Write a program that creates Car objects.

Your Car class should store the following qualities of a car:

- **Color** ('R'=red; 'G'=green; 'B'=blue; 'W'=white; 'S'=Silver)
- **Ignition** (false=off; true=on)
- **Position** (any X, Y coordinate in a 20 X 20 grid)

main() will work much the same way as in previous assignments. It should create an array of 10 Car instances with random color, ignition set to off/false, and position randomly assigned. Once you assign the beginning state to the cars you should have a loop which asks the user what she wants to do next. For example:

Which car would you like to use next (0-9)?

Then ask the user what she wants to do to the car she selected. For example:

What would you like to do next (1 - change ignition; 2 - change position of car; 3 - quit this program)?

If the user wants to change the car's ignition status it will call the car's instance method which changes the ignition state (explained below).

If the user wants to change the car's position, main() will prompt the user for a direction (e.g. What direction would you like to move the car (1 - horizontal; 2 - vertical)?) and the distance (e.g. How far (negative value to move left)?; How far (negative value to move up)?)). Then the appropriate method (move horizontal or move vertical -- see below) would be called.

You must use care to use the correct car's variables when calling the methods.

Each of the qualities will be represented by an instance variable in the Car class. You should write an instance method for each of the following:

- **color assignment**: Randomly picks one of the five colors (represented by a letter) and returns the corresponding char to main(). This method should be called once from the Car class’ constructor. The color will not change during the execution of the program.

- **move car horizontally**: This method will need the number of spaces you would like to move the car (a negative value will move the car left; positive value will move the car right) and will adjust the xPosition of the car.

  If the car's ignition is not on, this method tells the user that s/he must turn the ignition on first and leaves the xPosition unchanged.

  If the user tries to move the car beyond the border of the 20 X 20 grid, this method should leaves the xPosition unchanged; instead you should report an error message.
- **move car vertically**: This method will need the number of spaces you would like to move the car (a negative value will move the car up; positive value will move the car down) and will adjust the yPosition of the car.

  If the car’s ignition is not on, this method tells the user that s/he must turn the ignition on first and leaves the yPosition unchanged.

  If the user tries to move the car beyond the border of the 20 X 20 grid, this method should leave the yPosition unchanged; instead you should report an error message.

- **change ignition status**: This method will turn the car off when it is on or on when it is off. (i.e. change the value of the instance variable).

- **get color**: This method will return the color of the car as string.

- **get ignition status**: This method will return the ignition status of the car as a boolean.

- **get x position**: This method will return the x position of the car as an int.

- **get y position**: This method will return the y position of the car as an int.

- **toString**: After each action, you should print the "state" of the currently modified Car. This method will call the “get” methods and report the status of each (including a copy of the cars position in the grid). For example:

```
Car Stats:
  Color: Silver
  Ignition: Off
  Location: 14,9
```

```
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
S---------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
          --------------------
```

You only have to print one car at a time. It might be helpful to print the initial state of the grid as well so the user knows what he or she is doing.

Honors addendum:

Print all the cars on the grid at the same time, and none in the same position. This means you must check before initializing the car's position as well as when the user changes it that no other car is currently occupying that spot. Use good error checking style. The user should still only be able to manipulate one car at a time, but you can do an abridged Car Stats report to save space. For example:

Car Stats:
0   Red   On   5,17
1   Silver Off  14,9
2   Green On   3,1

.