Car (Part 2) - Arrays

Description

Using your code from the first Car assignment, redesign the program such that ten cars can be controlled instead of just one. All the methods from before will remain the same, in terms of functionality. Additionally, we are still dealing with a 20x20 grid that has strict bounds. All cars have the same attributes as the single car did in the first Car assignment. Also, the error checking from before should still be in place.
As a friendly reminder: do not use static arrays.

Design and Implementation

In order to create ten cars that can be controlled by the user, we will need to use arrays. There will be an array for each attribute that a car should have. For instance:

```
boolean[] ignitions = new boolean[10];
```

In essence, each car will have an associated index in every array. As an example, car 3 would correspond to the third element in every array. If we needed to get its ignition we would write “ignitions[2]”. Accessing other attributes would mean getting whatever was stored in the third elements of other arrays. Keep in mind that the car number will not be the index number exactly, but will be off by one (e.g., car 3 has index 2 in each array).

We will know which index to use because the user will supply it. Before the main menu is presented, the user will select a car. After the car is selected the main menu will appear. All of the options available to the user will be the same as before, but any changes made will affect *only* the car that the user has selected. Take care to ensure that changes to one car do not affect any other cars.

You may change the method signatures so that they will accept arrays, or even modify elements of those arrays directly. If you choose to go with this approach, keep in mind that it will probably not be necessary for some methods to return something. You can also choose to keep the methods’ return types as they are, meaning that you would use the return values in some meaningful way. For example, the methods would return values to the array elements in main() (or in a separate helper method).

In summary about methods: You can pass in the arrays and directly modify their values. You can make the methods return something or return nothing (i.e., void). Otherwise, if you are modifying the array elements *outside* of a method, then the method will certainly need a return type of some kind.

After the state is reported for the currently selected car, begin the process again by asking the user to select a car. The user could select the same car or a different one. No error checking is needed for getting the input for the car number.

Regarding the state reporting: it is not necessary to print out where every car is on the grid. You can let the method report on just one car at a time. The only change you will need to make is to print the car number.

[remainder of page intentionally left blank]

Example output

```
Which car would you like to use? (Choose from 1-10)
Input: 3

What would you like to do?
1: Turn the ignition on/off
2: Change the position of the car
Q: Quit this program
Input: 1

Car Information
Car#: 3       // the car number should be printed
Color: Green
Ignition: On
Location: (4, 6)

--------------------
--------------------
--------------------
-----G--------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
```

// continue execution until user decides to quit

Grading Rubric

Points	Criteria
3	Creates ten cars through the use of arrays, successfully allows the user to choose different cars
2.5	Changes car attributes individually (a change to one car does not affect a change to another car)
2.5	Maintains same functionality as the original car assignment
2	Good style demonstrated in the code; sensible formatting