6/15/20

# Assignment 5 WITH EXAMPLE

Implement the next release of your term project, employing concurrency if it fits; otherwise create a different project with concurrency.

The same instructions as before apply to this completed Word document, the gray text, the 5 page limit, appendices, JUnit tests, and a ReadMe file.

## 5.1 SUMMARY DESCRIPTION

One- or two-paragraph overall description of your proposed term project. Color red the parts changed from Assignment 5 (all in red if this is a separate application).

<Your response replaces this>

This application simulates on-demand street lighting in one direction for 8 street lamps on a road. The user selects (simulated) vehicle arrival times at the first pole. The position of each vehicle is considered to be either (1) prior to this stretch of road, (2) past it, or (3) at a pole. The simulation assumes that vehicles take one second to travel from one pole to the next. The application reports the status of all vehicles approximately every half-second, showing the street light as being lit wherever a vehicle is located, as well as one pole ahead of a vehicle—otherwise off.

## 5.2 ADDITIONAL REQUIREMENTS (FEATURES) IMPLEMENTED IN THIS RELEASE

Provide the title and one or two sentences per requirement. Repeat requirements implemented for prior assignments if they are necessary to provide context. Make it clear which requirements are new vs. old.

### 5.2.1 Arrival Times (NEW)

The system prompts the user to enter vehicle arrival times.

### 5.2.2 Echo and Header ( NEW)

The system echoes the input arrival times and the following header appears on the console:

========AT CLOCK TIME 0========

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

### 5.2.3 Devices Listen (NEW)

The system starts devices listening at poles as long as vehicles are present at the pole. In that case, the current pole and the next one—if there is one—are lit.

### 5.2.4 Clock Advances (NEW)

The system advances the clock as long as vehicles are in play.

### 5.2.4 Reporting (NEW)

The system reports status of all cars and poles approximately every half second.

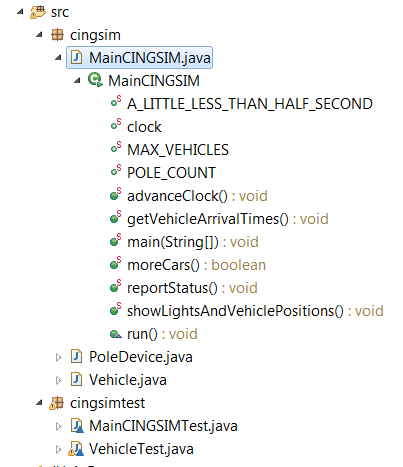
## 5.3 I/O AND FILES SUPPORTING THE REQUIREMENTS LISTED ABOVE

Provide an example of input / output showing the new features of your application.

Please see appendix 1.

## 5.4 YOUR DIRECTORY

Show a screenshot of your directory. Include all relevant files. This should include JUnit tests.



## 5.5 DESIGN

### 5.5.1 Class Model, Use Case, and Sequence Diagram

## Supply a main use case, the class model, and the sequence diagram corresponding to the use case. These should be consistent. Indicate in red where you applied the features listed below.

Please see [here](https://docs.google.com/spreadsheets/d/1tZtCTI_sFufblradHgS9u1UX3a4sFnewim_trYsdzJw/edit?usp=sharing).

### 5.5.2 Code showing where concurrency is *defined*

Location I

**public** **class** MainCINGSIM **extends** Thread {

**public** **static** **int** *clock* = 0; // seconds since start of simulation

**public** **static** **int** *A\_LITTLE\_LESS\_THAN\_HALF\_SECOND* = 495;

**public** **static** **int** *MAX\_VEHICLES* = 3;

**public** **static** **int** *POLE\_COUNT* = 8;

…

**public** **void** run() {

// Postcondition 1 (clock): As for advanceClock as long as there are vehicles

// Post2 (Last Light): light 7 is off

**while** (*moreCars*()) {

*advanceClock*();

}

PoleDevice.*theDevices*[7].lightOn = **false**; // no more cars

}

Location II: See Appendix 2

### 5.5.3 Code showing where concurrency is *used*

.

For locations I and II:

**public** **static** **void** main(String[] args) {

/\*

\* Postcondition 1: As for getVehicleArrivalTimes()

\* Post2: As for showLightsAndVehiclePositions()

\* Post3: Every PoleDevice.theDevices started

\* Post4: As for reportStatus()

\*/

*getVehicleArrivalTimes*();

System.***out***.println("========AT CLOCK TIME 0========");

*showLightsAndVehiclePositions*();

**for** (PoleDevice device : PoleDevice.*theDevices*) {

device.start();

}

(**new** MainCINGSIM()).start();

*reportStatus*();

}

## 5.6 YOUR CODE

Unless your facilitator arranges another method, copy your Eclipse project to your file system, zip it, and attach it. Please contact your facilitator in advance if you want to request an exception.

Eclipse image attached.

## 5.7 Instructor’s Evaluation



## Appendix 1: Example I/O

The “AT CLOCK TIME” statements appear approximately every half second.

Please enter up to 3 arrival times,

natural numbers, as in '2 3 5': 1 2 4

Vehicles arriving at--->1 2 4

Vehicle 0 created, arriving 1 seconds after sim begins.

Vehicle 1 created, arriving 2 seconds after sim begins.

Vehicle 2 created, arriving 4 seconds after sim begins.

========AT CLOCK TIME 0========

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F F F F F

0

1

2

==================AT APPROX. TIME 0==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F F F F F

0

1

2

==================AT APPROX. TIME 0==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F F F F F

0

1

2

==================AT APPROX. TIME 1==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

N N F F F F F F

0

1

2

==================AT APPROX. TIME 1==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

N N F F F F F F

0

1

2

==================AT APPROX. TIME 2==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

N N N F F F F F

0

1

2

==================AT APPROX. TIME 2==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

N N N F F F F F

0

1

2

==================AT APPROX. TIME 3==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F N N N F F F F

0

1

2

==================AT APPROX. TIME 3==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F N N N F F F F

0

1

2

==================AT APPROX. TIME 4==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

N N N N N F F F

0

1

2

==================AT APPROX. TIME 4==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

N N N N N F F F

0

1

2

==================AT APPROX. TIME 5==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F N N N N N F F

0

1

2

==================AT APPROX. TIME 5==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F N N N N N F F

0

1

2

==================AT APPROX. TIME 6==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F N N N N N F

0

1

2

==================AT APPROX. TIME 6==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F N N N N N F

0

1

2

==================AT APPROX. TIME 7==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F N N N N N

0

1

2

==================AT APPROX. TIME 7==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F N N N N N

0

1

2

==================AT APPROX. TIME 8==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F N N N N

0

1

2

==================AT APPROX. TIME 8==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F N N N N

0

1

2

==================AT APPROX. TIME 9==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F F N N N

0

1

2

==================AT APPROX. TIME 9==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F F N N N

0

1

2

==================AT APPROX. TIME 10==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F F F N N

0

1

2

==================AT APPROX. TIME 10==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F F F N N

0

1

2

==================AT APPROX. TIME 11==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F F F F N

0

1

2

==================AT APPROX. TIME 11==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F F F F N

0

1

2

==================AT APPROX. TIME 12==================

'T' = to arrive; 'G' = gone; 'N' = light on; 'F' = light off

T--0--1--2--3--4--5--6--7--G

F F F F F F F F

0

1

2

## Appendix 2 **class** PoleDevice **extends** Thread

// Devices on poles that switch lights on or off according to cars present

….

**public** **class** PoleDevice **extends** Thread {

….

**public** **void** run() {

/\*

\* Postconditions: Every MILLISECONDS\_BETWEEN\_SENSING milliseconds ...

\*

\* Postcondition 1 (Pole 0): EITHER

\* there is a car at pole 0 AND lights 0 and 1 are on

\* OR

\* (there is no car at 0 and there is a car at 1) AND (light 0 is off and 1 is on)

\* OR

\* (there is no car at 0 or 1) AND (lights 0 and 1 are off)

\*

\* Post2 (Poles 1-6): Poles 1-6 lights are on/off 0 or 1 poles ahead

\* depending on whether there is a car at the light or one ahead

\*/

**while** (MainCINGSIM.*moreCars*()) {

// Post1 (Pole 0):

**if** (poleNumber == 0 && vehicleAt(0)) { //turn on

lightOn = **true**; // here

*theDevices*[1].lightOn = **true**;

}

**if** (poleNumber == 0 && !vehicleAt(0) && vehicleAt(1)) {

lightOn = **false**; // here

*theDevices*[1].lightOn = **true**; // one ahead

}

**if** (poleNumber == 0 && !vehicleAt(0) && !vehicleAt(1)) {

lightOn = **false**; // here

*theDevices*[1].lightOn = **false**; // one ahead

}

// Post2 (Poles 1-6):

**if** (poleNumber > 0 && poleNumber < 7 && vehicleAt(poleNumber)) { // vehicle is here

lightOn = **true**; // here

*theDevices*[poleNumber + 1].lightOn = **true**; // one ahead

}

// No vehicle (turn off one ahead but not my own light necessarily)

**if** (poleNumber > 0 && poleNumber < 7 && !vehicleAt(poleNumber) &&

!vehicleAt(poleNumber + 1)) {

*theDevices*[poleNumber + 1].lightOn = **false**;

}

// Every half second

**try** {

Thread.*sleep*(MILLISECONDS\_BETWEEN\_SENSING);

}

**catch** (InterruptedException e) {

e.printStackTrace();

}

}

}