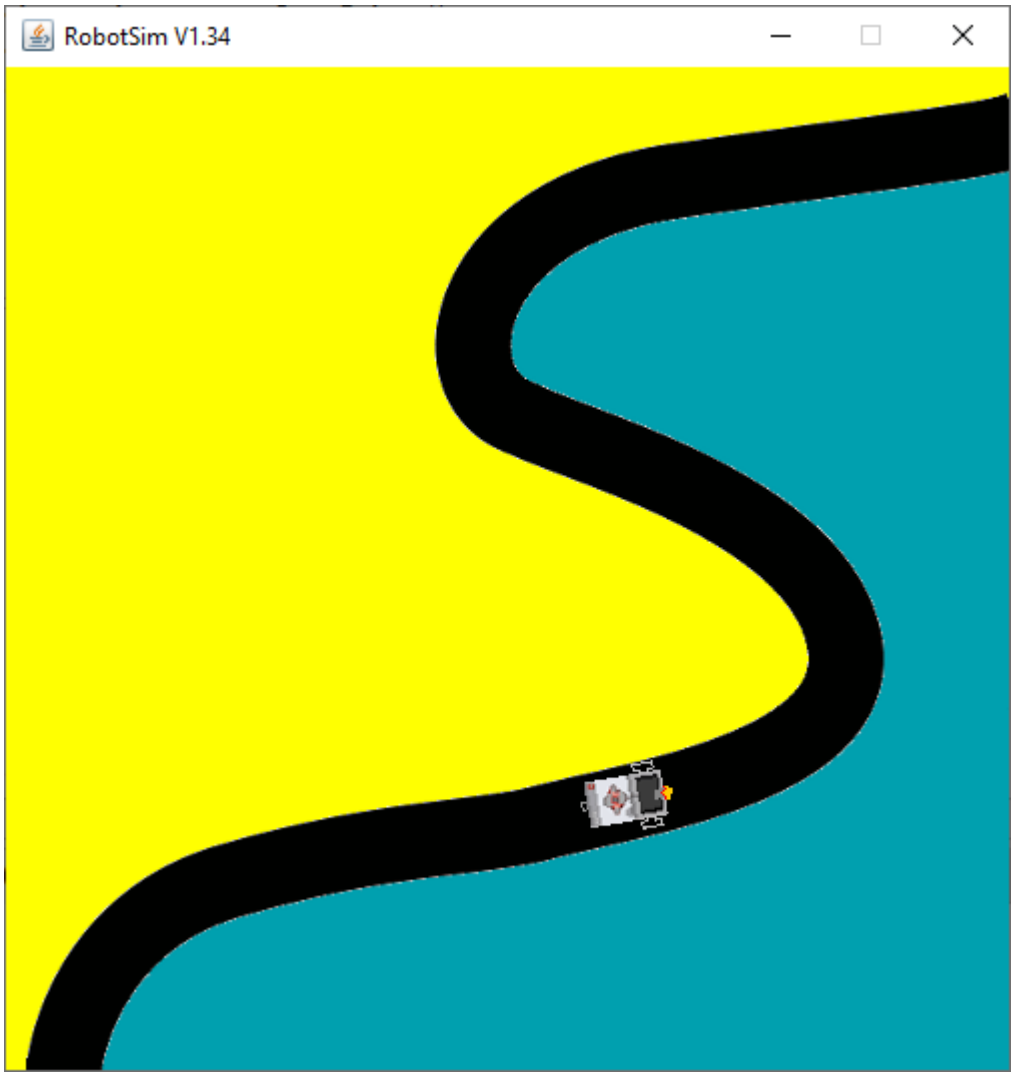
8. Gear(): This function creates an instance of a gear, with the right motor plugged into port A and the left motor plugged into port B. It seems to initialize the motors' configuration.
9. forward(int duration): This function starts a forward movement for the specified duration in milliseconds and then stops. It controls the robot's movement in the forward direction.
10. left(int duration): This function initiates a left rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the left.
11. right(int duration): Similar to the left function, this function starts a right rotation with the center of rotation at the middle of the wheel axes for the given duration in milliseconds and then stops. It causes the robot to rotate to the right.
12. setSpeed(int speed): This function sets the speed of the robot to the given value in arbitrary units. It allows you to control how fast the robot moves.
13. stop(): This function stops the movement of the robot. It halts any ongoing motion.
14. NxtRobot(): This function creates an instance of a turtle robot. The name "turtle robot" suggests that it might be designed for movement and potentially drawing, similar to the concept of a "turtle graphics" robot.

SOURCE CODE:

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

public class LineFollower {
    public LineFollower() {
        LegoRobot robot = new LegoRobot();
        Gear gear = new Gear();
        LightSensor ls = new LightSensor(SensorPort.S3);
        robot.addPart(gear);
        robot.addPart(ls);
        gear.setSpeed(20);

        while (true) {
            int v = ls.getValue();
            if (v < 100) {
                gear.forward();
            }
            if (v > 300 && v < 750) {
                gear.leftArc(0.05);
            }
            if (v > 800) {
                gear.rightArc(0.05);
```

```
            }
        }
    }

    public static void main(String args[]) {
        RobotContext.setStartPosition(50, 490);
        RobotContext.setStartDirection(-90);
        RobotContext.useBackground("sprites/road.gif");
        new LineFollower();
    }
}
```

OUTPUT:



Practical 6a: Pathfinding robot.

AIM: Write a program for a pathfinding robot.

DESCRIPTION:

1. leftArc(double radius): Starts to move to the left on an arc with given radius.

2. leftArc(double radius, int duration): Starts to move left on an arc with given radius for the given duration (in ms) and stops.

3. right(int duration): Starts to rotate right (center of rotation at middle of the wheel axes) for the given duration(in ms) and stops.

4. rightArc(double radius): Starts to move to the right on an arc with given radius.

5. void rightArc(double radius, int duration): Starts to move right on an arc with given radius for the given duration (in ms) and stops.

SOURCE CODE:

```
package robotics;

import ch.aplu.robotsim.Gear;
import ch.aplu.robotsim.LightSensor;
import ch.aplu.robotsim.NxtContext;
import ch.aplu.robotsim.NxtRobot;
import ch.aplu.robotsim.SensorPort;

public class PathFinder {
    PathFinder(){
    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();
    while(true){
    int rightValue=ls1.getValue();
    int leftValue=ls2.getValue();
    int d=rightValue-leftValue;
    if(d>100)
        gear.rightArc(0.1);
    if(d<-100)
```

```
        gear.leftArc(0.1);
    if(d>-100 && d<100 && rightValue>500)
     gear.forward();
    }

    }


    public static void main(String[] args) {
        new PathFinder();

    }
    static{
        NxtContext.setStartPosition(250,490);
        NxtContext.setStartDirection(-90);
        NxtContext.useBackground("sprites/path.gif");
    }

}
```

OUTPUT:

Practical 6b: Pathfinding robot on different image.

AIM: Write a program for a pathfinding robot, but for a different image.

DESCRIPTION:

1. leftArc(double radius): Starts to move to the left on an arc with given radius.

2. leftArc(double radius, int duration): Starts to move left on an arc with given radius for the given duration (in ms) and stops.

3. right(int duration): Starts to rotate right (center of rotation at middle of the wheel axes) for the given duration(in ms) and stops.

4. rightArc(double radius): Starts to move to the right on an arc with given radius.

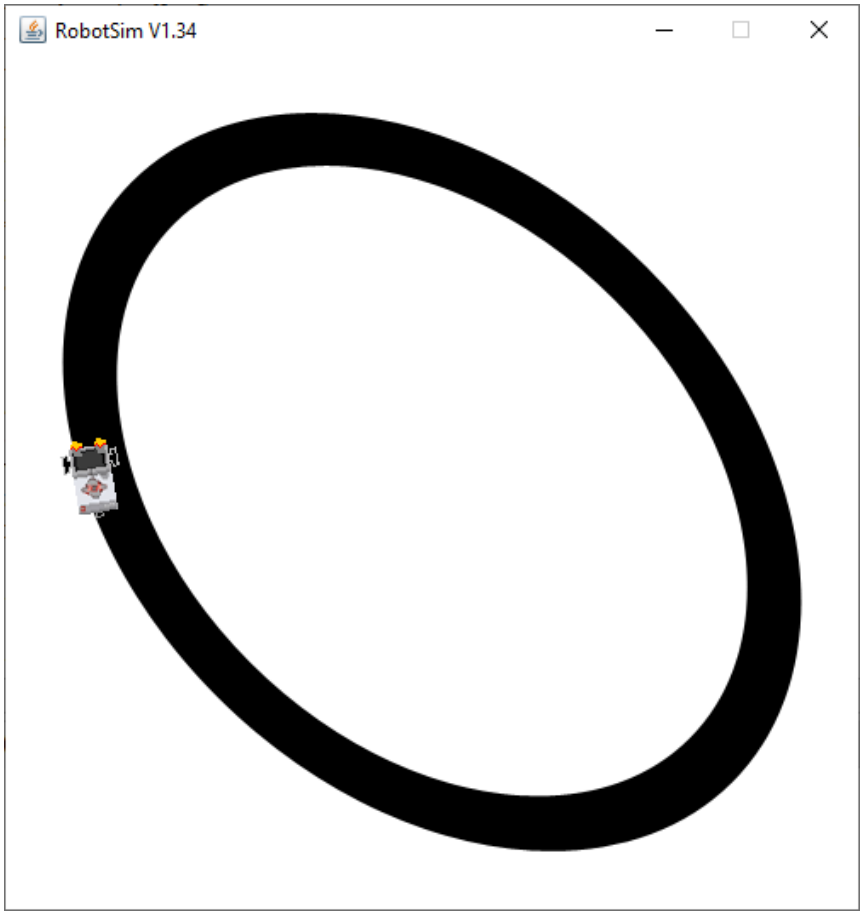
5. void rightArc(double radius, int duration): Starts to move right on an arc with given radius for the given duration (in ms) and stops.

SOURCE CODE:

```
package robotics;

import ch.aplu.robotsim.Gear;
import ch.aplu.robotsim.LightSensor;
import ch.aplu.robotsim.NxtContext;
import ch.aplu.robotsim.NxtRobot;
import ch.aplu.robotsim.SensorPort;

public class PathFinder {
    PathFinder(){
    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();
    while(true){
    int rightValue=ls1.getValue();
    int leftValue=ls2.getValue();
    int d=rightValue-leftValue;
    if(d<-100)
        gear.rightArc(0.1);
    if(d>100)
```

```
        gear.leftArc(0.1);
    if(d>-100 && d<100 && rightValue<500)
     gear.forward();
    }

    }


    public static void main(String[] args) {
        new PathFinder();

    }
    static{
        NxtContext.setStartPosition(255,460);
        NxtContext.setStartDirection(-90);
        NxtContext.useBackground("sprites/roundPath.gif");
    }

}
```

OUTPUT:



Practical 7: Obstacle resistance using touch sensor.

AIM: Write a program for obstacle resistance using a touch sensor.

DESCRIPTION:

1. setStartDirection(double direction): Sets the Nxt starting direction (zero to EAST).

2. static void setStartPosition(int x, int y): Sets the Nxt starting position (x-y-coordinates 0..500, origin at upper left).

3. useBackground(java.lang.String filename): Use the given image as background (playground size 501 x 501).

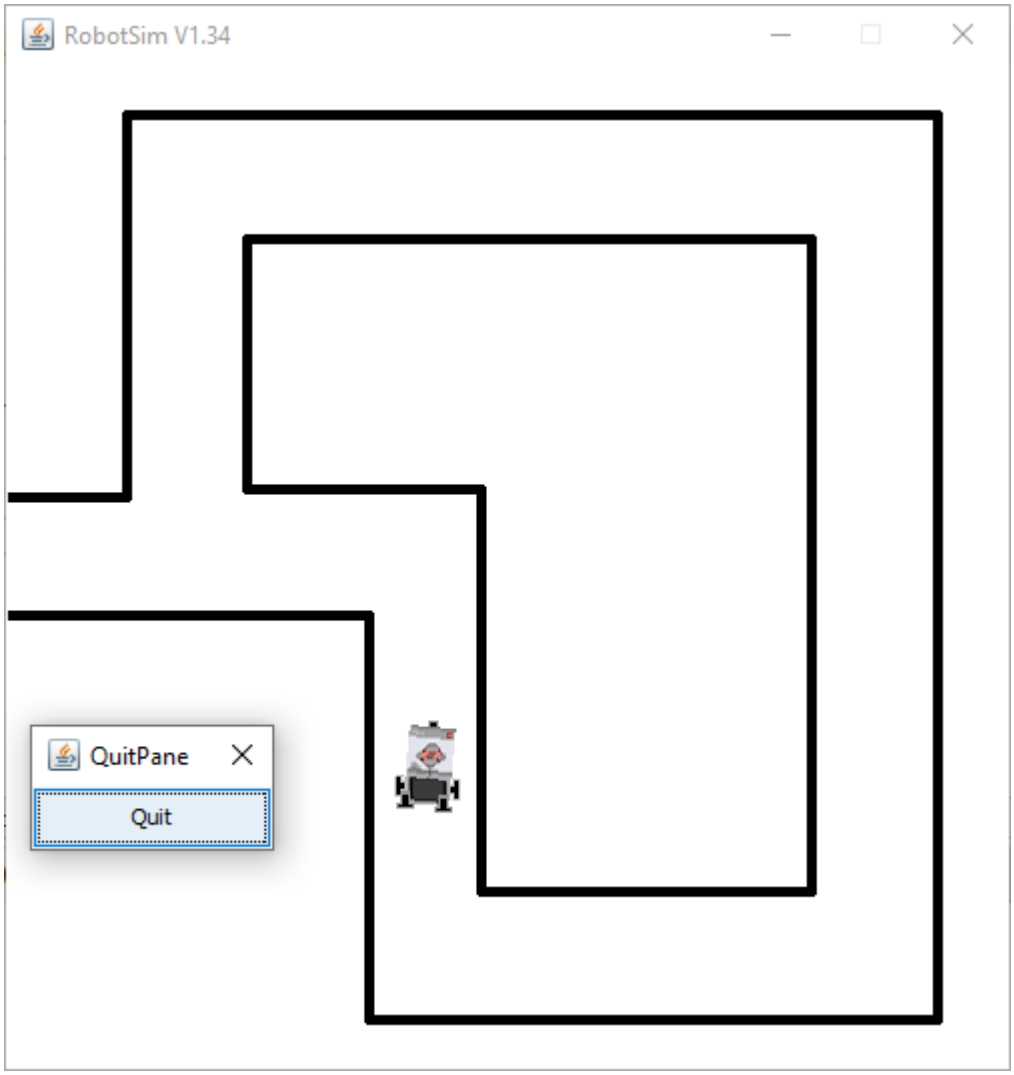
4. TouchSensor(SensorPort port): Creates a sensor instance connected to the given port

SOURCE CODE:

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

public class resistobst
{
    public resistobst()
    {
        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 resistobst();
    }
    static
    {
        RobotContext.setLocation(10,10);
        RobotContext.setStartDirection(5);
        RobotContext.setStartPosition(100,240);
        RobotContext.useObstacle(RobotContext.channel);
    }
}
```

OUTPUT:


RobotSim V1.34
QuitPane
Quit

Practical 8: Robot with ultrasonic sensor.

AIM: Write a program to create a robot with ultrasonic sensor.

DESCRIPTION:

1. UltrasonicSensor(SensorPort port): The port selection determines the position of the sensor and the direction of the beam axis.

2. setProximityCircleColor(java.awt.Color color): Sets the color of the circle with center at sensor location and radius equals to the current distance value.

3. useTarget(GGBitmap bm, java.awt.Point[] mesh, int x, int y): Creates a target for the ultrasonic sensor using the given GGBitmap.

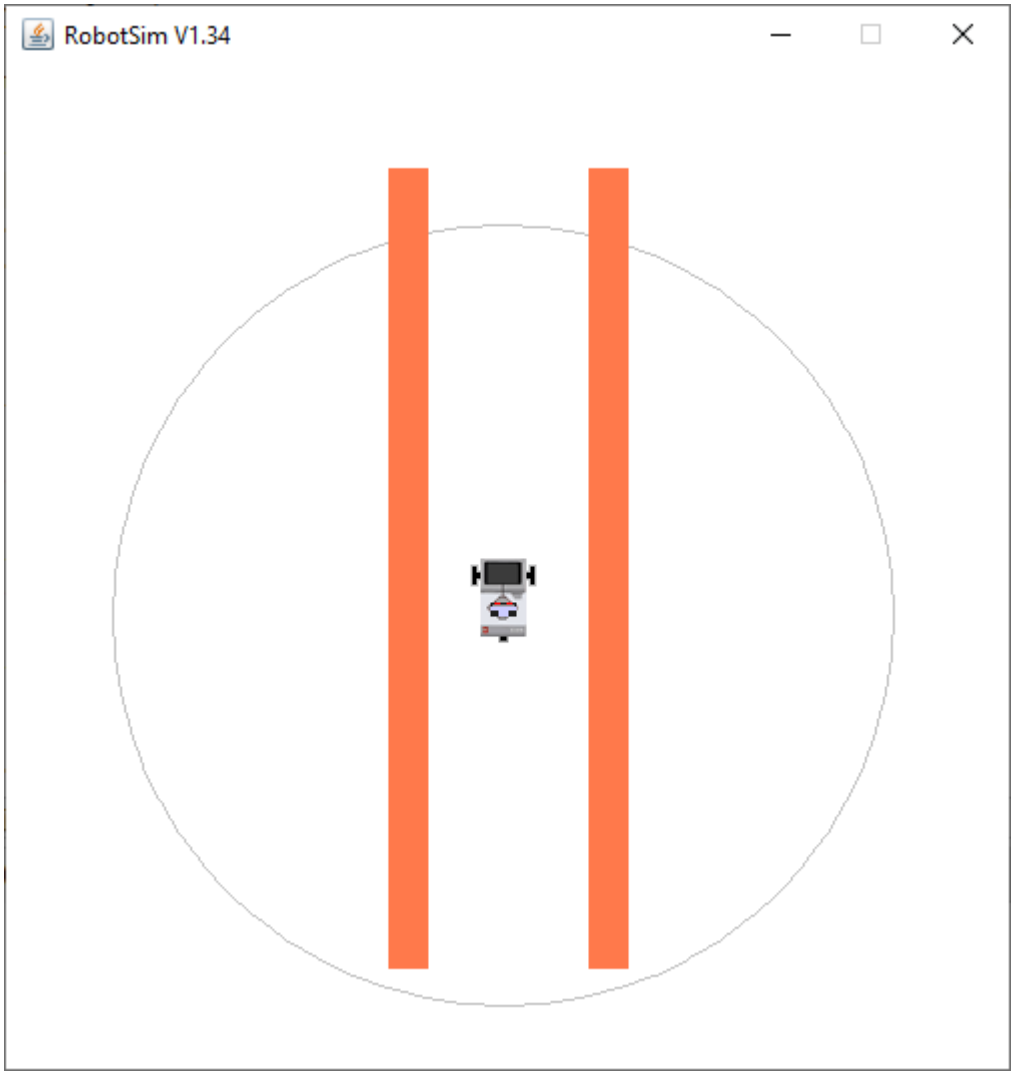
4. setStartPosition(int x, int y): Sets the Nxt starting position (x-y-coordinates 0..500, origin at upper left).

SOURCE CODE:

```
package Robotics;
import ch.aplu.robotsim.*;
import java.awt.Color;
import java.awt.Point;
import java.awt.color.*;

public class Ultra_Ex {

    public Ultra_Ex() {

        LegoRobot robot = new LegoRobot();
        Gear gear = new Gear();
        robot.addPart(gear);
        UltrasonicSensor us = new UltrasonicSensor(SensorPort.S1);
        robot.addPart(us);
        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 Ultra_Ex();
    }
}
```

OUTPUT:


RobotSim V1.34

Practical 9: Torch follower.

AIM: Write a program to create a torch-following robot.

DESCRIPTION:

1. UltrasonicSensor(SensorPort port): The port selection determines the position of the sensor and the direction of the beam axis.

2. setProximityCircleColor(java.awt.Color color): Sets the color of the circle with center at sensor location and radius equals to the current distance value.

3. useTarget(GGBitmap bm, java.awt.Point[] mesh, int x, int y): Creates a target for the ultrasonic sensor using the given GGBitmap.

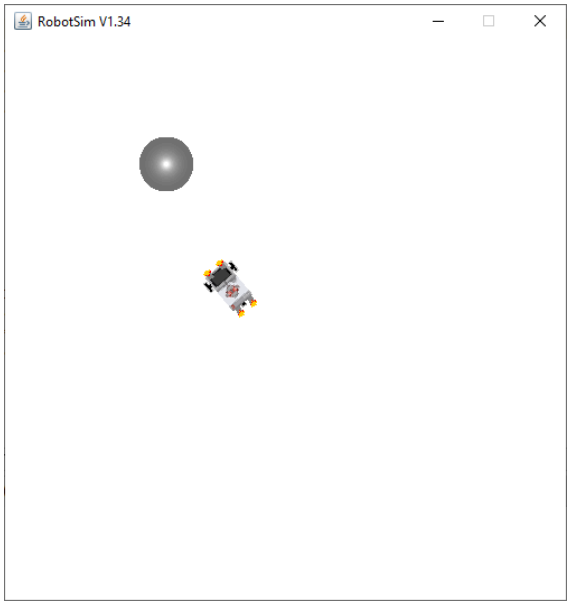
4. setStartPosition(int x, int y): Sets the Nxt starting position (x-y-coordinates 0..500, origin at upper left).

SOURCE CODE:

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

public class TorchFollower
{
  public 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)
        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)
  {
    new TorchFollower();
  }

  static
  {
    RobotContext.useTorch(1, 150, 250, 100);
  }
}
```

OUTPUT:



Practical 10: Shadow follower.

AIM: Write a program to create a shadow-following robot.

DESCRIPTION:

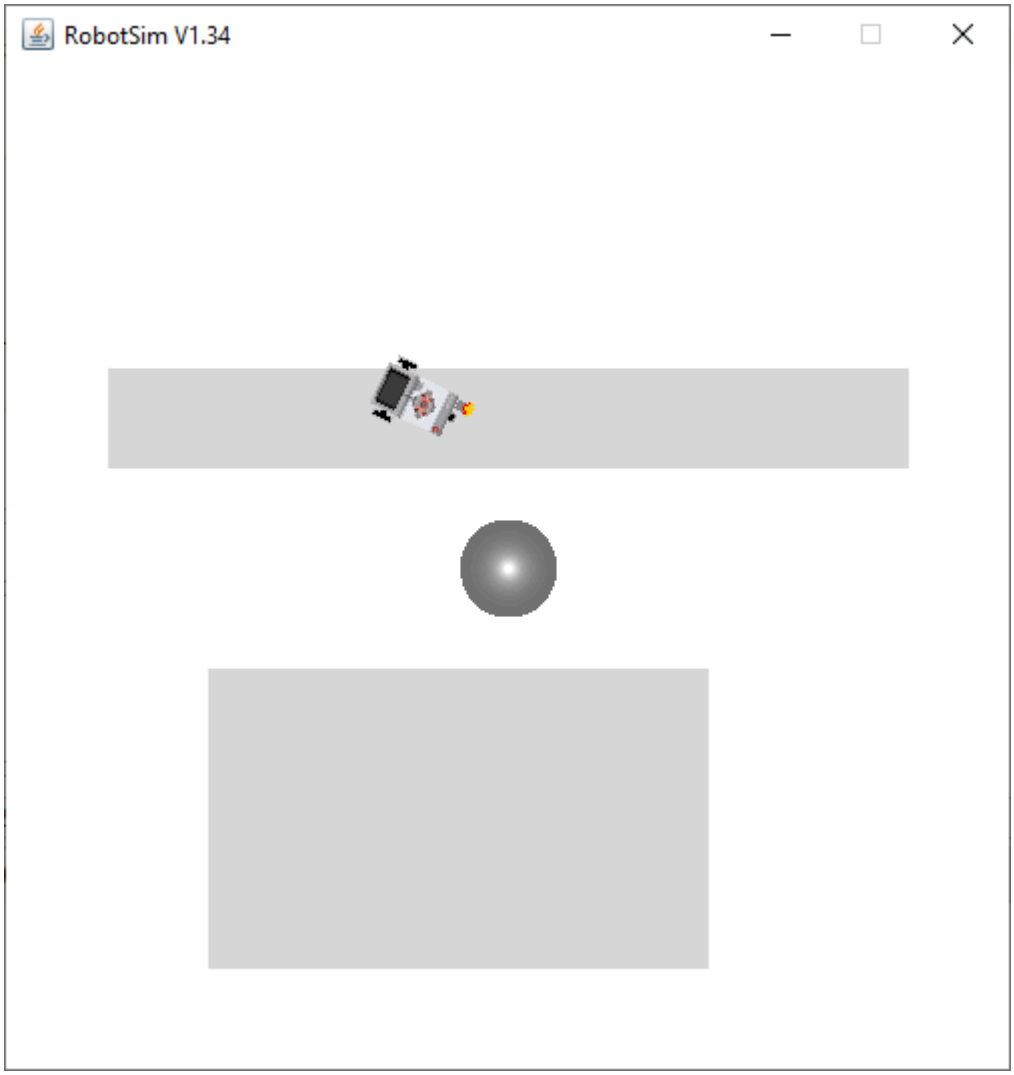
1. LegoRobot robot = new LegoRobot();: This line creates an instance of a LEGO robot.
2. LightSensor ls = new LightSensor(SensorPort.S3, true): This line creates a light sensor and associates it with a specific sensor port on the robot (SensorPort.S3) and sets it to active mode (true).
3. robot.addPart(ls): Adds the light sensor as a part of the robot, allowing it to interact with the robot's environment.
4. Gear gear = new Gear(): Creates an instance of the Gear class, which represents the robot's motorized gear or wheels.
5. robot.addPart(gear;: Adds the gear to the robot, allowing control over the robot's movement.
6. gear.leftArc(0.5): This line sets the robot to perform a left arc with a speed of 0.5. It makes the robot move in a curve to the left.
7. while (!robot.isEscapeHit()) { ... }: This is a loop that runs as long as the escape key is not pressed (robot.isEscapeHit() returns false). Inside the loop, the code reads the light sensor value, checks for changes in light conditions, and plays different tones accordingly.
8. int v = ls.getValue(): Reads the value from the light sensor, which represents the light intensity detected by the sensor.
9. Conditional statements inside the loop (if (!isLightOn && v == 0) and if (isLightOn && v > 0)) check for changes in light conditions and play different tones when the light conditions change.
10. Tools.delay(100): Pauses the execution of the loop for 100 milliseconds, controlling the loop's execution speed.
11. robot.exit(): Exits the robot simulation when the loop is terminated.
12. public static void main(String[] args): The main method of the program, where an instance of the SimEx20 class is created, initializing the robot and starting the simulation.
13. static { ... }: A static block that sets up the simulation environment using the RobotContext.useTorch method. It configures a torch (light source) in the simulated environment to interact with the light sensor.

SOURCE CODE:

```
import ch.aplu.robotsim.*;
import java.awt.Color;

public class SimEx20
{
  public SimEx20()
  {
    LegoRobot robot = new LegoRobot();
    LightSensor ls = new LightSensor(SensorPort.S3, true);
    robot.addPart(ls);
    Gear gear = new Gear();
    robot.addPart(gear);

    gear.leftArc(0.5);
```

```
    boolean isLightOn = false;
    while (!robot.isEscapeHit())
    {
      int v = ls.getValue();
      if (!isLightOn && v == 0)
      {
        isLightOn = true;
        robot.playTone(2000, 100);
      }
      if (isLightOn && v > 0)
      {
        isLightOn = false;
        robot.playTone(1000, 100);
      }
      Tools.delay(100);
    }
    robot.exit();
  }

  public static void main(String[] args)
  {
    new SimEx20();
  }

  // ------------------ Environment --------------------------
  static
  {
    RobotContext.useTorch(1, 250, 250, 100);
    RobotContext.useShadow(50, 150, 450, 200);
    RobotContext.useShadow(100, 300, 350, 450);
  }
}
```

OUTPUT:


RobotSim V1.34