```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 trafficsimulator.junctions.TrafficLight;
import trafficsimulator.junctions.TrafficLightJunction;

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

  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();
  abstract public double getOptimalAcceleration();

  private double getOptimalSpeedForDistance(double distance) {
    distance = Math.max(distance, 0.0);
    double time = Math.sqrt((2*distance)/getOptimalDeceleration());
    return getOptimalDeceleration() * time;
  }

  public double getOptimalFollowingDistance() {
    double speed2 = vehicle.getCurrentSpeed()*vehicle.getCurrentSpeed();
    double stoppingDistance = speed2 / (getOptimalDeceleration()*2);
    return 10 + stoppingDistance;
  }

  private void changeSpeed() {
    double speedDelta = getOptimalAcceleration();

    // Change speed based on following distance
    double nextVehicleDist = vehicle.getLane().getDistanceFromVehicleInFront(veh
icle);
    if (nextVehicleDist <= getOptimalFollowingDistance()) {
      double dist = nextVehicleDist - getOptimalFollowingDistance();
      double optimalSpeed = getOptimalSpeedForDistance(dist);
      double newSpeedDelta = optimalSpeed - vehicle.getCurrentSpeed();
      speedDelta = Math.min(speedDelta, newSpeedDelta);
    }

    //Change speed based on traffic lights
    Junction junction = vehicle.getLane().getJunction();
    if(junction instanceof TrafficLightJunction){
      TrafficLightJunction trafficLightJunction = (TrafficLightJunction)junction
;
      TrafficLight light = trafficLightJunction.getTrafficLightForLane(vehicle.g
etLane());
      if(light != null){
        boolean greenLight = light.getState() == TrafficLight.State.GREEN;
        double dist = Math.max(vehicle.getDistanceFromEndOfLane() − 30, 0);

        //Vehicle is waiting for green light
        if(dist == 0 && !greenLight && vehicle.getCurrentSpeed()==0){
          speedDelta = Math.min(speedDelta, 0);
        }else{
          double opSpeed = getOptimalSpeedForDistance(dist);
          boolean shouldSlowDown = opSpeed < vehicle.getCurrentSpeed();
          boolean canStop = (vehicle.getCurrentSpeed() − vehicle.getMaxDecelerat
ion()*dist) < 0;

          if(!greenLight && shouldSlowDown){
            double time = Math.sqrt((2*dist)/getOptimalDeceleration());
            double newSpeedDelta = − (vehicle.getCurrentSpeed() / time);
            speedDelta = Math.min(speedDelta, newSpeedDelta);
          }
        }

      }
    }

    vehicle.changeSpeed(speedDelta);

  }

  @Override
  public void step(long step) {
    // Change speed of vehicle
    changeSpeed();
  }

}
```

```
/*
 * 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);

  }

  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) {
      if(lane.getFreeSpace() > vehicle.getSize().height){
        //Add vehicle to system
        System.out.println(vehicle + " entered the system");
        vehicle.startTime = System.currentTimeMillis();
        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();
  }

  public List<Vehicle> getExitedVehicles() {
    return vehicles;
  }

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

```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;

/**
 *
 * @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.HashSet;
import java.util.List;
import java.util.Set;
import trafficsimulator.utils.Point;

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

  private final HashMap<Lane, List<Lane>> connections = new HashMap<>();
  private final List<Lane> lanes = new ArrayList<>();
  private final HashMap<Road, List<Point>> roadPoints = new HashMap<>();

  public Junction() {
  }

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

    //Store points
    if(!roadPoints.containsKey(source.getRoad())){
      if(source.getDirection() == Lane.Direction.IDENTICAL){
        List<Point> points = new ArrayList<>();

        points.add(source.getRoad().getLeftEndPoint());
        points.add(source.getRoad().getRightEndPoint());
        roadPoints.put(source.getRoad(), points);
      }else if(source.getDirection() == Lane.Direction.OPPOSITE){
        List<Point> points = new ArrayList<>();

        points.add(source.getRoad().getRightStartPoint());
        points.add(source.getRoad().getLeftStartPoint());
        roadPoints.put(source.getRoad(), points);
      }
    }
    if(!roadPoints.containsKey(destination.getRoad())){
      if(destination.getDirection() == Lane.Direction.OPPOSITE){
        List<Point> points = new ArrayList<>();

        points.add(destination.getRoad().getLeftEndPoint());
        points.add(destination.getRoad().getRightEndPoint());
        roadPoints.put(destination.getRoad(), points);
      }else if(destination.getDirection() == Lane.Direction.IDENTICAL){
        List<Point> points = new ArrayList<>();

        points.add(destination.getRoad().getRightStartPoint());
        points.add(destination.getRoad().getLeftStartPoint());
        roadPoints.put(destination.getRoad(), points);
      }
    }
  }

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

  public List<Road> getRoads(){
    return new ArrayList<>(roadPoints.keySet());
  }

  public List<Point> getPointsForRoad(Road road){
    return roadPoints.get(road);
  }

  public Point getCenterPoint(){
    Set<Point> allPoints = new HashSet<>();
    for(Road road: getRoads()){
      for(Point point:getPointsForRoad(road)){
        allPoints.add(point);
      }
    }
    return Point.centroid(new ArrayList(allPoints));
  }

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

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

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

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

  public static final double LANE_WIDTH = 25;

  public enum Direction {

    IDENTICAL, OPPOSITE
  }

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

  public Lane(Direction direction, Point startPoint, Point endPoint, Point laneP
ointStart, Point lanePointStop) {
    this.direction = direction;
    this.startPoint = startPoint;
    this.endPoint = endPoint;
    this.lanePointStart = lanePointStart;
    this.lanePointStop = lanePointStop;
    exitPoint = new ExitPoint(this);
  }

  public Lane(Direction direction, Point startPoint, Point endPoint){
    this.direction = direction;
    this.startPoint = startPoint;
    this.endPoint = endPoint;
    exitPoint = new ExitPoint(this);
  }

  public Point getStartPoint() {
    return startPoint;
  }

  public Point getEndPoint() {
    return endPoint;
  }
  public double getLength(){
    return startPoint.distance(endPoint);
  }

  public Point getLaneStart(){
    return lanePointStart;
  }

  public Point getLaneStop(){
    return lanePointStop;
  }

  public void setStartPoint(Point startPoint) {
    this.startPoint = startPoint;
  }

  public void setEndPoint(Point endPoint) {
    this.endPoint = endPoint;
  }

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

    return lanes.get(index);
  }

  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 getDirectionVector() {
    return getEndPoint().minus(getStartPoint());
  }

  private Vehicle getVehicleInFront(Vehicle vehicle){
    double minDistance = Double.MAX_VALUE;
    Vehicle vehicleInFront = null;

    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;
          vehicleInFront = v;
        }
      }
    }

    return vehicleInFront;
  }

  public double getDistanceFromVehicleInFront(Vehicle vehicle) {
    Vehicle vehicleInFront = getVehicleInFront(vehicle);
    if(vehicleInFront == null)
      if (getNextLane() != null) return getNextLane().getFreeSpace();
      else  return Double.MAX_VALUE;
    double distance = vehicle.getPosition().distance(vehicleInFront.getPosition(
));
    distance -= vehicle.getSize().width;
    return distance;
  }


  public Vehicle getLastVehicle(){
    Vehicle vehicle = null;
    double minDistance = Double.MAX_VALUE;
    for(Vehicle v:vehicles){
      double distance = v.getPosition().distance(startPoint);
      if(distance < minDistance){
        minDistance = distance;
        vehicle = v;
      }
    }
    return vehicle;
  }

  public double getFreeSpace(){
    Vehicle lastVehicle = getLastVehicle();
    if(lastVehicle != null){
      return lastVehicle.getPosition().distance(startPoint) - lastVehicle.getSiz
e().height;
    }else{
      return getLength();
    }
  }


  public List<Vehicle> getVehicles(){
  return this.vehicles;
  }

  public double getLaneLength(){
   return Point.distanceBetweenPoints(startPoint, endPoint);
}

}
```

```
/*
 * 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);
  }

}
```

```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.List;
import trafficsimulator.utils.Point;

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

  private List<Lane> lanes;

  private Point leftStartPoint;
  private Point leftEndPoint;

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

  public Lane addLane(Lane.Direction direction) {
    double offsetX = (lanes.size() * Lane.LANE_WIDTH + Lane.LANE_WIDTH/2) * Math
.cos(acrossRoadUnitVector().angleVector());
    double offsetY = (lanes.size() * Lane.LANE_WIDTH + Lane.LANE_WIDTH/2) * Math
.sin(acrossRoadUnitVector().angleVector());
    double offsetX_2 = (lanes.size() * Lane.LANE_WIDTH + Lane.LANE_WIDTH) * Math
.cos(acrossRoadUnitVector().angleVector());
    double offsetY_2 = (lanes.size() * Lane.LANE_WIDTH + Lane.LANE_WIDTH) * Math
.sin(acrossRoadUnitVector().angleVector());
    Point startPoint;
    Point endPoint;
    Point lanePointStart;
    Point lanePointStop;
    if(direction == Lane.Direction.IDENTICAL){
      startPoint = leftStartPoint.plus(new Point(offsetX, offsetY));
      endPoint = leftEndPoint.plus(new Point(offsetX, offsetY));
      lanePointStart = leftStartPoint.plus(new Point(offsetX_2, offsetY_2));
      lanePointStop = leftEndPoint.plus(new Point(offsetX_2, offsetY_2));
    }else{
      startPoint = leftEndPoint.plus(new Point(offsetX, offsetY));
      endPoint = leftStartPoint.plus(new Point(offsetX, offsetY));
      lanePointStart = leftEndPoint.plus(new Point(offsetX_2, offsetY_2));
      lanePointStop = leftStartPoint.plus(new Point(offsetX_2, offsetY_2));
    }
    Lane lane = new Lane(direction, startPoint, endPoint, lanePointStart, lanePo
intStop);
    lanes.add(lane);
    lane.setRoad(this);
    return lane;
  }
  public List<Lane> getLanes() {
    return 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() {
    return lanes.size()*Lane.LANE_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.math.BigDecimal;
import java.math.RoundingMode;
import java.util.ArrayList;
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;
  private long duration;
  public int counter = 0;
  protected int longestSimulationTime;
  protected boolean peaktime;
  protected boolean congestionControl;

  public Simulation(boolean peaktime, boolean congestionControl, int longestSimu
lationTime) {
    this.peaktime = peaktime;
    this.congestionControl = congestionControl;
    this.longestSimulationTime = longestSimulationTime;
  }

  protected abstract void init();

  @Override
  public void run() {

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

    if (!isRunning()) {
      printStats();
      System.out.println("Simulation end");
      timer.cancel();
      return;
    }

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

```java
    }

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

    for (Vehicle vehicle : getVehicles()) {
      vehicle.getDriver().step(stepCounter);
    }

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

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

  public boolean isRunning(){
    if (numberOfVehiclesAtExitPoints() == vehicles.size()) {
      return false;
    }

    if (stepCounter/10 >= duration) {
      return false;
    };

    return true;
  }

  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;
  }

  public 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 void setDuration(long duration){
    this.duration = duration;
  }

  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> exitedVehicles = new ArrayList<>();
    for (ExitPoint ep : exitPoints) {
      exitedVehicles.addAll(ep.getExitedVehicles());
    }
    return exitedVehicles;
  }

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

  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(" 0 second");
    else return new Text(" " + String.valueOf(average) + " seconds");
  }

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

  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(" 0 second");
    else return new Text(" " + String.valueOf(shortest) + " seconds");
  }

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

}
```

```
/*
 * 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.drivers.NormalDriver;
import trafficsimulator.utils.Point;
import trafficsimulator.utils.Size;

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

  private Lane lane;
  private Lane nextLane;
  private Point position;
  private double currentSpeed = 0;
  private double acceleration = 0;
  protected double topSpeed;
  protected double maxAcceleration;
  protected double maxDeceleration;
  protected Size size;
  protected Driver driver;

  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 = new NormalDriver("Default Driver");
    } else {
      this.driver = driver;
    }
    this.driver.setVehicle(this);
  }

  public Driver getDriver() {
    return driver;
  }

  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.getStartPoint();
    }
    this.lane = lane;
    this.lane.enter(this);
    setNextLane(this.lane.getNextLane());
  }

  public double getCurrentSpeed() {
    return currentSpeed;
  }

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

  public double getDistanceFromEndOfLane() {
    double distance = getLane().getEndPoint().distance(this.getPosition());
    return distance;
  }

  private boolean leftRoad(Point oldPosition, Point newPosition) {
    Point endPoint = lane.getEndPoint();
    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 void setNextLane(Lane nextLane) {
    this.nextLane = nextLane;
  }

  private Lane getNextLane() {
    return nextLane;
  }

  public Point getDirectionVector() {
    Point dir = getLane().getDirectionVector();
    return dir.unitVector();
  }

  private Point getDisplacementVector() {
    double angleVector = getDirectionVector().angleVector();
    double x = (getCurrentSpeed() + acceleration / 2) * Math.cos(angleVector);
    double y = (getCurrentSpeed() + acceleration / 2) * Math.sin(angleVector);
    return new Point(x, y);
  }

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

  @Override
  public void step(long stepCounter) {
    if (!isInSystem()) {
      return;
    }

    System.out.print(getType() + "#" + hashCode());

    // Calculate new position
    Point newPosition = position.plus(getDisplacementVector());
    setCurrentSpeed(getCurrentSpeed() + acceleration);

    // Check if vehicle has to change lane
    if (leftRoad(this.position, newPosition)) {
      // Move vehicle to random next lane
      Lane newLane = nextLane;
      if (newLane != null) {
        this.lane.exit(this);
        this.position = newLane.getStartPoint();
        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()) + ", " + Ma
th.round(position.getY()) + " speed: " + Math.round(currentSpeed));
  }

  public void changeSpeed(double speedDelta) {
    if (speedDelta > getMaxAcceleration()) {
      speedDelta = getMaxAcceleration();
    }
    if (speedDelta < 0 - getMaxDeceleration()) {
      speedDelta = 0 - getMaxDeceleration();
    }

    acceleration = speedDelta;
  }

}
```

CautiousDriver.java

```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;

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

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

  @Override
  public double getOptimalDeceleration() {
    return 0.5 * vehicle.getMaxDeceleration();
  }

  @Override
  public double getOptimalAcceleration() {
    return 0.5 * vehicle.getMaxAcceleration();
  }

}
```

```
/*
 * 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;

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

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

  @Override
  public double getOptimalDeceleration() {
    return 0.75 * vehicle.getMaxDeceleration();
  }

  @Override
  public double getOptimalAcceleration() {
    return 0.75 * vehicle.getMaxAcceleration();
  }
}
```

RecklessDriver.java

```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;

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

  public RecklessDriver(String name) {

    super(name);
  }

  @Override
  public double getOptimalDeceleration() {
    return vehicle.getMaxDeceleration();
  }

  @Override
  public double getOptimalAcceleration() {
    return vehicle.getMaxAcceleration();
  }
}
```

```
/*
 * 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.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.geometry.Rectangle2D;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.ColumnConstraints;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.RowConstraints;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.stage.Modality;
import javafx.stage.Screen;
import javafx.stage.Stage;

/**
 *
 * @author yukolthep
 */
public class DurationInputError extends Stage{

  private Button OK;
  private GridPane pane;
  private Text t;
  private Scene scene;
  private VBox container;
  private BorderPane subContainer;

  public DurationInputError(Stage primaryStage){
    initModality(Modality.WINDOW_MODAL);
    initOwner(primaryStage);
    OK = new Button("OK");
    OK.setPrefSize(75, 25);
    OK.setOnAction(new ButtonHandler());
    OK.setOnKeyPressed(new KeyHandler());
    t = new Text("Please input a value between 1 − 1000!");
    t.setStyle("−fx−font−size: 25px;");
    subContainer = new BorderPane();
    subContainer.setCenter(OK);
    container = new VBox();
    container.setPadding(new Insets(10,10,10,10));
    container.setSpacing(40);
    container.getChildren().addAll(t, subContainer);
    pane = new GridPane();
    RowConstraints row1 = new RowConstraints();
    RowConstraints row2 = new RowConstraints();
    RowConstraints row3 = new RowConstraints();
    row1.setPercentHeight(15);
    row2.setPercentHeight(70);
    row3.setPercentHeight(15);
    ColumnConstraints col1 = new ColumnConstraints();
    ColumnConstraints col2 = new ColumnConstraints();
    ColumnConstraints col3 = new ColumnConstraints();
    col1.setPercentWidth(15);
    col2.setPercentWidth(70);
    col3.setPercentWidth(15);
    pane.getRowConstraints().addAll(row1, row2, row3);
    pane.getColumnConstraints().addAll(col1, col2, col3);
    pane.add(container, 1, 1);
    scene = new Scene(pane,600,200);
    setScene(scene);
    Rectangle2D primScreenBounds = Screen.getPrimary().getVisualBounds();
    setX((primScreenBounds.getWidth()/2) − scene.getWidth()/2);
    setY((primScreenBounds.getHeight()/2) − scene.getHeight()/2);
    show();
  }

  class ButtonHandler implements EventHandler<ActionEvent>{
    @Override
    public void handle(ActionEvent event) {
      close();
    }
  }

  class KeyHandler implements EventHandler<KeyEvent>{
    @Override
    public void handle(KeyEvent event) {
      if(event.getCode().equals(KeyCode.ENTER)){
        OK.fire();
      }
    }
  }
}
```

```
/*
 * 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.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import trafficsimulator.core.Junction;
import trafficsimulator.core.Lane;
import trafficsimulator.core.Road;
import trafficsimulator.junctions.TrafficLight;
import trafficsimulator.junctions.TrafficLightJunction;
import trafficsimulator.utils.Point;
import trafficsimulator.utils.PointCWComparator;

/**
 *
 * @author balazs
 */
public class JunctionRenderer implements IRenderer {

  private Junction junction;
  private GraphicsContext gc;

  public JunctionRenderer(GraphicsContext gc, Junction junction) {
    this.gc = gc;
    this.junction = junction;
  }

  private class RoadCWComparator implements Comparator<Road> {

    private final Junction junction;

    public RoadCWComparator(Junction junction) {
      this.junction = junction;
    }

    @Override
    public int compare(Road o1, Road o2) {
      List<Point> cPoints1 = junction.getPointsForRoad(o1);
      List<Point> cPoints2 = junction.getPointsForRoad(o2);

      Point cPoint1 = Point.centroid(cPoints1);
      Point cPoint2 = Point.centroid(cPoints2);

      PointCWComparator comparator = new PointCWComparator(junction.getCenterPoi
nt());
      return comparator.compare(cPoint1, cPoint2);
    }

  }

  @Override
  public void render() {
    List<Road> roads = junction.getRoads();
    Collections.sort(roads, new RoadCWComparator(junction));

    List<Point> points = new ArrayList<>();
    for (Road road : roads) {
      points.add(junction.getPointsForRoad(road).get(0));
      points.add(junction.getPointsForRoad(road).get(1));
    }

    double[] xPoints = new double[points.size()];
    double[] yPoints = new double[points.size()];
    for (int i = 0; i < points.size(); i++) {
      xPoints[i] = points.get(i).getX();
      yPoints[i] = points.get(i).getY();
    }
    gc.setFill(Color.GRAY);
    gc.fillPolygon(xPoints, yPoints, points.size());

    renderTrafficLights();
  }

  private void renderTrafficLights(){
    if(junction instanceof TrafficLightJunction){
      TrafficLightJunction trafficLightJunction = (TrafficLightJunction)junction
;
      List<TrafficLight> lights = trafficLightJunction.getTrafficLights();
      for(TrafficLight light : lights){
        Point pos = light.getPosition();
        if(light.getState() == TrafficLight.State.GREEN){
          gc.setFill(Color.GREEN);
        }else if(light.getState() == TrafficLight.State.RED){
          gc.setFill(Color.RED);
        }else if(light.getState() == TrafficLight.State.REDYELLOW){
          gc.setFill(Color.YELLOW);
        }else if(light.getState() == TrafficLight.State.YELLOW){
          gc.setFill(Color.YELLOW);
        }
        gc.fillOval(pos.getX()-5, pos.getY()-5, 10, 10);
      }
    }
  }

}
```

```
/*
 * 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.Insets;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.TextField;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;

/**
 *
 * @author yukolthep
 */
public class SceneComponents extends BorderPane {

  protected StackPane canvas_panel;
  protected Canvas canvas;
  protected StackPane setting_panel;
  public GraphicsContext gc;

  protected HBox policy_box;
  protected HBox peak_box;
  protected VBox policy_radio_button_box;
  protected VBox peaktime_radio_box;

  protected ToggleGroup policies_selector;
  protected RadioButton fixed_time;
  protected RadioButton congestion_control;
  protected ToggleGroup peakTime_selector;
  protected RadioButton peak;
  protected RadioButton offPeak;

  protected HBox duration_box;
  public TextField duration_field;

  protected HBox map_box;
  public ComboBox map_list;
  protected Label selection_label;

  protected BorderPane button_pane;
  protected HBox button_box;
  public Button startSim;
  public Button showResults;

  protected VBox container;
  public ToggleGroup peakTimeSelector;
  public ToggleGroup policySelector;

  public SceneComponents() {
    this.setLeft(getCanvasPanel());
    this.setCenter(getContainerBox());
  }

  private StackPane getCanvasPanel() {
    canvas_panel = new StackPane();
    canvas_panel.setStyle("−fx−background−color: white");
    canvas = new Canvas(800, 600);
    canvas_panel.getChildren().add(canvas);
    gc = canvas.getGraphicsContext2D();
    return canvas_panel;
  }

  private HBox getPolicyBox() {
    fixed_time = new RadioButton("Fixed Time policy");
    congestion_control = new RadioButton("Congestion Control policy");
    policies_selector = new ToggleGroup();
    fixed_time.setToggleGroup(policies_selector);
    fixed_time.setUserData(false);
    congestion_control.setToggleGroup(policies_selector);
    congestion_control.setUserData(true);
    fixed_time.setSelected(true);
    policy_radio_button_box = new VBox();
    policy_radio_button_box.setSpacing(15);
    policy_radio_button_box.getChildren().addAll(fixed_time, congestion_control)
;
    policy_box = new HBox();
    policy_box.setPadding(new Insets(10, 15, 10, 15));
    policy_box.setSpacing(10);
    policy_box.getChildren().addAll(new Text("Policy: "), policy_radio_button_box)
;
    policySelector = policies_selector;
    return policy_box;
  }

  private HBox getPeakTimeBox() {
    peak = new RadioButton("Peaktime");
    offPeak = new RadioButton("Off Peak");
    peakTime_selector = new ToggleGroup();
    peak.setToggleGroup(peakTime_selector);
    peak.setUserData(true);
    offPeak.setToggleGroup(peakTime_selector);
    offPeak.setUserData(false);
    peak.setSelected(true);
    peaktime_radio_box = new VBox();
    peaktime_radio_box.setSpacing(15);
    peaktime_radio_box.getChildren().addAll(peak, offPeak);
    peak_box = new HBox();
    peak_box.setPadding(new Insets(10, 15, 10, 15));
    peak_box.setSpacing(10);
    peak_box.getChildren().add(new Text("Peak/off−peak: "));
    peak_box.getChildren().add(peaktime_radio_box);
    peakTimeSelector = peakTime_selector;
    return peak_box;
  }
  private HBox getDurationBox() {
    duration_field = new TextField();
    duration_field.setText("120");
    duration_box = new HBox();
    duration_box.setPadding(new Insets(10, 15, 10, 15));
    duration_box.setSpacing(10);
    duration_box.getChildren().addAll(new Text("Duration: "), duration_field, new T
ext("seconds"));
    return duration_box;
  }

  private HBox getMapBox() {
    map_list = new ComboBox();
    map_list.getItems().addAll("Small Town", "New York", "London");
    map_list.setValue("Small Town");
    map_box = new HBox();
    map_box.getChildren().addAll(new Text("Map: "), map_list);
    return map_box;
  }

  private BorderPane getButtonPane() {
    startSim = new Button("Start");
    showResults = new Button("Result");
    startSim.setPrefSize(100, 50);
    showResults.setPrefSize(100, 50);
    showResults.setDisable(true);
    button_box = new HBox();
    button_box.setPadding(new Insets(10, 15, 10, 15));
    button_box.setSpacing(25);
    button_box.getChildren().addAll(startSim, showResults);
    button_pane = new BorderPane();
    button_pane.setCenter(button_box);
    return button_pane;
  }

  private VBox getContainerBox() {
    container = new VBox();
    container.setPadding(new Insets(10, 15, 10, 15));
    container.setSpacing(15);
    container.getChildren().addAll(getPolicyBox(), getPeakTimeBox(), getDuration
Box(), getMapBox(), getButtonPane());
    return container;
  }

  public String getMapValue() {
    return this.map_list.getValue().toString();
  }

  public void disableStartButton() {
    this.startSim.setDisable(true);
  }

  public void enableStartButton() {
    this.startSim.setDisable(false);
  }

  public void disableResultButton() {
    this.showResults.setDisable(true);
  }

  public void enableResultButton() {
    this.showResults.setDisable(false);
  }

  public String getSelectedPolicyText() {
    RadioButton temp = (RadioButton) this.policies_selector.getSelectedToggle();
    return temp.getText();
  }

}
```

```
/*
 * 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.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
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.Junction;
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();
        drawGrass();
        drawRoads();
        drawLanes();
        drawJunctions();
        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(), ri
ghtEndPoint.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) {
      int numLanes = road.getLanes().size();
      gc.setLineWidth(1);
      gc.setStroke(Color.WHITE);
      for(int i = 0 ; i < numLanes - 1 ; i++){
        Lane lane = road.getLanes().get(i);
        gc.strokeLine(lane.getLaneStart().x, lane.getLaneStart().y, lane.getLane
Stop().x, lane.getLaneStop().y);
      }
    }
  }

  private void drawJunctions() {
```

```java
    List<Junction> junctions = this.simulation.getMap().getJunctions();

    for (Junction junction : junctions) {

      JunctionRenderer renderer = new JunctionRenderer(gc, junction);
      renderer.render();

    }
  }

  private void drawVehicles() {
    List<Vehicle> vehicles = this.simulation.getVehicles();
    for (Vehicle vehicle : vehicles) {
      if (Car.class.isInstance(vehicle)) {
        Double angle = vehicle.getDirectionVector().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.getDirectionVector().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)
  }

  private void drawGrass(){
      gc.setFill(Color.GREEN);
      gc.fillRect(0, 0, 800, 600);
  }

  private void drawLaneSeparator(){

  }
}
```

```
/*
 * 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.ColumnConstraints;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.RowConstraints;
import javafx.scene.paint.Color;
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, int round,
 String map_no, String policy, String duration, boolean isPeaktime){
    double temp;
    if(isPeaktime){
      temp = Double.parseDouble(duration)/0.5;
    }else{
      temp = Double.parseDouble(duration)/1.5;
    }
    int totalVehicle = (int)temp - 1;
    initModality(Modality.NONE);
    initOwner(primaryStage);
    GridPane pane = new GridPane();
    RowConstraints row1 = new RowConstraints();
    RowConstraints row2 = new RowConstraints();
    RowConstraints row3 = new RowConstraints();
    RowConstraints row4 = new RowConstraints();
    RowConstraints row5 = new RowConstraints();
    RowConstraints row6 = new RowConstraints();
    RowConstraints row7 = new RowConstraints();
    RowConstraints row8 = new RowConstraints();
    RowConstraints row9 = new RowConstraints();
    row1.setPercentHeight(100/9);
    row2.setPercentHeight(100/9);
    row3.setPercentHeight(100/9);
    row4.setPercentHeight(100/9);
    row5.setPercentHeight(100/9);
    row6.setPercentHeight(100/9);
    row7.setPercentHeight(100/9);
    row8.setPercentHeight(100/9);
    row9.setPercentHeight(100/9);
    pane.getRowConstraints().addAll(row1, row2, row3, row4, row5, row6, row7, ro
w8, row9);
    ColumnConstraints column1 = new ColumnConstraints();
    column1.setPercentWidth(55);
```

```java
    ColumnConstraints column2 = new ColumnConstraints();
    column2.setPercentWidth(45);
    pane.getColumnConstraints().addAll(column1, column2);
    pane.setGridLinesVisible(true);
    pane.add(new Text("Simulation#"), 0, 0);
    pane.add(new Text("" + round), 1, 0);
    pane.add(new Text("Map:"), 0, 1);
    pane.add(new Text("" + map_no), 1, 1);
    pane.add(new Text("Policy:"), 0, 2);
    pane.add(new Text("" + policy), 1, 2);
    pane.add(new Text("Duration:"), 0, 3);
    pane.add(new Text("" + duration + " seconds"), 1, 3);
    pane.add(new Text("Total vehicle: "), 0, 4);
    pane.add(new Text("" + totalVehicle), 1, 4);
    pane.add(new Text("Number of vehicle(s) exited: "), 0, 5);
    pane.add(new Text("" + simulation.getExitedVehicles().size()), 1, 5);
    pane.add(new Text("Average time:"), 0, 6);
    pane.add(simulation.averageTime(), 1, 6);
    pane.add(new Text("Longest time:"), 0, 7);
    pane.add(simulation.longestTime(), 1, 7);
    pane.add(new Text("Shortest time:"), 0, 8);
    pane.add(simulation.shortestTime(), 1, 8);

    Scene dialogScene = new Scene(pane, 600, 240, Color.WHITE);
    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 java.security.Policy;

import trafficsimulator.core.Lane;

import trafficsimulator.policies.TrafficPolicy;
import trafficsimulator.utils.Point;

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


 public static final int GREEN_DURATION = 60;
 public static final int YELLOW_DURATION = 50;
 public static final int RED_DURATION = 100;

  public enum State {

    GREEN, YELLOW, RED, REDYELLOW
  }

  private final TrafficPolicy policy;



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

  public TrafficLight(Lane lane, TrafficPolicy policy) {
    this.lane = lane;

    this.policy = policy;

  }

  public State getState() {
    return state;
  }

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

  public Lane getLane() {
    return lane;
  }


  public TrafficPolicy getPolicy() {
        return policy;
    }


  public Point getPosition() {
    return lane.getEndPoint();
  }

  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;
    }
  }

}
```

```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.junctions;

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

import trafficsimulator.policies.TrafficPolicy;

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

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

  public TrafficLightJunction(TrafficPolicy policy) {
    this.policy = policy;
  }

  public List<TrafficLight> getTrafficLights(){
    return trafficLights;
  }

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

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

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

  @Override
  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));
    }
  }

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

    if(policy.isCongestionControl()){
        stepCounter++;
//
        if(stepCounter == TrafficLight.GREEN_DURATION || stepCounter > TrafficLi
ght.GREEN_DURATION){

            activeTrafficLight.setState(TrafficLight.State.RED);
            activateTrafficLight(getMostCongested());
            stepCounter = 0;
        }else if(stepCounter < TrafficLight.GREEN_DURATION){

            activeTrafficLight.setState(TrafficLight.State.GREEN);

        }else{

            activeTrafficLight.setState(TrafficLight.State.RED);
        }

    }else{
    stepCounter++;

    if (activeTrafficLight.getState() == TrafficLight.State.GREEN && stepCounter
 == activeTrafficLight.getPolicy().getGreenLightDuration()) {
      activeTrafficLight.nextState();
      stepCounter = 0;
    } else if (activeTrafficLight.getState() == TrafficLight.State.YELLOW && ste
pCounter == activeTrafficLight.getPolicy().getYellowLightDuration()) {
      activateNextTrafficLight();
    } else if (activeTrafficLight.getState() == TrafficLight.State.REDYELLOW &&
stepCounter == activeTrafficLight.getPolicy().getRedYellowDuration()) {
      activeTrafficLight.nextState();
      stepCounter = 0;
    } else if (activeTrafficLight.getState() == TrafficLight.State.RED && stepCo
unter == activeTrafficLight.getPolicy().getRedLightDuration()) {
      activeTrafficLight.nextState();
      stepCounter = 0;
    }
    }
  }

//  private boolean checkForCongestion(Lane lane){
//
//      List<Vehicle> vehiclesOnLane = lane.getVehicles();
//        double totalLengthOfVehicle = 0;
//        for(Vehicle v : vehiclesOnLane){
//             totalLengthOfVehicle = totalLengthOfVehicle + v.getSize().height;
//        }
//
//
//        return (totalLengthOfVehicle == (lane.getLaneLength()*0.3))||(totalLen
gthOfVehicle > (lane.getLaneLength()*0.3));
//  }

  private TrafficLight getMostCongested(){

      HashMap<Double, TrafficLight> hm = new HashMap<>();
        for(TrafficLight tf : trafficLights){
            List<Vehicle> vehiclesOnLane = tf.getLane().getVehicles();
            double totalLengthOfVehicle = 0;
            for(Vehicle v : vehiclesOnLane){
             totalLengthOfVehicle = totalLengthOfVehicle + v.getSize().height;
        }
            System.out.println( "Value: " +totalLengthOfVehicle);
            hm.put(totalLengthOfVehicle,tf);
        }

        Iterator<Double> keySetIterator = hm.keySet().iterator();

        double largest = 0;
        while(keySetIterator.hasNext()){
            Double key = keySetIterator.next();
            System.out.println("Key: "+ key+ "Value: " +hm.get(key));
            if(largest<key){
                largest = key;
            }
        }

        return hm.get(largest);
  }
}
```

```
/*
 * 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.policies;

/**
 *
 * @author Eddy
 */
public class TrafficPolicy {

    private int greenLightDuration;
    private int yellowLightDuration;
    private int redYellowDuration;
    private int redLightDuration;


    private final boolean congestionControl;
    private final boolean peakTime;


    public boolean isCongestionControl() {
        return congestionControl;
    }


    public TrafficPolicy(boolean peaktime, boolean congestionControl) {
        this.peakTime = peaktime;
        this.congestionControl = congestionControl;
        if(!congestionControl){
        if(this.peakTime){

            this.setGreenLightDuration(100);
            this.setYellowLightDuration(10);
            this.setRedLightDuration(100);
            this.setRedYellowDuration(30);
        }else if(!this.peakTime){
            this.setGreenLightDuration(50);
            this.setYellowLightDuration(20);
            this.setRedLightDuration(50);
            this.setRedYellowDuration(30);
        }
        }
    }

    public int getGreenLightDuration() {
        return greenLightDuration;
    }

    private void setGreenLightDuration(int greenLightDuration) {
        this.greenLightDuration = greenLightDuration;
    }

    public int getYellowLightDuration() {
        return yellowLightDuration;
    }
    private void setYellowLightDuration(int yellowLightDuration) {
        this.yellowLightDuration = yellowLightDuration;
    }

    public int getRedYellowDuration() {
        return redYellowDuration;
    }

    private void setRedYellowDuration(int redYellowDuration) {
        this.redYellowDuration = redYellowDuration;
    }

    public int getRedLightDuration() {
        return redLightDuration;
    }

    private void setRedLightDuration(int redLightDuration) {
        this.redLightDuration = redLightDuration;
    }
}
```

```
/*
 * 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 java.util.ArrayList;
import java.util.List;
import trafficsimulator.core.Driver;
import trafficsimulator.core.Junction;
import trafficsimulator.core.Lane;
import trafficsimulator.core.Road;
import trafficsimulator.core.Simulation;
import trafficsimulator.drivers.CautiousDriver;
import trafficsimulator.drivers.NormalDriver;
import trafficsimulator.drivers.RecklessDriver;
import trafficsimulator.junctions.TrafficLightJunction;
import trafficsimulator.policies.TrafficPolicy;
import trafficsimulator.utils.Point;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;
import java.util.Random;


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

  private List<Lane> entryLanes = new ArrayList<>();
  private List<String> vehicleTypes = new ArrayList<>();
  private Random randomGenerator = new Random();

  public Simulation1(boolean peaktime, boolean congestionControl, int longestSim
ulationTime) {
    super(peaktime, congestionControl, longestSimulationTime);
  }

  @Override
  protected void init() {
    Road r1 = new Road(new Point(425, 50), new Point(425, 275));
    Lane l11 = r1.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l11);
    Lane l12 = r1.addLane(Lane.Direction.OPPOSITE);
    Road r2 = new Road(new Point(425, 325), new Point(425, 550));
    Lane l21 = r2.addLane(Lane.Direction.IDENTICAL);
    Lane l22 = r2.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l22);
    Road r3 = new Road(new Point(150, 275), new Point(375, 275));
    Lane l31 = r3.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l31);
    Lane l32 = r3.addLane(Lane.Direction.OPPOSITE);
    Road r4 = new Road(new Point(425, 275), new Point(650, 275));
    Lane l41 = r4.addLane(Lane.Direction.IDENTICAL);
    Lane l42 = r4.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l42);
    TrafficPolicy policy = new TrafficPolicy(peaktime,congestionControl);

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

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

    longestSimulationTime = 5000;
    int vehicleFrequency;
    if(peaktime) {
      vehicleFrequency = 5;
    } else {
      vehicleFrequency = 15;
    }

    vehicleTypes.add("cautiousCar");
    vehicleTypes.add("normalCar");
    vehicleTypes.add("recklessCar");
    vehicleTypes.add("cautiousBus");
    vehicleTypes.add("normalBus");
    vehicleTypes.add("recklessBus");

    for (int i = 0; i < longestSimulationTime; i += vehicleFrequency) {
      int randomLaneindex = randomGenerator.nextInt(entryLanes.size());
      int randomVehicleIndex = randomGenerator.nextInt(vehicleTypes.size());
      String vehicleType = vehicleTypes.get(randomVehicleIndex);
        switch (vehicleType) {
          case "cautiousCar":
            Driver cautiousC = new CautiousDriver(Integer.toString(i));
            addVehicle(new Car(cautiousC), entryLanes.get(randomLaneindex), i);
            break;
          case "normalCar":
            Driver normalC = new NormalDriver(Integer.toString(i));
            addVehicle(new Car(normalC), entryLanes.get(randomLaneindex), i);
            break;
          case "recklessCar":
            Driver recklessC = new RecklessDriver(Integer.toString(i));
            addVehicle(new Car(recklessC), entryLanes.get(randomLaneindex), i);
            break;
          case "cautiousBus":
            Driver cautiousB = new CautiousDriver(Integer.toString(i));
            addVehicle(new Bus(cautiousB), entryLanes.get(randomLaneindex), i);
            break;
          case "normalBus":
            Driver normalB = new NormalDriver(Integer.toString(i));
            addVehicle(new Bus(normalB), entryLanes.get(randomLaneindex), i);
            break;
          case "recklessBus":
            Driver recklessB = new RecklessDriver(Integer.toString(i));
            addVehicle(new Bus(recklessB), entryLanes.get(randomLaneindex), i);
            break;
        }
    }

  }

}
```

```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.simulations;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import trafficsimulator.core.Driver;
import trafficsimulator.core.Junction;
import trafficsimulator.core.Lane;
import trafficsimulator.core.Road;
import trafficsimulator.core.Simulation;
import trafficsimulator.drivers.CautiousDriver;
import trafficsimulator.drivers.NormalDriver;
import trafficsimulator.drivers.RecklessDriver;
import trafficsimulator.junctions.TrafficLightJunction;
import trafficsimulator.policies.TrafficPolicy;
import trafficsimulator.utils.Point;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

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


  private List<Lane> entryLanes = new ArrayList<>();
  private List<String> vehicleTypes = new ArrayList<>();
  private Random randomGenerator = new Random();

  public Simulation2(boolean peaktime, boolean congestionControl, int longestSim
ulationTime) {
    super(peaktime, congestionControl, longestSimulationTime);
  }


  @Override
  protected void init() {
    Road r1 = new Road(new Point(0, 100), new Point(100, 100));
    Lane l11 = r1.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l11);
    Lane l12 = r1.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l12);
    Lane l13 = r1.addLane(Lane.Direction.OPPOSITE);
    Lane l14 = r1.addLane(Lane.Direction.OPPOSITE);

    Road r2 = new Road(new Point(150, 100), new Point(350, 100));
    Lane l21 = r2.addLane(Lane.Direction.IDENTICAL);
    Lane l22 = r2.addLane(Lane.Direction.IDENTICAL);
    Lane l23 = r2.addLane(Lane.Direction.OPPOSITE);
    Lane l24 = r2.addLane(Lane.Direction.OPPOSITE);

    Road r3 = new Road(new Point(450, 100), new Point(650, 100));
    Lane l31 = r3.addLane(Lane.Direction.IDENTICAL);
    Lane l32 = r3.addLane(Lane.Direction.IDENTICAL);
    Lane l33 = r3.addLane(Lane.Direction.OPPOSITE);
    Lane l34 = r3.addLane(Lane.Direction.OPPOSITE);

    Road r4 = new Road(new Point(700, 100), new Point(800, 100));
    Lane l41 = r4.addLane(Lane.Direction.IDENTICAL);
    Lane l42 = r4.addLane(Lane.Direction.IDENTICAL);
    Lane l43 = r4.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l43);
    Lane l44 = r4.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l44);

    Road r5 = new Road(new Point(0, 400), new Point(100, 400));
    Lane l51 = r5.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l51);
    Lane l52 = r5.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l52);
    Lane l53 = r5.addLane(Lane.Direction.OPPOSITE);
    Lane l54 = r5.addLane(Lane.Direction.OPPOSITE);

    Road r6 = new Road(new Point(150, 400), new Point(350, 400));
    Lane l61 = r6.addLane(Lane.Direction.IDENTICAL);
    Lane l62 = r6.addLane(Lane.Direction.IDENTICAL);
    Lane l63 = r6.addLane(Lane.Direction.OPPOSITE);
    Lane l64 = r6.addLane(Lane.Direction.OPPOSITE);

    Road r7 = new Road(new Point(450, 400), new Point(650, 400));
    Lane l71 = r7.addLane(Lane.Direction.IDENTICAL);
    Lane l72 = r7.addLane(Lane.Direction.IDENTICAL);
    Lane l73 = r7.addLane(Lane.Direction.OPPOSITE);
    Lane l74 = r7.addLane(Lane.Direction.OPPOSITE);

    Road r8 = new Road(new Point(700, 400), new Point(800, 400));
    Lane l81 = r8.addLane(Lane.Direction.IDENTICAL);
    Lane l82 = r8.addLane(Lane.Direction.IDENTICAL);
    Lane l83 = r8.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l83);
    Lane l84 = r8.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l84);

    Road r9 = new Road(new Point(150, 0), new Point(150, 100));
    Lane l91 = r9.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l91);
    Lane l92 = r9.addLane(Lane.Direction.OPPOSITE);

    Road r10 = new Road(new Point(150, 200), new Point(150, 400));
    Lane l101 = r10.addLane(Lane.Direction.IDENTICAL);
    Lane l102 = r10.addLane(Lane.Direction.OPPOSITE);

    Road r11 = new Road(new Point(150, 500), new Point(150, 600));
    Lane l111 = r11.addLane(Lane.Direction.IDENTICAL);
    Lane l112 = r11.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l112);

    Road r12 = new Road(new Point(450,0), new Point(450,100));
    Lane l121 = r12.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l121);
    Lane l122 = r12.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l122);
```

```java
    Lane l123 = r12.addLane(Lane.Direction.OPPOSITE);
    Lane l124 = r12.addLane(Lane.Direction.OPPOSITE);

    Road r13 = new Road(new Point(450, 200), new Point(450,400));
    Lane l131 = r13.addLane(Lane.Direction.IDENTICAL);
    Lane l132 = r13.addLane(Lane.Direction.IDENTICAL);
    Lane l133 = r13.addLane(Lane.Direction.OPPOSITE);
    Lane l134 = r13.addLane(Lane.Direction.OPPOSITE);

    Road r14 = new Road(new Point(450, 500), new Point(450, 600));
    Lane l141 = r14.addLane(Lane.Direction.IDENTICAL);
    Lane l142 = r14.addLane(Lane.Direction.IDENTICAL);
    Lane l143 = r14.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l143);
    Lane l144 = r14.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l144);

    Road r15 = new Road(new Point(700, 0), new Point(700, 100));
    Lane l151 = r15.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l151);
    Lane l152 = r15.addLane(Lane.Direction.OPPOSITE);

    Road r16 = new Road(new Point(700, 200), new Point(700, 400));
    Lane l161 = r16.addLane(Lane.Direction.IDENTICAL);
    Lane l162 = r16.addLane(Lane.Direction.OPPOSITE);

    Road r17 = new Road(new Point(700, 500), new Point(700, 600));
    Lane l171 = r17.addLane(Lane.Direction.IDENTICAL);
    Lane l172 = r17.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l172);

   TrafficPolicy policy = new TrafficPolicy(peaktime,congestionControl);

    Junction j1 = new TrafficLightJunction(policy);
    j1.connect(l11, l21);
    j1.connect(l11, l92);
    j1.connect(l12, l22);
    j1.connect(l12, l101);
    j1.connect(l23, l13);
    j1.connect(l23, l92);
    j1.connect(l24, l14);
    j1.connect(l24, l101);
    j1.connect(l91, l13);
    j1.connect(l91, l21);
    j1.connect(l91, l101);
    j1.connect(l102, l14);
    j1.connect(l102, l92);
    j1.connect(l102, l22);


    Junction j2 = new TrafficLightJunction(policy);
    j2.connect(l21, l31);
    j2.connect(l21, l124);
    j2.connect(l21, l131);
    j2.connect(l22, l32);
    j2.connect(l22, l123);
    j2.connect(l22, l132);
    j2.connect(l33, l23);
    j2.connect(l33, l123);
    j2.connect(l33, l132);
    j2.connect(l34, l24);
    j2.connect(l34, l124);
    j2.connect(l34, l131);
    j2.connect(l121, l24);
    j2.connect(l121, l31);
    j2.connect(l121, l131);
    j2.connect(l122, l23);
    j2.connect(l122, l32);
    j2.connect(l122, l132);
    j2.connect(l133, l23);
    j2.connect(l133, l32);
    j2.connect(l133, l123);
    j2.connect(l134, l24);
    j2.connect(l134, l31);
    j2.connect(l134, l124);

    Junction j3 = new TrafficLightJunction(policy);
    j3.connect(l31, l41);
    j3.connect(l31, l152);
    j3.connect(l32, l42);
    j3.connect(l32, l161);
    j3.connect(l43, l33);
    j3.connect(l43, l152);
    j3.connect(l44, l34);
    j3.connect(l44, l161);
    j3.connect(l151, l33);
    j3.connect(l151, l41);
    j3.connect(l151, l161);
    j3.connect(l162, l34);
    j3.connect(l162, l152);
    j3.connect(l162, l42);

    Junction j4 = new TrafficLightJunction(policy);
    j4.connect(l51, l61);
    j4.connect(l51, l102);
    j4.connect(l52, l62);
    j4.connect(l52, l111);
    j4.connect(l63, l53);
    j4.connect(l63, l102);
    j4.connect(l64, l54);
    j4.connect(l64, l111);
    j4.connect(l101, l61);
    j4.connect(l101, l53);
    j4.connect(l101, l111);
    j4.connect(l112, l54);
    j4.connect(l112, l102);
    j4.connect(l112, l62);

    Junction j5 = new TrafficLightJunction(policy);
    j5.connect(l61, l71);
    j5.connect(l61, l134);
    j5.connect(l61, l141);
    j5.connect(l62, l72);
    j5.connect(l62, l133);
    j5.connect(l62, l142);
    j5.connect(l73, l63);
    j5.connect(l73, l133);
    j5.connect(l73, l142);
    j5.connect(l74, l64);
    j5.connect(l74, l134);
    j5.connect(l74, l141);
    j5.connect(l131, l64);
    j5.connect(l131, l71);
    j5.connect(l131, l141);
    j5.connect(l132, l63);
    j5.connect(l132, l72);
    j5.connect(l132, l142);
    j5.connect(l143, l63);
    j5.connect(l143, l72);
    j5.connect(l143, l133);
    j5.connect(l144, l64);
    j5.connect(l144, l71);
    j5.connect(l144, l134);

    Junction j6 = new TrafficLightJunction(policy);
    j6.connect(l71, l81);
    j6.connect(l71, l162);
    j6.connect(l72, l82);
    j6.connect(l72, l171);
    j6.connect(l83, l73);
    j6.connect(l83, l162);
    j6.connect(l84, l74);
    j6.connect(l84, l171);
    j6.connect(l161, l73);
    j6.connect(l161, l81);
    j6.connect(l161, l171);
    j6.connect(l172, l74);
    j6.connect(l172, l162);
    j6.connect(l172, l82);


    map.addRoad(r1);
    map.addRoad(r2);
    map.addRoad(r3);
    map.addRoad(r4);
    map.addRoad(r5);
    map.addRoad(r6);
    map.addRoad(r7);
    map.addRoad(r8);
    map.addRoad(r9);
    map.addRoad(r10);
    map.addRoad(r11);
    map.addRoad(r12);
    map.addRoad(r13);
    map.addRoad(r14);
    map.addRoad(r15);
    map.addRoad(r16);
    map.addRoad(r17);
    map.addJunction(j1);
    map.addJunction(j2);
    map.addJunction(j3);
    map.addJunction(j4);
    map.addJunction(j5);
    map.addJunction(j6);

    longestSimulationTime = 5000;
    int vehicleFrequency;
    if(peaktime) {
      vehicleFrequency = 5;
    } else {
      vehicleFrequency = 15;
    }

    vehicleTypes.add("cautiousCar");
    vehicleTypes.add("normalCar");
    vehicleTypes.add("recklessCar");
    vehicleTypes.add("cautiousBus");
    vehicleTypes.add("normalBus");
    vehicleTypes.add("recklessBus");

    for (int i = 0; i < longestSimulationTime; i += vehicleFrequency) {
      int randomLaneindex = randomGenerator.nextInt(entryLanes.size());
      int randomVehicleIndex = randomGenerator.nextInt(vehicleTypes.size());
      String vehicleType = vehicleTypes.get(randomVehicleIndex);
        switch (vehicleType) {
          case "cautiousCar":
            Driver cautiousC = new CautiousDriver(Integer.toString(i));
            addVehicle(new Car(cautiousC), entryLanes.get(randomLaneindex), i);
            break;
          case "normalCar":
            Driver normalC = new NormalDriver(Integer.toString(i));
            addVehicle(new Car(normalC), entryLanes.get(randomLaneindex), i);
            break;
          case "recklessCar":
            Driver recklessC = new RecklessDriver(Integer.toString(i));
            addVehicle(new Car(recklessC), entryLanes.get(randomLaneindex), i);
            break;
          case "cautiousBus":
            Driver cautiousB = new CautiousDriver(Integer.toString(i));
            addVehicle(new Bus(cautiousB), entryLanes.get(randomLaneindex), i);
            break;
          case "normalBus":
            Driver normalB = new NormalDriver(Integer.toString(i));
            addVehicle(new Bus(normalB), entryLanes.get(randomLaneindex), i);
            break;
          case "recklessBus":
            Driver recklessB = new RecklessDriver(Integer.toString(i));
            addVehicle(new Bus(recklessB), entryLanes.get(randomLaneindex), i);
            break;
        }
    }

  }

}
```

```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.simulations;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import trafficsimulator.core.Driver;
import trafficsimulator.core.Junction;
import trafficsimulator.core.Lane;
import trafficsimulator.core.Road;
import trafficsimulator.core.Simulation;
import trafficsimulator.drivers.CautiousDriver;
import trafficsimulator.drivers.NormalDriver;
import trafficsimulator.drivers.RecklessDriver;
import trafficsimulator.junctions.TrafficLightJunction;
import trafficsimulator.policies.TrafficPolicy;
import trafficsimulator.utils.Point;
import trafficsimulator.vehicles.Bus;
import trafficsimulator.vehicles.Car;

/**
 *
 * @author yukolthep
 */
public class Simulation3 extends Simulation {

  private List<Lane> entryLanes = new ArrayList<>();
  private List<String> vehicleTypes = new ArrayList<>();
  private Random randomGenerator = new Random();

  public Simulation3(boolean peaktime, boolean congestionControl, int longestSim
ulationTime) {
    super(peaktime, congestionControl, longestSimulationTime);
  }


  @Override
  protected void init() {
    Road r1 = new Road(new Point(200, 0), new Point(200, 100));
    Lane l11 = r1.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l11);
    Lane l12 = r1.addLane(Lane.Direction.OPPOSITE);

    Road r2 = new Road(new Point(200, 150), new Point(200, 300));
    Lane l21 = r2.addLane(Lane.Direction.IDENTICAL);
    Lane l22 = r2.addLane(Lane.Direction.OPPOSITE);

    Road r3 = new Road(new Point(200, 325), new Point(190, 400));
    Lane l31 = r3.addLane(Lane.Direction.IDENTICAL);
    Lane l32 = r3.addLane(Lane.Direction.OPPOSITE);

    Road r4 = new Road(new Point(185,423), new Point(130,600));
    Lane l41 = r4.addLane(Lane.Direction.IDENTICAL);
    Lane l42 = r4.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l42);
    Road r5 = new Road(new Point(0,0), new Point(150, 100));
    Lane l51 = r5.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l51);

    Road r6 = new Road(new Point(150,150), new Point(0,250));
    Lane l61 = r6.addLane(Lane.Direction.IDENTICAL);

    Road r7 = new Road(new Point(0, 400), new Point(140, 400));
    Lane l71 = r7.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l71);

    Road r8 = new Road(new Point(200, 100), new Point(275, 100));
    Lane l81 = r8.addLane(Lane.Direction.IDENTICAL);
    Lane l82 = r8.addLane(Lane.Direction.OPPOSITE);

    Road r9 = new Road(new Point(200, 300), new Point(275, 300));
    Lane l91 = r9.addLane(Lane.Direction.IDENTICAL);

    Road r10 = new Road(new Point(290, 100), new Point(400, 0));
    Lane l101 = r10.addLane(Lane.Direction.IDENTICAL);

    Road r11 = new Road(new Point(325, 100), new Point(400, 100));
    Lane l111 = r11.addLane(Lane.Direction.IDENTICAL);
    Lane l112 = r11.addLane(Lane.Direction.OPPOSITE);

    Road r12 = new Road (new Point(325, 150), new Point(325, 300));
    Lane l121 = r12.addLane(Lane.Direction.IDENTICAL);
    Lane l122 = r12.addLane(Lane.Direction.OPPOSITE);

    Road r13 = new Road (new Point(325, 325), new Point(250, 600));
    Lane l131 = r13.addLane(Lane.Direction.IDENTICAL);
    Lane l132 = r13.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l132);

    Road r14 = new Road(new Point(535, 0), new Point(425, 100));
    Lane l141 = r14.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l141);

    Road r15 = new Road(new Point(425, 150), new Point(500, 260));
    Lane l151 = r15.addLane(Lane.Direction.IDENTICAL);

    Road r16 = new Road(new Point(325, 300), new Point(375, 300));
    Lane l161 = r16.addLane(Lane.Direction.IDENTICAL);

    Road r17 = new Road(new Point(425, 300), new Point(485, 275));
    Lane l171 = r17.addLane(Lane.Direction.IDENTICAL);
    Lane l172 = r17.addLane(Lane.Direction.OPPOSITE);

    Road r18 = new Road(new Point(425, 350), new Point(425, 600));
    Lane l181 = r18.addLane(Lane.Direction.IDENTICAL);
    Lane l182 = r18.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l182);

    Road r19 = new Road(new Point(530, 300), new Point(605, 400));
    Lane l191 = r19.addLane(Lane.Direction.IDENTICAL);

    Road r20 = new Road(new Point(425, 100), new Point(605, 100));
    Lane l201 = r20.addLane(Lane.Direction.IDENTICAL);
    Lane l202 = r20.addLane(Lane.Direction.OPPOSITE);

    Road r21 = new Road(new Point(655, 0), new Point(655, 100));
    Lane l211 = r21.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l211);
    Lane l212 = r21.addLane(Lane.Direction.OPPOSITE);

    Road r22 = new Road(new Point(655, 150), new Point(655, 400));
    Lane l221 = r22.addLane(Lane.Direction.IDENTICAL);
    Lane l222 = r22.addLane(Lane.Direction.OPPOSITE);

    Road r23 = new Road(new Point(655, 450), new Point(655, 600));
    Lane l231 = r23.addLane(Lane.Direction.IDENTICAL);
    Lane l232 = r23.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l232);

    Road r24 = new Road(new Point(655, 100), new Point(800, 100));
    Lane l241 = r24.addLane(Lane.Direction.IDENTICAL);
    Lane l242 = r24.addLane(Lane.Direction.OPPOSITE);
    entryLanes.add(l242);

    Road r25 = new Road(new Point(800, 450), new Point(655, 450));
    Lane l251 = r25.addLane(Lane.Direction.IDENTICAL);
    entryLanes.add(l251);
    Lane l252 = r25.addLane(Lane.Direction.OPPOSITE);



    TrafficPolicy policy = new TrafficPolicy(peaktime,congestionControl);
    Junction j1 = new TrafficLightJunction(policy);
    j1.connect(l11, l21);
    j1.connect(l11, l61);
    j1.connect(l11, l81);
    j1.connect(l22, l12);
    j1.connect(l22, l81);
    j1.connect(l51, l21);
    j1.connect(l51, l81);
    j1.connect(l82, l12);
    j1.connect(l82, l21);
    j1.connect(l82, l61);

    Junction j2 = new TrafficLightJunction(policy);
    j2.connect(l21, l31);
    j2.connect(l21, l91);
    j2.connect(l32, l22);
    j2.connect(l32, l91);

    Junction j3 = new TrafficLightJunction(policy);
    j3.connect(l31, l41);
    j3.connect(l42, l32);
    j3.connect(l71, l32);
    j3.connect(l71, l41);

    Junction j4 = new TrafficLightJunction(policy);
    j4.connect(l81, l101);
    j4.connect(l81, l111);
    j4.connect(l81, l121);
    j4.connect(l112, l82);
    j4.connect(l112, l121);
    j4.connect(l122, l82);
    j4.connect(l122, l101);
    j4.connect(l122, l111);

    Junction j5 = new TrafficLightJunction(policy);
    j5.connect(l91, l122);
    j5.connect(l91, l161);
    j5.connect(l121, l131);
    j5.connect(l121, l161);
    j5.connect(l132, l122);
    j5.connect(l132, l161);

    Junction j6 = new TrafficLightJunction(policy);
    j6.connect(l111, l151);
    j6.connect(l111, l201);
    j6.connect(l141, l112);
    j6.connect(l202, l112);

    Junction j7 = new TrafficLightJunction(policy);
    j7.connect(l161, l171);
    j7.connect(l161, l181);
    j7.connect(l172, l181);
    j7.connect(l182, l171);

    Junction j8 = new TrafficLightJunction(policy);
    j8.connect(l151, l172);
    j8.connect(l151, l191);
    j8.connect(l171, l191);

    Junction j9 = new TrafficLightJunction(policy);
    j9.connect(l201, l212);
    j9.connect(l201, l221);
    j9.connect(l201, l241);
    j9.connect(l211, l221);
    j9.connect(l211, l202);
    j9.connect(l211, l241);
    j9.connect(l222, l212);
    j9.connect(l222, l202);
    j9.connect(l222, l241);
    j9.connect(l242, l212);
    j9.connect(l242, l221);
    j9.connect(l242, l202);

    Junction j10 = new TrafficLightJunction(policy);
    j10.connect(l191, l231);
    j10.connect(l191, l252);
    j10.connect(l221, l231);
    j10.connect(l221, l252);
    j10.connect(l232, l222);
    j10.connect(l232, l252);
    j10.connect(l251, l222);
    j10.connect(l251, l231);



    map.addRoad(r1);
    map.addRoad(r2);
    map.addRoad(r3);
    map.addRoad(r4);
    map.addRoad(r5);
    map.addRoad(r6);
    map.addRoad(r7);
    map.addRoad(r8);
    map.addRoad(r9);
    map.addRoad(r10);
    map.addRoad(r11);
    map.addRoad(r12);
    map.addRoad(r13);
    map.addRoad(r14);
    map.addRoad(r15);
    map.addRoad(r16);
    map.addRoad(r17);
    map.addRoad(r18);
    map.addRoad(r19);
    map.addRoad(r20);
    map.addRoad(r21);
    map.addRoad(r22);
    map.addRoad(r23);
    map.addRoad(r24);
    map.addRoad(r25);
    map.addJunction(j1);
    map.addJunction(j2);
    map.addJunction(j3);
    map.addJunction(j4);
    map.addJunction(j5);
    map.addJunction(j6);
    map.addJunction(j7);
    map.addJunction(j8);
    map.addJunction(j9);
    map.addJunction(j10);

    longestSimulationTime = 5000;
    int vehicleFrequency;
    if(peaktime) {
      vehicleFrequency = 5;
    } else {
      vehicleFrequency = 15;
    }

    vehicleTypes.add("cautiousCar");
    vehicleTypes.add("normalCar");
    vehicleTypes.add("recklessCar");
    vehicleTypes.add("cautiousBus");
    vehicleTypes.add("normalBus");
    vehicleTypes.add("recklessBus");

    for (int i = 0; i < longestSimulationTime; i += vehicleFrequency) {
      int randomLaneindex = randomGenerator.nextInt(entryLanes.size());
      int randomVehicleIndex = randomGenerator.nextInt(vehicleTypes.size());
      String vehicleType = vehicleTypes.get(randomVehicleIndex);
        switch (vehicleType) {
          case "cautiousCar":
            Driver cautiousC = new CautiousDriver(Integer.toString(i));
            addVehicle(new Car(cautiousC), entryLanes.get(randomLaneindex), i);
            break;
          case "normalCar":
            Driver normalC = new NormalDriver(Integer.toString(i));
            addVehicle(new Car(normalC), entryLanes.get(randomLaneindex), i);
            break;
          case "recklessCar":
            Driver recklessC = new RecklessDriver(Integer.toString(i));
            addVehicle(new Car(recklessC), entryLanes.get(randomLaneindex), i);
            break;
          case "cautiousBus":
            Driver cautiousB = new CautiousDriver(Integer.toString(i));
            addVehicle(new Bus(cautiousB), entryLanes.get(randomLaneindex), i);
            break;
          case "normalBus":
            Driver normalB = new NormalDriver(Integer.toString(i));
            addVehicle(new Bus(normalB), entryLanes.get(randomLaneindex), i);
            break;
          case "recklessBus":
            Driver recklessB = new RecklessDriver(Integer.toString(i));
            addVehicle(new Bus(recklessB), entryLanes.get(randomLaneindex), i);
            break;
        }
    }

  }

}
```

```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.utils;

import java.util.Comparator;

/**
 *
 * @author balazs
 */
public class PointCWComparator implements Comparator<Point> {

  private final Point center;

  public PointCWComparator(Point center) {
    this.center = center;
  }

  private boolean less(Point a, Point b) {
    if (a.x - center.x >= 0 && b.x - center.x < 0) {
      return true;
    }
    if (a.x - center.x < 0 && b.x - center.x >= 0) {
      return false;
    }
    if (a.x - center.x == 0 && b.x - center.x == 0) {
      if (a.y - center.y >= 0 || b.y - center.y >= 0) {
        return a.y > b.y;
      }
      return b.y > a.y;
    }

    // compute the cross product of vectors (center -> a) x (center -> b)
    double det = (a.x - center.x) * (b.y - center.y) - (b.x - center.x) * (a.y -
 center.y);
    if (det < 0) {
      return true;
    }
    if (det > 0) {
      return false;
    }

    // points a and b are on the same line from the center
    // check which point is closer to the center
    double d1 = (a.x - center.x) * (a.x - center.x) + (a.y - center.y) * (a.y -
center.y);
    double d2 = (b.x - center.x) * (b.x - center.x) + (b.y - center.y) * (b.y -
center.y);
    return d1 > d2;
  }

  @Override
  public int compare(Point o1, Point o2) {
    if (less(o1, o2)) {
      return -1;
    }
    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.utils;

import java.util.List;

/**
 *
 * @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;
  }

  @Override
  public boolean equals(Object o){
    if(o == null) return false;
    if(!(o instanceof Point)) return false;
    Point p = (Point)o;
    return x==p.x && y==p.y;
  }

  @Override
  public int hashCode() {
    int hash = 7;
    hash = 59 * hash + (int) (Double.doubleToLongBits(this.x) ^ (Double.doubleTo
LongBits(this.x) >>> 32));
    hash = 59 * hash + (int) (Double.doubleToLongBits(this.y) ^ (Double.doubleTo
LongBits(this.y) >>> 32));
    return hash;
  }

  @Override
  public String toString(){
    return "("+x+","+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 Point unitVector(){
    return div(distanceFromOrigin());
  }

  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);
    }
  }


  public static Point centroid(List<Point> points){
    double x = 0.;
    double y = 0.;

    for (Point point : points) {
      x += point.getX();
      y += point.getY();
    }

    x = x/points.size();
    y = y/points.size();

    return new Point(x, y);
  }

  public  static double distanceBetweenPoints(Point x, Point y){
      return Math.sqrt(Math.pow((y.getX()-x.getX()),2) + Math.pow((y.getY()-x.ge
tX()),2));
```

```
    }
}
```

```
/*
 * 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;
  }
}
```

```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.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;
    size = new Size(20, 10);
  }

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

}
```

Car.java

```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;
    size = new Size(14, 8);
  }

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

}
```