```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

/**
 *
 * @author Eddy
 */
public abstract class Driver {

  protected String name;
  protected Vehicle vehicle;

  public Driver(String name){
    this.name = name;
  }

  public void setVehicle(Vehicle vehicle){
    this.vehicle = vehicle;
  }

  abstract public double getOptimalDeceleration();

  public double getOptimalSpeedForDistance(double distance) {
    double speed = getOptimalDeceleration() * distance;

    // Capping for max speed
    if (speed > vehicle.getTopSpeed()) {
      speed = vehicle.getTopSpeed();
    }

    return speed;
  }

  public double getOptimalFollowingDistance() {
    double stoppingDistance = vehicle.getCurrentSpeed() / getOptimalDeceleration
();
    return 30.0 + stoppingDistance;
  }

  public boolean AccelerationStatus(double currentSpeed, double optimalFollowing
Dist, double distanceFromNextVechicle, double distanceFromEOLane) {
    boolean choice;
    //no car ahead
    if (distanceFromEOLane == Double.MAX_VALUE) {
      choice = true;
    }
    if (distanceFromNextVechicle <= optimalFollowingDist) {
      choice = false;
    } else {
      choice = true;
    }

    return choice;
  }
```

```
  public boolean DecelerationStatus(double currentSpeed, double optimalFollowing
Dist, double distanceFromNextVechicle, double distanceFromEOLane) {
    boolean choice;
    if (distanceFromEOLane == Double.MAX_VALUE) {
      //This will depend on the state of the traffic light
    }
    if (distanceFromNextVechicle <= optimalFollowingDist) {
      choice = true;
    } else {
      choice = false;
    }

    return choice;

  }

}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 *
 * @author balazs
 */
public class EntryPoint implements ISteppable{

  private Lane lane;
  private Map<Long, List<Vehicle>> steps = new HashMap<>();
  private Map<Vehicle, Long> vehicles = new HashMap<>();

  public EntryPoint(Lane lane) {
    this.lane = lane;
  }

  public Lane getLane() {
    return lane;
  }

  public void addVehicle(Vehicle vehicle, long step) {
    vehicles.put(vehicle, step);

    List stepList = steps.get(step);
    if (stepList == null) {
      stepList = new ArrayList<Vehicle>();
      steps.put(step, stepList);
    }
    stepList.add(vehicle);
    vehicle.startTime = System.currentTimeMillis();
  }

  public int numberOfVehicles() {
    return vehicles.size();
  }

  public void step(long step) {
    List<Vehicle> vehiclesForStep = steps.get(step);
    if (vehiclesForStep == null) {
      return;
    }
    for (Vehicle vehicle : vehiclesForStep) {
      //Add vehicle to system
      System.out.println(vehicle + " entered the system");
      vehicle.setLane(lane);
    }
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author balazs
 */
public class ExitPoint {

  private Lane lane;
  private List<Vehicle> vehicles = new ArrayList<>();

  ExitPoint(Lane lane) {
    this.lane = lane;
  }

  public int numberOfVehicles() {
    return vehicles.size();
  }

  void addVehicle(Vehicle vehicle) {
    System.out.println(vehicle + " exited the system");
    vehicles.add(vehicle);
    vehicle.endTime = System.currentTimeMillis();
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

/**
 *
 * @author balazs
 */
public interface ISteppable {
  public void step(long step);
}
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 *
 * @author balazs
 */
public abstract class Junction implements ISteppable{

  private HashMap<Lane, List<Lane>> connections;

  public Junction() {
    connections = new HashMap<>();
  }

  public void connect(Lane source, Lane destination) {
    if (!connections.containsKey(source)) {
      connections.put(source, new ArrayList<Lane>());
    }
    List<Lane> lanes = connections.get(source);
    lanes.add(destination);
    source.setJunction(this);
  }

  public List<Lane> getConnectedLanes(Lane lane) {
    return connections.get(lane);
  }

  public boolean shouldVehicleEnterJunction(Vehicle vehicle){
    return true;
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.List;
import trafficsimulator.utils.Point;

/**
 *
 * @author balazs
 */
public class Lane {

  public static double laneWidth = 22;

  public enum Direction {

    IDENTICAL, OPPOSITE
  }

  private Road road;
  private List<Vehicle> vehicles = new ArrayList<>();
  private Junction junction;
  private Direction direction;
  private ExitPoint exitPoint;

  public Lane(Direction direction) {
    this.direction = direction;
    exitPoint = new ExitPoint(this);
  }

  public void enter(Vehicle vehicle) {
    vehicles.add(vehicle);
  }

  public void exit(Vehicle vehicle) {
    vehicles.remove(vehicle);
  }

  public Junction getJunction() {
    return junction;
  }

  public void setJunction(Junction junction) {
    this.exitPoint = null;
    this.junction = junction;
  }

  public ExitPoint getExitPoint() {
    return exitPoint;
  }

  public Road getRoad() {
    return road;
  }
```

```
  public void setRoad(Road road) {
    this.road = road;
  }

  public Direction getDirection() {
    return direction;
  }

  public void setDirection(Direction direction) {
    this.direction = direction;
  }

  public Point getLeftStartPoint() {
    Road road = getRoad();
    int pos = road.getLaneIndexPosition(this);
    if (getDirection() == Direction.IDENTICAL) {
      return road.getLeftStartPoint().plus(acrossLaneVector().mult(pos));
    } else {
      return road.getLeftEndPoint().minus(acrossLaneVector().mult(pos + 1));
    }
  }

  public Point getLeftEndPoint() {
    Road road = getRoad();
    int pos = road.getLaneIndexPosition(this);
    if (getDirection() == Direction.IDENTICAL) {
      return road.getLeftEndPoint().plus(acrossLaneVector().mult(pos));
    } else {
      return road.getLeftStartPoint().minus(acrossLaneVector().mult((pos + 1)));
    }
  }

  private Point calculateRightPoints(Point p) {
    return p.plus(acrossLaneVector());
  }

  public Point getRightStartPoint() {
    return calculateRightPoints(getLeftStartPoint());
  }

  public Point getRightEndPoint() {
    return calculateRightPoints(getLeftEndPoint());
  }

  public Point getCenterStartPoint() {
    return (getLeftStartPoint().plus(getRightStartPoint())).div(2);
  }

  public Point getCenterEndPoint() {
    return (getLeftEndPoint().plus(getRightEndPoint())).div(2);
  }

  public Point getDirectionVector() {
    Road road = getRoad();
    if (getDirection() == Direction.IDENTICAL) {
      return road.getLeftEndPoint().minus(road.getLeftStartPoint());
    } else {
      return road.getLeftStartPoint().minus(road.getLeftEndPoint());
```

```
    }
  }

  private Point acrossLaneUnitVector() {
    Point dir = getDirectionVector();
    Point unitDir = dir.div(dir.distanceFromOrigin());
    Point rotateUnitDir = unitDir.rotateVector(Math.PI / 2);
    return rotateUnitDir;
  }

  private Point acrossLaneVector() {
    double x = Math.floor(laneWidth * Math.cos(acrossLaneUnitVector().angleVecto
r()));
    double y = Math.floor(laneWidth * Math.sin(acrossLaneUnitVector().angleVecto
r()));
    return new Point(x, y);
  }

  public double getDistanceFromNextVehicle(Vehicle vehicle) {
    double minDistance = Double.MAX_VALUE;

    for (Vehicle v : vehicles) {
      if (vehicle == v) {
        continue;
      }

      double distance = vehicle.getPosition().distance(v.getPosition());
      if (distance < minDistance) {
        Point dir = v.getPosition().minus(vehicle.getPosition());
        if (dir.inSameQuadrant(getDirectionVector())) {
          minDistance = distance;
        }
      }
    }

    return minDistance;
  }

}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author balazs
 */
public class Map {

  private List<Road> roads;
  private List<Junction> junctions;

  public Map() {
    roads = new ArrayList<>();
    junctions = new ArrayList<>();
  }

  public List<Road> getRoads() {
    return roads;
  }

  public void addRoad(Road road) {
    roads.add(road);
  }

  public List<Junction> getJunctions() {
    return junctions;
  }

  public void addJunction(Junction junction) {
    junctions.add(junction);
  }

}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.List;
import trafficsimulator.utils.Point;

/**
 *
 * @author balazs
 */
public class Road {

  private List<Lane> lanes;
  //The road is initialised by specifying the left paramiters of the road.
  //Each lane will be added to the right these paramiters and the right
  //paramiters of the road will be calculated by the numbers of lanes on the roa
d.
  private Point leftStartPoint;
  private Point leftEndPoint;

  public Road(Point leftStartPoint, Point leftEndPoint) {
    lanes = new ArrayList<>();
    this.leftStartPoint = leftStartPoint;
    this.leftEndPoint = leftEndPoint;
  }

  public void addLane(Lane lane) {
    lanes.add(lane);
    lane.setRoad(this);
  }

  public List<Lane> getLanes() {
    return lanes;
  }

  public void setLanes(List<Lane> lanes) {
    this.lanes = lanes;
  }

  public Point getLeftStartPoint() {
    return leftStartPoint;
  }

  public void setLeftStartPoint(Point leftStartPoint) {
    this.leftStartPoint = leftStartPoint;
  }

  public Point getLeftEndPoint() {
    return leftEndPoint;
  }

  public void setLeftEndPoint(Point leftEndPoint) {
    this.leftEndPoint = leftEndPoint;
  }
```

```
  public Point getRandomPosition() {
    Point dir = leftEndPoint.minus(leftStartPoint);
    return leftStartPoint.plus(dir.mult(Math.random()));
  }

  public Point getDirectionVector() {
    return leftEndPoint.minus(leftStartPoint);
  }

  public int getLaneIndexPosition(Lane l) {
    return lanes.indexOf(l);
  }

  public double calculateWidth() {
    double width = 0;
    for (Lane l : lanes) {
      width += Lane.laneWidth;
    }
    return width;
  }

  private Point acrossRoadUnitVector() {
    Point dir = getDirectionVector();
    Point unitDir = dir.div(dir.distanceFromOrigin());
    Point rotateUnitDir = unitDir.rotateVector(Math.PI / 2);
    return rotateUnitDir;
  }

  private Point acrossRoadVector() {
    double x = Math.round(calculateWidth() * Math.cos(acrossRoadUnitVector().ang
leVector()));
    double y = Math.round(calculateWidth() * Math.sin(acrossRoadUnitVector().ang
leVector()));
    return new Point(x, y);
  }

  public Point getRightStartPoint() {
    Point rightStartPoint = leftStartPoint.plus(acrossRoadVector());
    return rightStartPoint;
  }

  public Point getRightEndPoint() {
    Point rightEndPoint = leftEndPoint.plus(acrossRoadVector());
    return rightEndPoint;
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;
import javafx.scene.text.Text;
import trafficsimulator.gui.IRenderer;

/**
 *
 * @author balazs
 */
public abstract class Simulation extends TimerTask {

  private long stepCounter = 0;
  protected Timer timer = new Timer();
  protected Map map = new Map();
  protected List<Vehicle> vehicles = new ArrayList<>();
  protected List<EntryPoint> entryPoints = new ArrayList<>();
  protected List<ExitPoint> exitPoints = new ArrayList<>();
  protected IRenderer renderer;

  public Simulation() {

  }

  public Simulation(IRenderer renderer) {
    this.renderer = renderer;
  }

  protected abstract void init();

  @Override
  public void run() {

    stepCounter++;
    System.out.println("Step " + stepCounter);

    if (numberOfVehiclesAtExitPoints() == vehicles.size()) {
      printStats();
      System.out.println("Simulation end");
      timer.cancel();
      return;
    }

    for (ISteppable ep : entryPoints) {
      ep.step(stepCounter);
    }

    for (ISteppable junction : map.getJunctions()) {
      junction.step(stepCounter);
    }
```

```
    for (ISteppable vehicle : getVehicles()) {
      vehicle.step(stepCounter);
    }

    if (renderer != null) {
      renderer.render();
    }
  }

  private EntryPoint getEntryPointForLane(Lane lane) {
    for (EntryPoint ep : entryPoints) {
      if (ep.getLane() == lane) {
        return ep;
      }
    }
    EntryPoint ep = new EntryPoint(lane);
    entryPoints.add(ep);
    return ep;
  }

  protected void addVehicle(Vehicle vehicle, Lane lane, long step) {
    EntryPoint ep = getEntryPointForLane(lane);
    ep.addVehicle(vehicle, step);
    vehicles.add(vehicle);
  }

  private List<ExitPoint> getExitPoints() {
    List<ExitPoint> exitPoints = new ArrayList<>();
    for (Road road : getMap().getRoads()) {
      for (Lane lane : road.getLanes()) {
        ExitPoint ep = lane.getExitPoint();
        if (ep == null) {
          continue;
        }
        exitPoints.add(ep);
      }
    }
    return exitPoints;
  }

  private int numberOfVehiclesAtExitPoints() {
    int n = 0;
    for (ExitPoint ep : exitPoints) {
      n += ep.numberOfVehicles();
    }
    return n;
  }

  public void start() {
    init();
    this.exitPoints = getExitPoints();
    timer.scheduleAtFixedRate(this, 0, 100);
  }

  public IRenderer getRenderer() {
    return renderer;
  }
```

```
  public void setRenderer(IRenderer renderer) {
    this.renderer = renderer;
  }

  public Map getMap() {
    return map;
  }

  public List<Vehicle> getVehicles() {
    List<Vehicle> vehiclesInSystem = new ArrayList<>();
    for (Vehicle vehicle : vehicles) {
      if (!vehicle.isInSystem()) {
        continue;
      }
      vehiclesInSystem.add(vehicle);
    }
    return vehiclesInSystem;
  }

  public List<Vehicle> getExitedVehicles() {
    List<Vehicle> vehiclesInSystem = new ArrayList<>();
    for (Vehicle vehicle : vehicles) {
      if (vehicle.isInSystem()) {
        continue;
      }
      vehiclesInSystem.add(vehicle);
    }
    return vehiclesInSystem;
  }

  public void printStats() {
    for (Vehicle vehicle : getExitedVehicles()) {
      System.out.println(vehicle.getType() + " was in the system for " + vehicle.timeSpe
ntInSystem() + " seconds");
    }
  }

  public Text averageTime() {
    double total = 0;
    double average = 0;
    for (Vehicle vehicle : getExitedVehicles()) {
      total += vehicle.timeSpentInSystem();
    }
    average = total/getExitedVehicles().size();

    if ( getExitedVehicles().isEmpty() ) return new Text("Average time: 0");
    else return new Text(String.valueOf("Average time: " + average));
  }

  public Text longestTime() {
    double longest = 0;
    for (Vehicle vehicle : getExitedVehicles()) {
      if (longest < vehicle.timeSpentInSystem()) {
        longest = vehicle.timeSpentInSystem();
      }
    }
    if ( getExitedVehicles().isEmpty() ) return new Text("Longest time: 0");
    else return new Text(String.valueOf("Longest time: " + longest));
  }
```

```
  public Text shortestTime() {
    double shortest = Integer.MAX_VALUE;
    for (Vehicle vehicle : getExitedVehicles()) {
      if (shortest > vehicle.timeSpentInSystem()) {
        shortest = vehicle.timeSpentInSystem();
      }
    }
    if ( getExitedVehicles().isEmpty() ) return new Text("Shortest time: 0");
    else return new Text(String.valueOf("Shortest time: " + shortest));
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

/**
 *
 * @author balazs
 */
public class TrafficSimulator {

  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {

  }

}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.core;

import java.util.List;
import java.util.Random;
import trafficsimulator.utils.Point;
import trafficsimulator.utils.Size;

/**
 *
 * @author balazs
 */
public abstract class Vehicle implements ISteppable{

  private Lane lane;
  private Point position;
  private double currentSpeed = 0;
  protected double topSpeed;
  protected double maxAcceleration;
  protected double maxDeceleration;
  protected double optimalDeceleration;
  protected Size size;
  protected Driver driver;
  protected boolean accelerate;
  protected boolean decelerate;

  protected String type = "Vehicle Base Object";
  public long startTime = 0;
  public long endTime = 0;

  public Vehicle(Driver driver) {
    this.currentSpeed = 0;
    if (driver == null) {
      this.driver = NormalDriver("Default Driver");
    } else {
      this.driver = driver;
    }
    this.driver.setVehicle(this);
  }

  public Size getSize() {
    return size;
  }

  public double getTopSpeed() {
    return topSpeed;
  }

  public double getMaxAcceleration() {
    return maxAcceleration;
  }

  public double getMaxDeceleration() {
    return maxDeceleration;
  }
```

```
  public String getType() {
    return type;
  }

  public Point getPosition() {
    return position;
  }

  public Lane getLane() {
    return lane;
  }

  public boolean isInSystem() {
    return lane != null;
  }

  public void setLane(Lane lane) {
    if (lane == null) {
      this.lane = null;
      return;
    }
    if (!isInSystem()) {
      this.position = lane.getCenterStartPoint();
    }
    this.lane = lane;
    this.lane.enter(this);
  }

  public double getCurrentSpeed() {
    return currentSpeed;
  }

  private void setCurrentSpeed(double speed) {
    if (speed > getTopSpeed()) {
      currentSpeed = getTopSpeed();
    } else if (speed < 0) {
      currentSpeed = 0;
    } else {
      currentSpeed = speed;
    }
  }

  private double getDistanceFromEOLane() {

    double distance = getLane().getLeftEndPoint().distance(this.getPosition());
    return distance;
  }

  private void changeSpeed() {
    accelerate = driver.AccelerationStatus(this.currentSpeed, driver.getOptimalF
ollowingDistance(), getLane().getDistanceFromNextVehicle(this), getDistanceFromE
OLane());
    decelerate = driver.DecelerationStatus(this.currentSpeed, driver.getOptimalF
ollowingDistance(), getLane().getDistanceFromNextVehicle(this), getDistanceFromE
OLane());

    if (accelerate) {
      accelerate();
```

```
    } else if (decelerate) {
      decelerate();
    } else {
      currentSpeed = currentSpeed;
    }
  }

  private boolean leftRoad(Point oldPosition, Point newPosition) {
    Point endPoint = lane.getCenterEndPoint();
    if (oldPosition.getX() <= endPoint.getX() && newPosition.getX() > endPoint.g
etX()) {
      return true;
    }
    if (oldPosition.getX() >= endPoint.getX() && newPosition.getX() < endPoint.g
etX()) {
      return true;
    }
    if (oldPosition.getY() <= endPoint.getY() && newPosition.getY() > endPoint.g
etY()) {
      return true;
    }
    if (oldPosition.getY() >= endPoint.getY() && newPosition.getY() < endPoint.g
etY()) {
      return true;
    }
    return false;
  }

  private Lane chooseRandomNewLane() {
    Junction junction = lane.getJunction();
    if (junction == null) {
      return null;
    }
    List<Lane> lanes = junction.getConnectedLanes(lane);
    if (lanes.isEmpty()) {
      return null;
    }
    Random randomGenerator = new Random();
    int index = randomGenerator.nextInt(lanes.size());
    return lanes.get(index);
  }

  public Point getDisplacementVector() {
    Point dir = getLane().getDirectionVector();
    Point unitDir = dir.div(dir.distanceFromOrigin());
    double x = getCurrentSpeed() * Math.cos(unitDir.angleVector());
    double y = getCurrentSpeed() * Math.sin(unitDir.angleVector());
    return new Point(x, y);
  }

  public double timeSpentInSystem() {
    return (endTime - startTime) / 1000;
  }

  public void step(long stepCounter) {
    System.out.print(getType() + "#" + hashCode());

    // Change speed of vehicle
    changeSpeed();
```

```
    // Calculate new position
    Point newPosition = position.plus(getDisplacementVector());

    // Check if vehicle has to change lane
    if (leftRoad(this.position, newPosition)) {
      // Move vehicle to random next lane
      Lane newLane = chooseRandomNewLane();
      if (newLane != null) {
        this.lane.exit(this);
        this.position = newLane.getCenterStartPoint();
        this.setLane(newLane);
      } else {
        this.lane.exit(this);
        this.lane.getExitPoint().addVehicle(this);
        this.setLane(null);
      }
    } else {
      //Move vehicle
      position = newPosition;
    }

    System.out.println(" position: " + Math.round(position.getX()) + ", " + Math.roun
d(position.getY()) + " speed: " + Math.round(currentSpeed));
  }

  protected void accelerate() {
    double dist = getLane().getDistanceFromNextVehicle(this) - driver.getOptimal
FollowingDistance();

    double optimalSpeed = driver.getOptimalSpeedForDistance(dist);

    if (optimalSpeed > getCurrentSpeed()) {
      double speedDifference = optimalSpeed - getCurrentSpeed();
      if (speedDifference < getMaxAcceleration()) {
        setCurrentSpeed(getCurrentSpeed() + speedDifference);
      } else {
        setCurrentSpeed(getCurrentSpeed() + getMaxAcceleration());
      }
    }
  }

  protected void decelerate() {

    double dist = getLane().getDistanceFromNextVehicle(this) - driver.getOptimal
FollowingDistance();

    double optimalSpeed = driver.getOptimalSpeedForDistance(dist);

    if (optimalSpeed < getCurrentSpeed()) {
      double speedDifference = getCurrentSpeed() - optimalSpeed;
      if (speedDifference < getMaxDeceleration()) {
        setCurrentSpeed(getCurrentSpeed() - speedDifference);
      } else {
        setCurrentSpeed(getCurrentSpeed() - getMaxDeceleration());
      }
    }
  }
```

```
  private Driver NormalDriver(String default_Driver) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body o
f generated methods, choose Tools | Templates.
  }


}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.drivers;

import trafficsimulator.core.Driver;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author Eddy
 */
public class CautiousDriver extends Driver {

  public CautiousDriver(String name) {
    super(name);
  }

  @Override
  public double getOptimalDeceleration() {
    if (Car.class.isInstance(vehicle)) {
      return 3;
    } else if (Bus.class.isInstance(vehicle)) {
      return 2;
    } else {
      return 1;
    }
  }

}
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.drivers;

import trafficsimulator.core.Driver;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author Eddy
 */
public class NormalDriver extends Driver {

  public NormalDriver(String name) {

    super(name);
  }

  @Override
  public double getOptimalDeceleration() {
    if (Car.class.isInstance(vehicle)) {
      return 3;
    } else if (Bus.class.isInstance(vehicle)) {
      return 2;
    } else {
      return 1;
    }
  }
}
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.drivers;

import trafficsimulator.core.Driver;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author Eddy
 */
public class RecklessDriver extends Driver {

  public RecklessDriver(String name) {

    super(name);
  }

  @Override
  public double getOptimalDeceleration() {
    if (Car.class.isInstance(vehicle)) {
      return 3;
    } else if (Bus.class.isInstance(vehicle)) {
      return 2;
    } else {
      return 1;
    }
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.gui;

/**
 *
 * @author balazs
 */
public interface IRenderer {

  public void render();
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.gui;

import java.util.List;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
import javafx.animation.FillTransition;
import javafx.animation.ParallelTransition;
import javafx.animation.RotateTransition;
import javafx.animation.ScaleTransition;
import javafx.animation.Timeline;
import javafx.animation.TranslateTransition;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.*;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Button;
import javafx.scene.image.Image;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.transform.Rotate;
import javafx.stage.Stage;
import javafx.util.Duration;
import trafficsimulator.core.Lane;

import trafficsimulator.core.Lane.Direction;
import trafficsimulator.core.Road;
import trafficsimulator.core.Simulation;
import trafficsimulator.core.Vehicle;
import trafficsimulator.utils.Point;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author yukolthep
 */
public class SimulationRenderer implements IRenderer {

  private Stage stage;
  private Simulation simulation;
  private GraphicsContext gc;

  Image car_image = new Image("pic/car_tran.gif", 20, 0, true, false);
  Image car = new Image("pic/car.jpg");
  Image bus = new Image("pic/bus.jpg");

  public SimulationRenderer(GraphicsContext gc, Simulation simulation) {
    this.stage = stage;
```

```
    this.simulation = simulation;
    this.gc = gc;
  }

  public void render() {
    Platform.runLater(new Runnable() {

      @Override
      public void run() {
        clear();
        drawRoads();
        drawLanes();
        drawVehicles();
      }
    });

  }

  /*Clear canvas before painting updated components*/
  private void clear() {
    gc.clearRect(0, 0, 700, 700);
  }

  private void drawRoads() {
    List<Road> roads = this.simulation.getMap().getRoads();
    for (Road road : roads) {
      Point leftStartPoint = road.getLeftStartPoint();
      Point rightStartPoint = road.getRightStartPoint();
      Point leftEndPoint = road.getLeftEndPoint();
      Point rightEndPoint = road.getRightEndPoint();
      gc.setFill(Color.GRAY);
      gc.fillPolygon(new double[] {leftStartPoint.getX(),leftEndPoint.getX(),rig
htEndPoint.getX(),rightStartPoint.getX()}, new double[] {leftStartPoint.getY(),
leftEndPoint.getY(), rightEndPoint.getY(), rightStartPoint.getY()}, 4);
    }
  }

  private void drawLanes(){
    List<Road> roads = this.simulation.getMap().getRoads();
    for (Road road : roads) {
      List<Lane> lanes = road.getLanes();
      for (int index = 0 ; index < lanes.size()-1 ; index++) {
        Lane lane = lanes.get(index);
        Point leftStartPoint = lane.getLeftStartPoint();
        Point leftEndPoint = lane.getLeftEndPoint();
        Point rightStartPoint = lane.getRightStartPoint();
        Point rightEndPoint = lane.getRightEndPoint();
        gc.setLineWidth(1);
        gc.setStroke(Color.BLACK);
        if(index == lanes.size()-1){
          break;
        }
        if(lane.getDirection() == Direction.IDENTICAL){
          gc.strokeLine(rightStartPoint.getX(), rightStartPoint.getY(), rightEnd
Point.getX(), rightEndPoint.getY());
        }else{
          gc.strokeLine(leftStartPoint.getX(), leftStartPoint.getY(), leftEndPoi
nt.getX(), leftEndPoint.getY());
        }
```

```
      }
    }
  }

  private void drawVehicles() {
    List<Vehicle> vehicles = this.simulation.getVehicles();
    for (Vehicle vehicle : vehicles) {
      if (Car.class.isInstance(vehicle)) {
        Double angle = vehicle.getDisplacementVector().angleVectorDegree();
        drawRotatedImage(gc, car, angle, (vehicle.getPosition().getX() − car.get
Width() / 2), (vehicle.getPosition().getY() − car.getHeight() / 2));
      }else if (Bus.class.isInstance(vehicle)) {
        Double angle = vehicle.getDisplacementVector().angleVectorDegree();
        drawRotatedImage(gc, bus, angle, (vehicle.getPosition().getX() − bus.get
Width() / 2), (vehicle.getPosition().getY() − bus.getHeight() / 2));
      }
    }
  }


  private void rotate(GraphicsContext gc, double angle, double px, double py) {
    Rotate r = new Rotate(angle, px, py);
    gc.setTransform(r.getMxx(), r.getMyx(), r.getMxy(), r.getMyy(), r.getTx(), r
.getTy());
  }

  private void drawRotatedImage(GraphicsContext gc, Image image, double angle, d
ouble tlpx, double tlpy) {
    gc.save(); // saves the current state on stack, including the current transf
orm
    rotate(gc, angle, tlpx + image.getWidth() / 2, tlpy + image.getHeight() / 2)
;
    gc.drawImage(image, tlpx, tlpy);
    gc.restore(); // back to original state (before rotation)
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.gui;


import javafx.geometry.Rectangle2D;
import javafx.scene.Scene;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.stage.Modality;
import javafx.stage.Screen;
import javafx.stage.Stage;
import trafficsimulator.core.Simulation;

/**
 *
 * @author yukolthep
 */
public class SimulationResults extends Stage{
  public SimulationResults(Stage primaryStage, Simulation simulation){
    initModality(Modality.NONE);
    initOwner(primaryStage);
    VBox dialogVbox = new VBox(20);
    dialogVbox.getChildren().add(simulation.averageTime());
    dialogVbox.getChildren().add(simulation.longestTime());
    dialogVbox.getChildren().add(simulation.shortestTime());
    Scene dialogScene = new Scene(dialogVbox, 300, 200);
    setScene(dialogScene);
    Rectangle2D primScreenBounds = Screen.getPrimary().getVisualBounds();
    setX((primScreenBounds.getWidth() - getWidth()) / 2);
    setY((primScreenBounds.getHeight() - getHeight()) / 4);
    show();
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.junctions;

import trafficsimulator.core.Lane;

/**
 *
 * @author balazs
 */
public class TrafficLight {
  public static final int GREEN_DURATION = 10;
  public static final int YELLOW_DURATION = 2;
  public static final int RED_DURATION = 10;
  public static final int REDYELLOW_DURATION = 3;

  public enum State {

    GREEN, YELLOW, RED, REDYELLOW
  }

  private State state = State.RED;
  private Lane lane;

  public TrafficLight(Lane lane){
    this.lane = lane;
  }

  public State getState() {
    return state;
  }

  public void setState(State state) {
    this.state = state;
  }

  public Lane getLane() {
    return lane;
  }

  public void nextState(){
    switch(state){
      case GREEN:
        setState(State.YELLOW);
        break;
      case YELLOW:
        setState(State.RED);
        break;
      case RED:
        setState(State.REDYELLOW);
        break;
      case REDYELLOW:
        setState(State.GREEN);
        break;
    }
  }
```

```
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.junctions;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import trafficsimulator.core.Junction;
import trafficsimulator.core.Lane;
import trafficsimulator.core.Vehicle;

/**
 *
 * @author balazs
 */
public class TrafficLightJunction extends Junction{

  private List<TrafficLight> trafficLights = new ArrayList();
  private TrafficLight activeTrafficLight;
  private int stepCounter = 0;

  private TrafficLight getTrafficLightForLane(Lane lane){
    for(TrafficLight trafficLight : trafficLights){
      if(trafficLight.getLane()==lane){
        return trafficLight;
      }
    }
    return null;
  }

  public void connect(Lane source, Lane destination) {
    super.connect(source, destination);

    if (getTrafficLightForLane(source) == null) {
      TrafficLight trafficLight = new TrafficLight(source);
      trafficLights.add(trafficLight);
    }
  }

  public boolean shouldVehicleEnterJunction(Vehicle vehicle){
    TrafficLight trafficLight = getTrafficLightForLane(vehicle.getLane());
    if(trafficLight.getState() == TrafficLight.State.GREEN){
      return true;
    }else{
      return false;
    }
  }

  private void activateTrafficLight(TrafficLight activeTrafficLight){
    // Making sure all traffic lights are red
    for(TrafficLight trafficLight : trafficLights){
      trafficLight.setState(TrafficLight.State.RED);
    }

    // Activating light
```

```
    this.activeTrafficLight = activeTrafficLight;
    activeTrafficLight.nextState();
    stepCounter = 0;
  }

  private void activateNextTrafficLight(){
    int index = trafficLights.indexOf(activeTrafficLight);
    if(index == trafficLights.size()-1){
      activateTrafficLight(trafficLights.get(0));
    }else{
      activateTrafficLight(trafficLights.get(index+1));
    }
  }

  public void step(long step) {
    if(activeTrafficLight == null){
      activateTrafficLight(trafficLights.get(0));
      return;
    }

    stepCounter++;

    if(activeTrafficLight.getState() == TrafficLight.State.GREEN && stepCounter
== TrafficLight.GREEN_DURATION){
      activeTrafficLight.nextState();
      stepCounter = 0;
    }else if(activeTrafficLight.getState() == TrafficLight.State.YELLOW && stepC
ounter == TrafficLight.YELLOW_DURATION){
      activateNextTrafficLight();
    }else if(activeTrafficLight.getState() == TrafficLight.State.REDYELLOW && st
epCounter == TrafficLight.REDYELLOW_DURATION){
      activeTrafficLight.nextState();
      stepCounter = 0;
    }else if(activeTrafficLight.getState() == TrafficLight.State.RED && stepCoun
ter == TrafficLight.RED_DURATION){
      activeTrafficLight.nextState();
      stepCounter = 0;
    }
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.simulations;

import trafficsimulator.core.Driver;
import trafficsimulator.core.Junction;
import trafficsimulator.core.Lane;
import trafficsimulator.core.Road;
import trafficsimulator.core.Simulation;
import trafficsimulator.core.Vehicle;
import trafficsimulator.drivers.CautiousDriver;
import trafficsimulator.drivers.NormalDriver;
import trafficsimulator.drivers.RecklessDriver;
import trafficsimulator.junctions.TrafficLightJunction;
import trafficsimulator.utils.Point;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;


/**
 *
 * @author balazs
 */
public class Simulation1 extends Simulation{

  @Override
  protected void init() {
    Road r1 = new Road(new Point(70, 20), new Point(500, 20));
    Lane l11 = new Lane(Lane.Direction.IDENTICAL);
    Lane l12 = new Lane(Lane.Direction.OPPOSITE);
    r1.addLane(l11);
    r1.addLane(l12);
    Road r2 = new Road(new Point(500, 20), new Point(500, 450));
    Lane l21 = new Lane(Lane.Direction.IDENTICAL);
    Lane l22 = new Lane(Lane.Direction.OPPOSITE);
    r2.addLane(l21);
    r2.addLane(l22);
    Road r3 = new Road(new Point(500, 450), new Point(20, 100));
    Lane l31 = new Lane(Lane.Direction.IDENTICAL);
    Lane l32 = new Lane(Lane.Direction.OPPOSITE);
    r3.addLane(l31);
    r3.addLane(l32);
    Road r4 = new Road(new Point(500, 20), new Point(600, 20));
    Lane l41 = new Lane(Lane.Direction.IDENTICAL);
    Lane l42 = new Lane(Lane.Direction.OPPOSITE);
    r4.addLane(l41);
    r4.addLane(l42);
    Road r5 = new Road(new Point(600, 20), new Point(600, 450));
    Lane l51 = new Lane(Lane.Direction.IDENTICAL);
    Lane l52 = new Lane(Lane.Direction.OPPOSITE);
    r5.addLane(l51);
    r5.addLane(l52);
    Road r6 = new Road(new Point(600, 450), new Point(500, 450));
    Lane l61 = new Lane(Lane.Direction.IDENTICAL);
    Lane l62 = new Lane(Lane.Direction.OPPOSITE);
    r6.addLane(l61);
```

```
    r6.addLane(l62);
    Road r7 = new Road(new Point(600, 450), new Point(650, 450));
    Lane l71 = new Lane(Lane.Direction.IDENTICAL);
    r7.addLane(l71);

    Junction j1 = new TrafficLightJunction();
    j1.connect(l11, l21);
    j1.connect(l11, l41);
    j1.connect(l22, l12);
    j1.connect(l22, l41);
    j1.connect(l42, l12);
    j1.connect(l42, l21);
    Junction j2 = new TrafficLightJunction();
    j2.connect(l21, l31);
    j2.connect(l21, l62);
    j2.connect(l32, l22);
    j2.connect(l32, l62);
    j2.connect(l61, l22);
    j2.connect(l61, l31);
    Junction j3 = new TrafficLightJunction();
    j3.connect(l31, l11);
    j3.connect(l12, l32);
    Junction j4 = new TrafficLightJunction();
    j4.connect(l41, l51);
    j4.connect(l52, l42);
    Junction j5 = new TrafficLightJunction();
    j5.connect(l51, l61);
    j5.connect(l51, l71);
    j5.connect(l62, l52);
    j5.connect(l62, l71);

    map.addRoad(r1);
    map.addRoad(r2);
    map.addRoad(r3);
    map.addRoad(r4);
    map.addRoad(r5);
    map.addRoad(r6);
    map.addRoad(r7);
    map.addJunction(j1);
    map.addJunction(j2);
    map.addJunction(j3);
    map.addJunction(j4);
    map.addJunction(j5);

    Driver tom = new CautiousDriver("Tom");
    Driver mary = new NormalDriver("Mary");
    Driver jerry = new RecklessDriver("Jerry");


    addVehicle(new Car(tom), l11, 1);
    addVehicle(new Bus(jerry), l11, 20);

  }

}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.simulations;

import trafficsimulator.core.Driver;
import trafficsimulator.core.Junction;
import trafficsimulator.core.Lane;
import trafficsimulator.core.Road;
import trafficsimulator.core.Simulation;
import trafficsimulator.core.Vehicle;
import trafficsimulator.drivers.CautiousDriver;
import trafficsimulator.drivers.NormalDriver;
import trafficsimulator.drivers.RecklessDriver;
import trafficsimulator.junctions.TrafficLightJunction;
import trafficsimulator.utils.Point;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author yukolthep
 */
public class Simulation2 extends Simulation {


  @Override
  protected void init() {
    Road r1 = new Road(new Point(70, 300), new Point(270, 100));
    Lane l11 = new Lane(Lane.Direction.IDENTICAL);
    Lane l12 = new Lane(Lane.Direction.OPPOSITE);
    r1.addLane(l11);
    r1.addLane(l12);
    Road r2 = new Road(new Point(270, 100), new Point(470, 300));
    Lane l21 = new Lane(Lane.Direction.IDENTICAL);
    Lane l22 = new Lane(Lane.Direction.OPPOSITE);
    r2.addLane(l21);
    r2.addLane(l22);
    Road r3 = new Road(new Point(470, 300), new Point(270, 500));
    Lane l31 = new Lane(Lane.Direction.IDENTICAL);
    Lane l32 = new Lane(Lane.Direction.OPPOSITE);
    r3.addLane(l31);
    r3.addLane(l32);
    Road r4 = new Road(new Point(270, 500), new Point(70, 300));
    Lane l41 = new Lane(Lane.Direction.IDENTICAL);
    Lane l42 = new Lane(Lane.Direction.OPPOSITE);
    r4.addLane(l41);
    r4.addLane(l42);

    Junction j1 = new TrafficLightJunction();
    j1.connect(l11, l21);
    j1.connect(l22, l12);
    Junction j2 = new TrafficLightJunction();
    j2.connect(l21, l31);
    j2.connect(l32, l22);
    Junction j3 = new TrafficLightJunction();
    j3.connect(l31, l41);
    j3.connect(l42, l32);
```

```
    Junction j4 = new TrafficLightJunction();
    j4.connect(l41, l11);
    j4.connect(l12, l42);

    map.addRoad(r1);
    map.addRoad(r2);
    map.addRoad(r3);
    map.addRoad(r4);
    map.addJunction(j1);
    map.addJunction(j2);
    map.addJunction(j3);
    map.addJunction(j4);

    Driver tom = new CautiousDriver("Tom");
    Driver jerry = new RecklessDriver("Jerry");

    Vehicle olo = new Car(tom);
    Vehicle olo_v2 = new Car(jerry);

    addVehicle(olo, l11, 1);
    addVehicle(olo_v2, l22, 1);

  }

}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.utils;

/**
 *
 * @author balazs
 */
public class Point {

  public double x, y;

  public Point() {
    x = 0;
    y = 0;
  }

  public Point(double x, double y) {
    this.x = x;
    this.y = y;
  }

  public double getX() {
    return x;
  }

  public void setX(double x) {
    this.x = x;
  }

  public double getY() {
    return y;
  }

  public void setY(double y) {
    this.y = y;
  }

  public Point plus(Point p) {
    return new Point(this.x + p.x, this.y + p.y);
  }

  public Point minus(Point p) {
    return new Point(this.x - p.x, this.y - p.y);
  }

  public Point mult(double k) {
    return new Point(this.x * k, this.y * k);
  }

  public Point div(double k) {
    return new Point(this.x / k, this.y / k);
  }

  public double distanceFromOrigin() {
    Point origin = new Point();
```

```
    return distance(origin);
  }

  public double distance(Point p) {
    double dx = x - p.x;
    double dy = y - p.y;
    double distance = Math.sqrt(dx * dx + dy * dy);
    return distance;
  }

  public boolean inSameQuadrant(Point p) {
    if (getX() > 0 && p.getX() < 0) {
      return false;
    }
    if (getX() < 0 && p.getX() > 0) {
      return false;
    }
    if (getY() > 0 && p.getY() < 0) {
      return false;
    }
    if (getY() < 0 && p.getY() > 0) {
      return false;
    }
    return true;
  }

  public Point rotateVector(double degrees) {
    double X = Math.round(this.x * Math.cos(degrees) - this.y * Math.sin(degrees
));
    double Y = Math.round(this.x * Math.sin(degrees) + this.y * Math.cos(degrees
));
    return new Point(X, Y);
  }

  public double angleVector() {
    if (y == 0) {
      if (x < 0) {
        return Math.PI;
      } else {
        return 0;
      }
    } else if (x < 0) {
      if (y > 0) {
        return Math.atan(this.y / this.x) + Math.PI;
      } else {
        return Math.atan(this.y / this.x) - Math.PI;
      }
    } else {
      return Math.atan(this.y / this.x);
    }
  }

    public double angleVectorDegree() {
    if (y == 0) {
      if (x < 0) {
        return Math.PI*(180/Math.PI);
      } else {
        return 0;
      }
```

```
    } else if (x < 0) {
      if (y > 0) {
        return (Math.atan(this.y / this.x) + Math.PI)*(180/Math.PI);
      } else {
        return (Math.atan(this.y / this.x) − Math.PI)*(180/Math.PI);
      }
    } else {
      return Math.atan(this.y / this.x)*(180/Math.PI);
    }
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.utils;

/**
 *
 * @author balazs
 */
public class Size {

  public double width;
  public double height;

  public Size(double width, double height) {
    this.width = width;
    this.height = height;
  }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.vehicles;

import trafficsimulator.core.Driver;
import trafficsimulator.core.Vehicle;
import trafficsimulator.utils.Size;

/**
 *
 * @author snorri
 */
public class Bus extends Vehicle {

  public Bus() {
    this(null);
  }

  public Bus(Driver driver) {
    super(driver);
    topSpeed = 6;
    maxAcceleration = 1;
    maxDeceleration = 3;
    optimalDeceleration = 2;
    size = new Size(20, 10);
  }

  @Override
  public String getType() {
    return "Bus";
  }

}
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package trafficsimulator.vehicles;

import trafficsimulator.core.Vehicle;
import trafficsimulator.utils.Size;
import trafficsimulator.core.Driver;

/**
 *
 * @author snorri
 */
public class Car extends Vehicle {

  public Car() {
    this(null);
  }

  public Car(Driver driver) {
    super(driver);
    topSpeed = 10;
    maxAcceleration = 2;
    maxDeceleration = 4;
    optimalDeceleration = 3;
    size = new Size(14, 8);
  }

  @Override
  public String getType() {
    return "Car";
  }

}
```