# Car (Part 1)

## Description

This program will create a virtual car that can then be controlled by the user through a menu. The car will exist not as a single object, but rather as a group of different variables.

The car should have an ignition, a color, and a position (given by an x coordinate and a y coordinate). The ignition state and the position state may change during the course of the program's execution, but the color will stay the same. Before the user is given control of the car via the menu, the car's attributes should already be assigned (i.e., it will have its ignition set to "off", it will be given a color, and it will have random position coordinates).

The user will be able to turn the car on or off, and they will also be able to move the car around on a 20x20 grid. The car's current location will be represented by a char that stands for its color (e.g., "R" for red).

Specific error checking will be expected. The car should be prevented from going out of bounds, for instance. Also, if the ignition is off then the car clearly should not be able to move anywhere.

Lastly, the most current grid and the status of the car should be printed after each user action (i.e., turning the ignition on/off, legal movement of the car, and when the user chooses to quit).

## Design and Implementation

The car's three attributes are its ignition, color, and position (represented by an x coordinate and a y coordinate). Instead of simply assigning values by choosing one arbitrarily ahead of time, we will use methods. In fact, methods will be used to do a lot of the work in this assignment. Note that the method headers are not complete, since some methods may require parameters while others will not. The types of the methods should not be changed (so the assignColor() method should return a char, not a String).

Important note: the methods with return types need to return values to something. If the methods return values to nothing, then the car's ignition and position will never be changed. If you pass in the car's current ignition status, make sure that the changeIgnition() method returns a boolean value to the ignition variable, like so:

```
ignition = ignitionSwitch(ignition);
```

## Methods to implement

`public static int randomizePosition`
*Generate a random number between 1 and 20 and then return it.*

`public static char assignColor`
*Return one of the following colors: R, G, B, W, or S. The selection should be random.*

`public static boolean ignitionSwitch`
*This method should take as its parameter the car's current ignition switch and then it should return the opposite of that ignition's value. If the car is off, then this method will return true.*

`public static int moveHorizontally`
*This method requires the car's x coordinate, the user's inputted movement distance, and the ignition state of the car.*
*If the ignition is off, then the method should notify the user as such and just return the current, unchanged x coordinate. If the ignition is on, then the method should check to see if the movement is in bounds. If it is, then change the x coordinate and return its value. If it is not in bounds, then notify the user as such and return the unchanged x coordinate.*

`public static int moveVertically`
*This method requires the car's y coordinate, the user's inputted movement distance, and the ignition state of the car.*

*If the ignition is off, then the method should notify the user as such and just return the current, unchanged y coordinate. If the ignition is on, then the method should check to see if the movement is in bounds. If it is, then change the y coordinate and return its value. If it is not in bounds, then notify the user as such and return the unchanged y coordinate.*

`public static void reportState`
*This method will need all of the car's parameters in order to display the car's current status and generate the grid correctly (that is, with the car in its position). The status of the car should be properly formatted, so the ignition should not say "true" or "false" but rather "on" or "off". Similarly with the color of the car, the full word should be spelled out ("Red", "Green", "Black", "White", or "Silver"). Lastly, the position of the car should be given as an (X, Y) coordinate pair.*

*In order to make the grid, consider how loops might be used to iterate through the different rows and columns of the grid. A "-" would be printed for each spot except for the one where the car is currently location.*

Now that the methods have been specified, let's discuss the main() method. First, the car should be "created" through assigning values to its attributes. The methods that you made will accomplish this task.

Afterwards, the program will present the user with a menu options (including one to quit the program; think back on the Calculator assignment and how you made a program that can run indefinitely before being stopped by the user). Depending on the option that is selected, different methods will be called. Remember that the car's variables should be changed within main(), *not* within the different methods.

As for the movement option, the secondary menu should be generated within the main() method - *not* in the moveHorizontally() or moveVertically() methods. Once the user makes a valid choice and enters a movement distance, one of the two methods may be called.

As for error checking, the program should be able to handle invalid input at the main menu (e.g., the user inputs "zzz") and print a message to let the user know that what they just entered cannot be accepted. Similar error checking should occur at the movement menu, when the user inputs either "H" or "V". You may assume that the movement distance will always be an integer, *but you cannot assume that the movement will not potentially move the car out of bounds* (recall that separate error checking is performed in the two movement methods).

*[remainder of page blank]*

**Example Output**

*Turn on the ignition first*

```
  What would you like to do?
  1: turn the ignition on/off
  2: change the position of car
  Q: quit this program
  Input: 1
```

*Print the current grid since a user action was performed*

```
  Car Information
  Color: Red
  Ignition: On
  Location: (8, 18)

--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
-------R------------
--------------------
--------------------
```

*Move the car*

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


  In which direction do you want to move the car?
  H: Horizontal
  V: Vertical
  Direction: H


  Enter a movement distance: 2
```

*Print the grid again*

```
  Car Information
  Color: Red
  Ignition: On
  Location: (10, 18)

--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
---------R----------
--------------------
--------------------
```

**Formatting and Style**

The car status and grid display should be clear and informative, as they are in the example output. Also, the various menus should have brief but descriptive options that allow for the user to easily choose what they want to do.

**Grading Rubric**

| Points | Criteria |
|---|---|
| 5 | Creates car's variables through the use of methods; properly maintains the car's attributes and makes use of methods to change them |
| 2 | Correct printing of car's status and grid |
| 2 | Catches invalid input at each menu |
| 1 | Good style demonstrated in the code; sensible formatting |