We've 2 objects: **Car** and **ParkingLot**.

Object **Car** has 2 properties :
1. Number (Registration Number of Car)
2. Color (Color of Car)

Object **ParkingLot** has 4 properties :
1. **Max Slots** - Number of parking slots a parking lot is made up of.
2. **Parking Slots** - Array which has details of every parking slot.
3. **Empty Slots** - Array with Number of slots which isn't occupied.
4. **Occupied Slots** - Array with Number of slots which are occupied.

Public Methods for **ParkingLot** :

1. **getParkingStatus** : it is to display the parking status of the entire parking slot in tabular format.so we just iterate through the Array parkingSlots and check if Car object is available at position or not, if Car is available we can get the Number and Color properties of the car and push it to array else we just push '-' (dash) to represent empty space at a particular spot.

2. **getAllEmptySlots** : in this method we just display empty slots where the car is not parked, so we just iterate through the emptySlots array and display the same in the console.

3. **getAllOccupiedSlots** : if parking is not empty then in this method we just display empty slots where the car is not parked, so we just iterate through emptySlots array and display the same in the console otherwise it'll simply say that "Parking is Empty".

4. **parkCar** : if parking is not full then only this method will try to locate nearest slot from entry point and, we'll park the car at the nearest slot available, it'll create car object by taking carNumber and carColor as an argument and we'll create a new Car object and we'll store it at the nearest position identified in the array as well. we'll remove this particular slot from the emptySlots array and push it to the occupiedSlots array.

5. **parkCarAtGivenSlot** : if given slot is not already occupied then only this method will take slot from entry point and, we'll park the car at the given slot, it'll create car object by taking carNumber and carColor as an argument and we'll create a new Car object and we'll store it at the position identified in the array as well. we'll remove this particular slot from the emptySlots array and push it to the occupiedSlots array.

6. **getAllCarsWithSameColor** : this method will query the parkingLot and will check with a given color if the car of provided color is available in the parking lot or not, if so then we'll display it in the same tabular pattern. So in this method we'll again iterate through parkingSlots but only where the car is parked (data is available in occupiedSlots array) and check if color matches or not, if there are multiple cars available then it'll display all the cars along with where it's parked.

7. **getSlotByCarNumber** : in this method we'll take the carNumber and we'll search for the particular car from parkingSlots array and if found then we'll respond with it's position inside array and of course, if parking is not empty and if not found, then we'll say "Can't find a car with `givenNumber`".

8. **takeCarByNumber** : in this method we'll take the carNumber and we'll search for the particular car from parkingSlots array and if found then we'll store null with it's position inside array and of course, if parking is not empty and if not found, then we'll say "Can't find a car with `givenNumber`" and again we've to update occupiedSlots and emptySlots to make it available.

9. **takeCarBySlotNumber** : in this method similarly like above method "takeCarByNumber" we'll check whether a car is present at a given slot or not and again it includes updating emptySlots and occupiedSlots arrays.

Private Methods for **ParkingLot** :

1. **isFull** : we'll compare the length of occupiedSlots and maxSlots arrays.
2. **isEmpty** : we'll compare the length of emptySlots and maxSlots arrays.
3. **findNearestParkingSlot** : we'll identify the minimum number of slots available from the emptySlots array.