Vehicle.java

```java
/**
 * Represent a vehicle with an id−tag (String) and a born−time
 */

public class Vehicle {
   private String id;
   private int born;

   public Vehicle(String id, int born) {
     this.id = id;
     this.born = born;
   }

   public String toString() {
     ...
   }

   public int getBorn() {
     return born;
   }
}
```

Signal.java

```java
/**
 * Represents a traffic signal.
 * The signal has three states:
 * allowing westbound vehicles to pass,
 * allowing northbound vehicles to pass and
 * allowing no one to pass
 */

public class Signal {

  private int timeWest;
  private int timeNorth;
  private int timeStop;
  private int clock;

  /**
   * Defines the signal characteristics
   * @param timeWest Number of timesteps allowing westbounds to pass
   * @param timeNorth Number of time steps allowing northbounds to pass
   * @param timeStop Number of time steps allowing no one to pass
   */
  public Signal(int timeWest, int timeNorth, int timeStop) {
    this.timeWest = timeWest;
    this.timeNorth = timeNorth;
    this.timeStop = timeStop;
    this.clock = 0;
  }

  /**
   * Advance the internal clock
   */
  public void step() {
    clock++;
    if (clock == timeWest + timeNorth + timeStop)
      clock = 0;
  }

  /**
   * Check if westbounds are allowed to pass
   * @return true if westbounds are allowed to pass, else false
   */
  public boolean greenWest() {
    ...
  }

  /**
   * Check if northbounds are allowed to pass
   * @return true if northbounds are allowed to path, else false
   */
  public boolean greenNorth() {
    ...
  }

  /**
   * @return a string indicating the status of the signal
   */
  public String toString() {
    ...
  }

}
```

```
/**
 * Represents a traffic crossing controlled by a traffic signal.
 * There are two queues in front of the signal - one for
 * westbound and one for northbound vehicles.
 * The system also represent the vehicle just passed the crossing.
 */


public class TrafficSystem {
  private Signal signal;
  private ArrayList<Vehicle> northBound;
  private ArrayList<Vehicle> westBound;
  private Vehicle passed; // Vehicle just passed the signal
  private int time;

  // Variables for statistics
  ...
  ...



  /**
   * Constructor
   * @param signal The signal to be used in the crossing
   */
  public TrafficSystem(Signal signal) {
    this.signal = signal;
  }

  /**
   * Add a vehicle to the westbound queue
   * @param v The vehicle to be added
   */
  public void addWestbound(Vehicle v) {
    ...
  }

  /**
   * Add a vehicle to the northbound queue
   * @param v The vehicle to be added
   */
  public void addNorthbound(Vehicle v) {
    ...
  }

  /**
   * Advance the system one time step.
   * Steps the signal, takes a vehicle from the queue
   * with green signal past the crossing, collects
   * data for statistics and increases the internal clock
   */
  public void step() {

    signal.step();
    passed = null;

    if (signal.greenWest()) {
      if (westBound.size()>0) {
       ...
       ...
       ...


    time++;
  }
```

```
  /**
   * Produce a snapshot of the current situation
   */
  public void print() {
    System.out.println("--------------------");

    System.out.println("     " + northBound);
    if (passed==null)
      System.out.print("  ");
    else
      System.out.print(passed + "\t");
    System.out.println(signal);

    System.out.println("     " + westBound);
  }

  /**
   * Produce statistics
   */
  public void printStatistics() {
    System.out.println("\nStatistics:\n");
    System.out.println("Number of northbound cars: " + ... );
    System.out.println("Average queue time    : " + ... );

    System.out.println();
    System.out.println("Number of westbound cars : " + ... );
    System.out.println("Average queue time    : " + ... );
  }

  /**
   * Small test program
   */
  public static void main(String[] args) throws InterruptedException {
    Signal s = new Signal(5, 3, 2);
    TrafficSystem ts = new TrafficSystem(s);
    for (int i= 0; i<10; i++) {
      Thread.sleep(10);
      if (Math.random()<0.3)
        ts.addNorthbound(new Vehicle("n", i));
      if (Math.random()<0.4)
        ts.addWestbound(new Vehicle("w", i));
      ts.step();
      ts.print();
    }
    ts.printStatistics();
  }

}
```