Design a Parking lot

Requirements:

1. User must be able to get a token while entering the lot
2. Find an empty spot in the lot
3. Park the Vehicle
4. Insert token and pay using credit card while exiting the lot

---

```
Class ParkingLotTokenMachine
Data: List<ParkingSpots> parkingSpots, List<ParkingSpots> availableSpots, Boolean doorStatus
Behavior: updateSpotAvailibility(spotId, state) {
        If (state == TRUE) {
            parkingSpots[spotId].IsAvailable = TRUE;
        } else {
            parkingSpots[spotId].IsAvailable = FALSE;
        }
    }
Behavior: dispenseToken() {
        foreach parkingSpot in parkingSpots {
            If (parkingSpots.IsAvailable == TRUE) {
                availableSpots.add(parkingSpot);
            }
        }
        If (availableSpots != NULL) {
            // vend out token
            this.DoorStatus = OPEN;
            // wait for Customer Vehicle to Pass
            this.DoorStatus = CLOSE;
        } else {
            System.Out.Println("Sorry Parking lot is Full")
        }
    }
Behavior: collectPayment(customerCreditCard) {
        CreditCard.isCardValid(customerCreditCard);
        this.doorStatus = OPEN;
        // wait for Customer Vehicle to Pass
        this.doorStatus = CLOSE;

    }
```

```
Class Customer
Data: Vehicle myVehicle, CreditCard myCreditCard etc
Behavior: getToken() {
                ParkingLotTokenMachine.DispenseToken();
                // collect and save the token safely.
        }
Behavior: parkTheVehicle() {
                // Check the available spots on display
                myVehicle.drive(to nearest open parking spot);
                if (ParkingSpot.GetDimensions() == Vehicle.Dimensions) {
                        myVehicle.park();
                } else {
                        myVehicle.drive(to next spot which fits the vehicle);
                        myVehicle.park();
                }
        }
Behavior: exitTheParkinglot() {
                // Insert token to machine
                ParkingLotTokenMachine.CollectPayment(myCreditCard);
                If ( ParkingLotTokenMachine.doorStatus == OPEN) {
                        myVehicle.drive(outside of lot);
                } else {
                        // Try other card or call customer service
                }
        }
```

```
Class Vehicle
Data: vehicleDimensions
Behavior: drive() {
                // Move the vehicle forward
        }

Behavior: park() {
                // Stop the vehicle and pull the hand brakes
                // Put the vehicle on park
        }
```

```
Class ParkingSpot
Data: parkingSpotId, parkinSpotDimensions, isAvailable
Behavior: initializeParkingSpot() {
                // set the ParkinSpotDimensions
        }
Behavior: getDimensions() {
                Return ParkinSpotDimensions;
        }
```

```
Class Sensor
Data: spotId, states // Possible values GREEN and RED
Behavior: senseAndUpdateAvailibility() {
                If ( state change from GREEN - > RED) {
                        ParkingLotTokenMachine.UpdateSpotAvailibility(SpotID, FALSE);
                } else if (RED -> GREEN) {
                        ParkingLotTokenMachine.UpdateSpotAvailibility(SpotID, TRUE);
                }
        }
```

```
Class Display
Behavior: displayToCustomer (List<ParkinSpotIds> spotIds) {
                System.Out.Println(ParkingLotMap.DisplayInGreen(spotIds));
        }
```

```
Class CreditCard
Data: bankName, cardNumber, expiryDate, cVV
Behaviour:  isCardValid(creditCardDetails) {
        If (creditCardDetails == VALID) { // This involves a way of checking with the Bank
                return TRUE;
        }
        return FALSE;
  }
```