Assignment 2 – Implementing a Car Showroom using Multithreading

- *Please create a separate file for each of your Java classes.*
- *Please copy the output from a run of your completed code into a .txt file and submit this also.*
- *Put all files into a single zip-file named "Assignment2-YourName.zip" and submit via Blackboard.*
- *Deadline (strict): Wednesday, 19th February, 23:59. Late submission only with a medical certificate.*
- *Please note that the graders are advised to check all submissions for plagiarism. Assignments need to be solved individually (no group work).*
- *If you have questions about the assignment or need help with it, the best approach is to ask a tutor during one of the lab sessions (it is recommended to visit all lab sessions).*
- *Use Java comments to explain your code. Marks will be awarded for detailed comments **(10%)** and clear legible code style **(10%)**.*

In this assignment you will implement a simulation of car dealership in Java. This simulation will be an example of the Producer-Consumer model (see Lecture 3b), where there are multiple producers (car sellers) and consumers (car buyers). Each producer and consumer will run on a separate thread, and there can be multiple producer and consumer threads active in the application at any one time. You should create 5 .java files in total, as detailed below: Car.java, CarShowroom.java, Buyer.java, Seller.java and Main.java.

Part 1: Car.java (10%)

Create a class `Car`. This class should have private fields for the following details: registration, sale value and colour. When the constructor of this class is called, it should create a new instance of the `Car` class that has randomly generated values for these details.

Registrations should be stored as strings and have a format like the following "00-G-2137". Sale values can be rounded to the nearest euro and the generated values should be between €1,000 and €20,000. Colours should stored as strings and should be chosen randomly from one of a finite list of options (hint: create a static final array listing the colours, then select one of these at random for each new `Car` object that is created).

It may be helpful to override the `toString()` method, so that the details of a `Car` may be easily printed to the console as per the code sample below:

```
// create a new Car object with a randomly chosen registration, colour and sale value
Car c = new Car();
// prints e.g. "yellow car with registration 15-G-5123 worth €18198"
System.out.println(c.toString());
```

Part 2: CarShowroom.java (10%)

Create a class `CarShowroom`. This class should have a private field that specifies its capacity (i.e. the maximum number of cars the showroom can hold at once). Try using a capacity of 5 or 10 cars. It should also have a private field that holds a list of cars (using e.g. an `ArrayList`).

`CarShowroom` should have a method `isEmpty()` that returns a `boolean` indicating whether the showroom is currently empty, and a method `isFull()` that returns a `boolean` indicating whether the showroom is currently full.

`CarShowroom` should also have an `addCar()` method to add a new `Car` to the list, and a `takeCar()` method to remove the first (oldest) `Car` in the list and return it to the caller. You may assume that a Buyer will always buy the first available `Car` in the `ArrayList`.

Part 3: Buyer.java (20%)

This class will represent customers who wish to buy cars from the `CarShowroom`. Each instance of `Buyer` will run as a separate thread, and so the `Buyer` class should implement the `Runnable` interface.

Each `Buyer` instance should be assigned a unique ID when it first tries to interact with the `CarShowroom` (i.e. when its `run()` method is called). This unique ID should be stored in a private integer field. Static thread safe fields (e.g. instances of `AtomicInteger`) should be used to count the total number of buyers created, and the total number of purchases successfully completed.

The `Buyer` constructor should accept an instance of `CarShowroom` as an argument, and then store the reference to this object in a private field.

The `Buyer` `run()` method should attempt to buy a `Car` from the showroom. It should first check that there is at least one Car available to buy in the showroom (by calling the `isEmpty()` method of the `CarShowroom` instance). If there are no cars available, the `Buyer` should wait until a car becomes available before proceeding. The `Buyer` should then proceed to take a car from the `CarShowroom` using the `takeCar()` method. Once a `Buyer` has successfully completed a purchase, its `run()` method can terminate (i.e. the thread has finished its work). Some of the operations in the `run()` method will need to be made be made thread safe to prevent race conditions (e.g. two buyers trying to buy the same `Car`). Your answer should use the `wait()` and `notify()/notifyAll()` methods.

A message should be printed to the console each time a new `Buyer` reaches the dealership (directly after its `run()` method is called), each time a `Buyer` waits to buy a `Car`, and each time a `Buyer` successfully buys a `Car` from the `CarShowroom` (see example program output on the last page).

Part 4: Seller.java (20%)

This class will represent customers who wish to sell cars to the showroom. Each instance of `Seller` will run as a separate thread, and so the `Seller` class should implement the `Runnable` interface.

Each `Seller` instance should be assigned a unique ID when it first tries to interact with the `CarShowroom` (i.e. when its `run()` method is called). Static thread safe fields (e.g. instances of `AtomicInteger`) should be used to count the total number of sellers created, and the total number of sales successfully completed.

The `Seller` constructor should accept an instance of `CarShowroom` as an argument, and then store the reference to this object in a private field.

The `Seller` `run()` method should attempt to buy a `Car` from the showroom. It should first check that the `CarShowroom` has space available for another `Car` (by calling the `.isFull()` method of the `CarShowroom` instance). If there is no space available, the `Seller` should wait until a space becomes available before proceeding. The Seller should then proceed to add its `Car` instance to the `CarShowroom`. Once a `Seller` has successfully completed a sale, its `run()` method can terminate (i.e. the thread has finished its work). Some of the operations in the `run()` method should be made thread safe to prevent race conditions (e.g. two sellers trying to sell cars at the same time and causing the `CarShowroom` capacity to be exceeded). Your answer should use the `wait()` and `notify()/notifyAll()` methods.

A message should be printed to the console each time a new `Seller` reaches the dealership (directly after its `run()` method is called), each time a `Seller` waits to sell a `Car`, and each time a `Seller` successfully sells a `Car` to the `CarShowroom` (see example program output on the last page).

Part 5: Main.java (20%)

This file will contain the `main()` method that will run your car dealership simulation. It should create a single instance of the `CarShowroom` class; this instance should be passed into the constructors of any `Buyer` or `Seller` objects that are created.

Your main method should simulate a number of days (e.g. 30) at the car dealership (hint: use e.g. a while loop and repeat the same operations for each day). A message should print to the console announcing the number of cars available at the start of each day (see example program output on the last page).

Each day a random number of `Buyers` (between 0 and e.g. 3) and `Sellers` (between 0 and e.g. 3) can arrive at the dealership. For each new `Buyer`/`Seller` a new thread must be created and started. You should also add a small delay at the end of each day (loop iteration) to slow down the execution of the program.

Bonus harder features (these do not need to be implemented to be eligible for full marks)

- Try limiting the purchasing power of `Buyers`, i.e. give them a random amount of money to spend. If there are no `Cars` within a `Buyer`'s budget, they would have to wait until an affordable `Car` shows up on the sales floor.
- Try implementing randomly generated preferences for each `Buyer` that is created e.g. a preference for a certain colour of `Car`, or for `Cars` registered after a certain year. `Buyers` would then choose to buy only a `Car` that matches their preference (and perhaps wait until an appropriate `Car` shows up in the `CarShowroom`)
- Adda markup for the dealership so that the sale price to `Buyers` is e.g. 20% higher than the price paid to `Sellers`

Sample (truncated) program output:

```
Day 1 beginning. There are 0 cars in the showroom today.
A new buyer #1 just appeared.
Buyer #1 wants to buy a car, but the showroom is empty.
A new seller #1 just appeared with a yellow car with registration 11-G-7212 worth €3944
to sell.
A new seller #3 just appeared with a black car with registration 18-G-2743 worth €14040
to sell.
Seller #1 sold their yellow car with registration 11-G-7212 worth €3944 to the showroom.
This is sale number 1.
A new seller #2 just appeared with a silver car with registration 18-G-2469 worth €1449
to sell.
Buyer #1 bought a yellow car with registration 11-G-7212 worth €3944 from the showroom.
This is purchase number 1.
Seller #3 sold their black car with registration 18-G-2743 worth €14040 to the showroom.
This is sale number 2.
Seller #2 sold their silver car with registration 18-G-2469 worth €1449 to the showroom.
This is sale number 3.
Day 2 beginning. There are 2 cars in the showroom today.
A new seller #4 just appeared with a black car with registration 18-G-5011 worth €10859
to sell.
```

```
A new seller #5 just appeared with a black car with registration 15-G-8043 worth €11984
to sell.
Seller #4 sold their black car with registration 18-G-5011 worth €10859 to the showroom.
This is sale number 4.
Seller #5 sold their black car with registration 15-G-8043 worth €11984 to the showroom.
This is sale number 5.
Day 3 beginning. There are 4 cars in the showroom today.
A new seller #6 just appeared with a green car with registration 18-G-756 worth €19839
to sell.
A new seller #7 just appeared with a black car with registration 16-G-2453 worth €18573
to sell.
Seller #6 sold their green car with registration 18-G-756 worth €19839 to the showroom.
This is sale number 6.
Seller #7 is trying to sell a car, but the showroom is full.
Day 4 beginning. There are 5 cars in the showroom today.
A new buyer #2 just appeared.
A new seller #8 just appeared with a green car with registration 10-G-5572 worth €11462
to sell.
Buyer #2 bought a black car with registration 18-G-2743 worth €14040 from the showroom.
This is purchase number 2.
Seller #7 sold their black car with registration 16-G-2453 worth €18573 to the showroom.
This is sale number 7.
Seller #8 is trying to sell a car, but the showroom is full.
Day 5 beginning. There are 5 cars in the showroom today.
A new buyer #3 just appeared.
A new seller #9 just appeared with a green car with registration 17-G-2593 worth €1501
to sell.
Buyer #3 bought a silver car with registration 18-G-2469 worth €1449 from the showroom.
This is purchase number 3.
Seller #8 sold their green car with registration 10-G-5572 worth €11462 to the showroom.
This is sale number 8.
Seller #9 is trying to sell a car, but the showroom is full.
Day 6 beginning. There are 5 cars in the showroom today.
A new buyer #4 just appeared.
A new buyer #5 just appeared.
A new seller #10 just appeared with a silver car with registration 12-G-6077 worth €1122
to sell.
Buyer #4 bought a black car with registration 18-G-5011 worth €10859 from the showroom.
This is purchase number 4.
Seller #9 sold their green car with registration 17-G-2593 worth €1501 to the showroom.
This is sale number 9.
A new seller #12 just appeared with a black car with registration 17-G-91 worth €1457 to
sell.
A new seller #11 just appeared with a silver car with registration 20-G-6505 worth
€16467 to sell.
Seller #10 is trying to sell a car, but the showroom is full.
Buyer #5 bought a black car with registration 15-G-8043 worth €11984 from the showroom.
This is purchase number 5.
Seller #10 sold their silver car with registration 12-G-6077 worth €1122 to the
showroom. This is sale number 10.
Seller #11 is trying to sell a car, but the showroom is full.
Seller #12 is trying to sell a car, but the showroom is full.
Day 7 beginning. There are 5 cars in the showroom today.
Day 8 beginning. There are 5 cars in the showroom today.
A new buyer #6 just appeared.
```

```
A new buyer #7 just appeared.
A new seller #13 just appeared with a black car with registration 12-G-8021 worth €5192
to sell.
Buyer #6 bought a green car with registration 18-G-756 worth €19839 from the showroom.
This is purchase number 6.
Seller #12 sold their black car with registration 17-G-91 worth €1457 to the showroom.
This is sale number 11.
Seller #11 is trying to sell a car, but the showroom is full.
Seller #13 is trying to sell a car, but the showroom is full.
Buyer #7 bought a black car with registration 16-G-2453 worth €18573 from the showroom.
This is purchase number 7.
Seller #13 sold their black car with registration 12-G-8021 worth €5192 to the showroom.
This is sale number 12.
Seller #11 is trying to sell a car, but the showroom is full.
Day 9 beginning. There are 5 cars in the showroom today.
Day 10 beginning. There are 5 cars in the showroom today.
A new buyer #8 just appeared.
Buyer #8 bought a green car with registration 10-G-5572 worth €11462 from the showroom.
This is purchase number 8.
A new buyer #9 just appeared.
A new seller #15 just appeared with a green car with registration 20-G-4476 worth €13058
to sell.
A new seller #14 just appeared with a yellow car with registration 11-G-8395 worth €8989
to sell.
Seller #11 sold their silver car with registration 20-G-6505 worth €16467 to the
showroom. This is sale number 13.
A new seller #16 just appeared with a yellow car with registration 10-G-5035 worth €8286
to sell.
Seller #14 is trying to sell a car, but the showroom is full.
Seller #15 is trying to sell a car, but the showroom is full.
Buyer #9 bought a green car with registration 17-G-2593 worth €1501 from the showroom.
This is purchase number 9.
Seller #15 sold their green car with registration 20-G-4476 worth €13058 to the
showroom. This is sale number 14.
Seller #14 is trying to sell a car, but the showroom is full.
Seller #16 is trying to sell a car, but the showroom is full.
```