Traffic light controller

Danila Prymak

Main Idea:

The controller gets both implicit(clock which is used to measure the light cycle) and explicit(signals from inductive loops about awaiting or passing by vehicles) input to make a decision which output it's going to be (red (no go), red&yellow (ready), green (go), yellow (stop)).

The time for `Red and Green` lights the same (`20 sec.`) because while `RY` and `Y` cars cannot move.

Time for car to pass the light(means it won't influence the inductive loop) is `2 sec.`

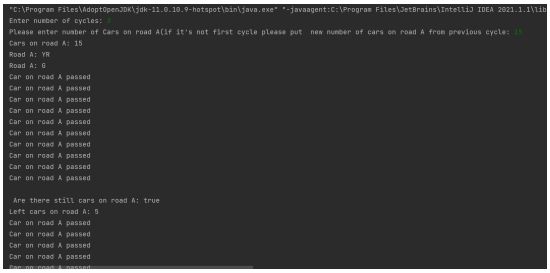
Assumptions because I had problems to implement it in more complex way:

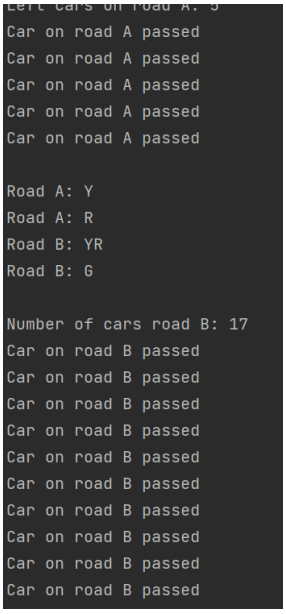
- For one cycle of Green 10 cars can pass due to chosen time limits(20 sec of green / 2 sec one car passes = 10 cars)
- User should enter number of cycles of simulating road traffic
- User should enter number of cars on road A before every cycle(number of cars on road B is always randomised)

> 💡 Attention: if it is not the first cycle of simulation user should put number of cars on road A from previous cycle(it will be output in the and of the cycle named "new number of cars on road A"

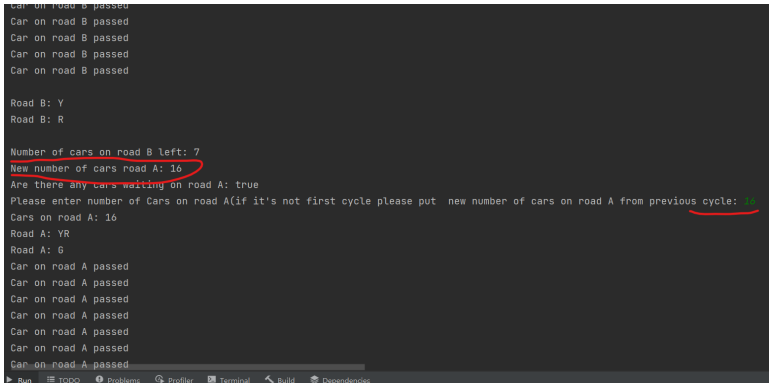
1. Report:

How code works





Here we see what number we need to put on next cycle



- what was to be difficult to accomplish?
  - repeating the loop of simulating the road traffic, here I have clumsy implementation by user inputing number of cars on road A every time
- where did you fail?
- there was a problem when I needed to simulate the number of cars on road A while the previous cycle is not ended(while loop was used)
  and I solved it by letting the user to input it before every cycle of simulation
- what makes your solution special? what deserves appreciation?
  - The solution itself isn't really interesting and it's pretty simple because i did it in only functional way, but I think solving the problem with number on road A at the end of cycle worth appreciation.

```
package trafficController;

import java.util.Scanner;
import java.util.concurrent.ThreadLocalRandom;


public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        //User decides how many cycles of moving  program will simulate
        System.out.print("Enter number of cycles: ");
        int numberOfCycles = Integer.valueOf(scanner.nextLine());

        while (numberOfCycles != 0) {
            System.out.print("Please enter number of Cars on road A(if it's not first cycle please put " +
                    " new number of cars on road A from previous cycle: ");
            int numberOfCarsA = Integer.valueOf(scanner.nextLine());
            driver(numberOfCarsA, numberOfCarsB());
            numberOfCycles--;
        }
    }

    private static void driver(int numberOfCarsA, int numberOfCarsB) {
        /*
        Param:
            int numberOfCarsA - input from user and if second and more cycle you need to put new number from previous cylce
            int numberOfCarsB - random number from method numberOfCarsB()
        Usage:
            do one cycle of using both A and B roads, using 4 priority rules and stops after
            cycle of road B
            if there is no cars on road A still uses rule for and grand another cycle for road B
         */

        System.out.println("Cars on road A: " + numberOfCarsA);

        System.out.println("Road A: YR");
        System.out.println("Road A: G");
        //priority rule 1
```

```
        for (int i = 20; i != 0; i -= 2) {
            if (!inductiveLoops(numberOfCarsA)) {
                break;
            }
            System.out.printf("Car on road A passed \n");
            numberOfCarsA--;
        }
        System.out.println("\n Are there still cars on road A: " + inductiveLoops(numberOfCarsA));
        System.out.println("Left cars on road A: " + numberOfCarsA);

        //priority rule 2
        if (inductiveLoops(numberOfCarsA)) {
            while (numberOfCarsA != 0) {
                System.out.println("Car on road A passed");
                numberOfCarsA--;
            }
        }

        System.out.println("\nRoad A: Y");
        System.out.println("Road A: R");
        System.out.println("Road B: YR");
        System.out.println("Road B: G\n");

        System.out.println("Number of cars road B: " + numberOfCarsB);

        //priority rule 3
        for (int i = 20; i != 0; i -= 2) {
            if (!inductiveLoops(numberOfCarsB)) {
                break;
            }
            System.out.println("Car on road B passed ");
            numberOfCarsB--;
        }

        System.out.println("\nRoad B: Y");
        System.out.println("Road B: R\n");

        int newNumberOfCarsA = numberOfCarsA();
        System.out.println("Number of cars on road B left: " + numberOfCarsB);
        System.out.println("New number of cars road A: " + newNumberOfCarsA);
        System.out.println("Are there any cars waiting on road A: " + inductiveLoops(newNumberOfCarsA));

        //priority rule 4
        if (!inductiveLoops(newNumberOfCarsA)) {
            System.out.println("Road B: YR");
            System.out.println("Road B: G");
            while (numberOfCarsB != 0) {
                System.out.println("Car on road B passed");
                numberOfCarsB--;
            }
            System.out.println("Road B: Y");
            System.out.println("Road B: R");
        }
    }

    private static boolean inductiveLoops(int numberOfCars) {
        /*
        return true if there are cars staying on the traffic light
        false if there are no cars
         */
        if (numberOfCars > 0) {
            return true;
        }
        return false;
    }

    //rendomising number of cars on the road not more than 15
    private static int numberOfCarsA() {
        int numberOfCarsA = ThreadLocalRandom.current().nextInt(0, 20 + 1);
        return numberOfCarsA;
    }

    private static int numberOfCarsB() {
        int numberOfCarsB = ThreadLocalRandom.current().nextInt(0, 20 + 1);
        return numberOfCarsB;
    }
}
```