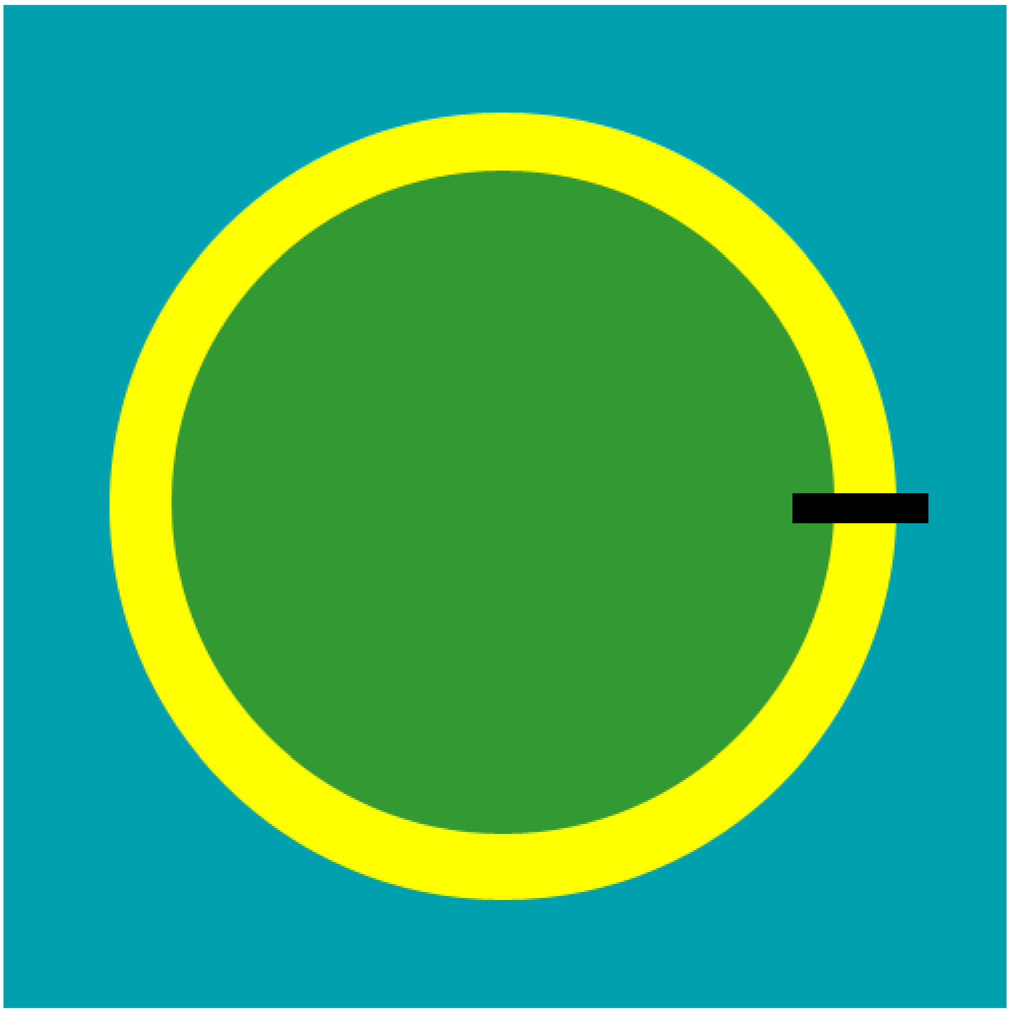
Aim: Create a robot which will move only on the yellow path in given image.



Description:

- Gear: Combines two motors on an axis to perform a car-like movement.
  - **Constructor:**
    - Gear(): Creates a gear instance with right motor plugged into port A, left motor plugged into port B.
  - **Methods:**
    - void forward(): Starts the forward movement.
    - void leftArc(double *radius*): Starts to move to the left on an arc with given radius.

        - void rightArc(double *radius*): Starts to move to the right on an arc with given radius.
        - void setSpeed(int *speed*): Sets the speed to the given value (arbitrary units).
        - void stop(): Stops the movement.

- LegoRobot: Class that represents a simulated NXT or EV3 robot brick. Parts (e.g. motors, sensors) may be assembled into the robot to make it doing the desired job. Each instance creates its own square playground (501 x 501 pixels). Some initial conditions may be modified by calling static methods of the class RobotContext in a static block.
    - **Constructor**:
        - LegoRobot(): Creates a robot with its playground using defaults from RobotContext.

    - **Methods**:
        - void addPart(Part *part*): Assembles the given part into the robot.

- LightSensor: Class that represents a light sensor
    - **Constructor**:
        - LightSensor(SensorPort port): Creates a sensor instance pointing downwards connected to the given port.

    - **Methods**:
        - int getValue(): For sensor ports 1, 2, 3, 4: returns the brightness of the background at the current location.

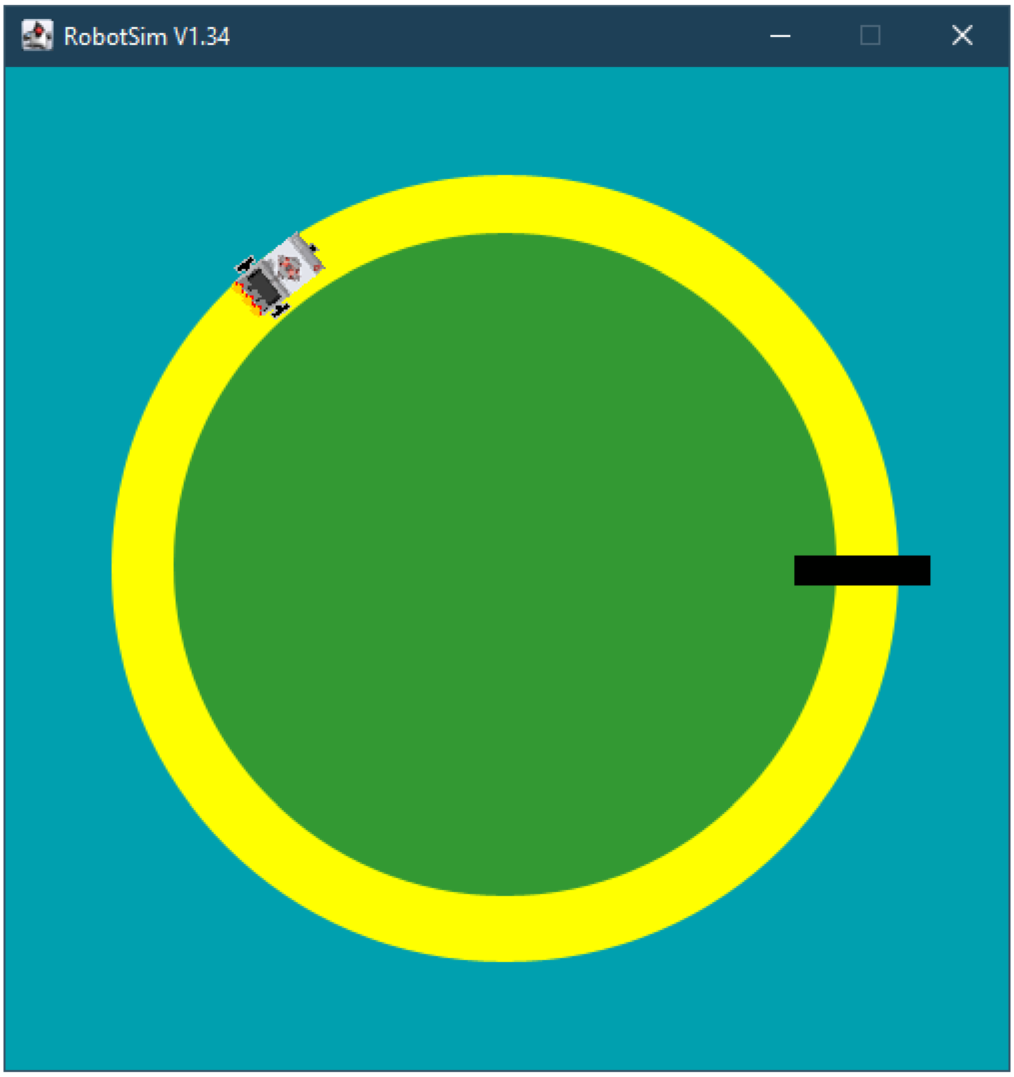
- RobotContext: Class to select user defined initial conditions of the playground and the Nxt or EV3 robot.
    - **Methods**:
        - static void setStartDirection(double direction): Sets the Nxt starting direction (zero to EAST).
        - static void setStartPosition(int x, int y): Sets the Nxt starting position ($x, y \in [0, 500]$, origin at upper left).
        - static void useBackground(String filename): Use the given image as background (playground size 501 x 501).

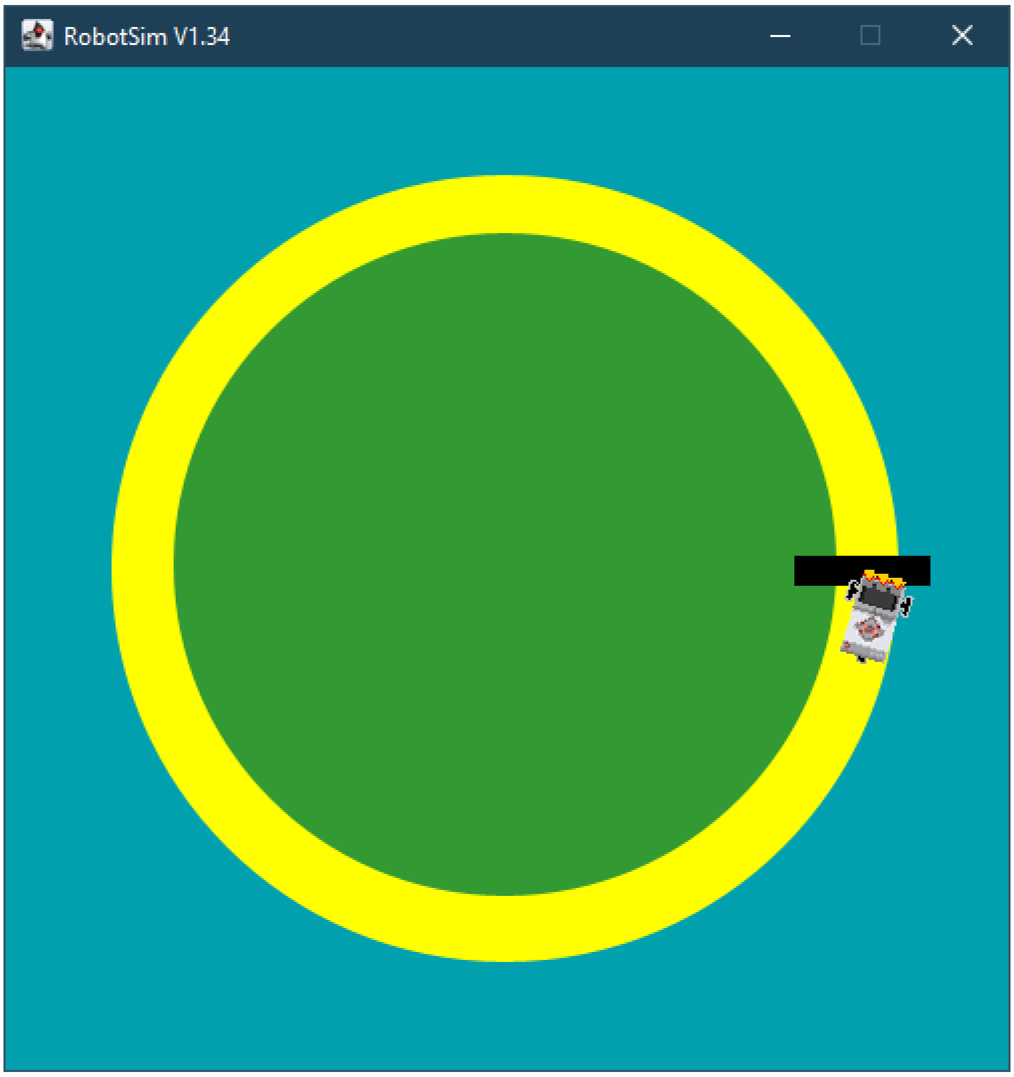
- SensorPort: Useful declarations for sensor port connections.
    - Fields:
        - static SensorPort *S1*: A sensor port for a sensor connected to port *S1*.
        - static SensorPort *S2*: A sensor port for a sensor connected to port *S2*.
        - static SensorPort *S3*: A sensor port for a sensor connected to port *S3*.

Source Code:

```java
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 CircularPathFollowerRobot {
static {
RobotContext.useBackground("sprites/yellowpath.gif");
RobotContext.setStartPosition(430,230);
RobotContext.setStartDirection(-90);
}

public CircularPathFollowerRobot() {
// Initialize required components and add them
// to the robot.
LegoRobot legoRobot = new LegoRobot();
Gear gear = new Gear();
LightSensor lightSensorL = new LightSensor(SensorPort.S2);
LightSensor lightSensorR = new LightSensor(SensorPort.S1);
LightSensor lightSensorM = new LightSensor(SensorPort.S3);
legoRobot.addPart(gear);
legoRobot.addPart(lightSensorL);
legoRobot.addPart(lightSensorR);
legoRobot.addPart(lightSensorM);
gear.forward();
gear.setSpeed(100);

double arcLength = 0.1;

while (true) {
int lightSensorRValue = lightSensorR.getValue();
int lightSensorDiff = lightSensorRValue - lightSensorL.getValue();

if (lightSensorM.getValue() < 100) {
gear.stop();
}

if(lightSensorDiff > 100) {
gear.rightArc(arcLength);
}
else if (lightSensorDiff < -100) {
gear.leftArc(arcLength);
}
else {
if (lightSensorRValue > 500) {
gear.forward();
}
}
}
}

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

Output:




RobotSim V1.34