GROUP 3

Team members:

1. Bobba Ruthvik CS19B007
2. G Vaibhav CS19B013
3. G Yashwanth CS19B016
4. N Deepika CS19B027
5. Shreetesh M CS19B037

Additional Features:

1. Added different kinds of parking spaces like Economy type, EV type and VIP type. EV type parking spots are specifically designed for electric vehicles and a provision of recharging the vehicles is provided. VIP spots are the premium spots where extra security and damage assurance will be provided, along with the option of vehicle wash.
2. Added an admin class for the administrator to access the floor plan and to reserve and unreserve any number of available spots at any floor. The administrator has access to the list of all the user's data which is stored in a file.
3. The list of information of users like vehicle type, vehicle registration number, time and date are stored in a file named ParkingLot.txt. The total earnings for the day is also stored.

Classes:

1. DisplayBoard class
2. Building class
3. Admin class
4. Abstract Floor class
5. Truck, Car, Bike classes extending Floor class

Object Oriented Principles Implementation:

1. Inheritance:

   Implemented Hierarchical-Inheritance where the classes Truck, Car, Bike extend the class Floor by Is-A relationship. Building class inherits Truck, Car and Bike by Has-A relationship. DisplayBoard class inherits Admin class by Has-A relationship. Inheritance is also used in Admin class.
2. Abstraction:

   Created an abstract Floor class which contains some fields and methods that are common to Truck, Car and Bike classes.
3. Encapsulation:

Achieved encapsulation by wrapping the data into private and protected fields and methods at multiple places. Implemented some getter methods to retrieve the data stored in the private fields and variables.

4. Polymorphism:

   Achieved polymorphism by overloading validate() and show_display() methods in Building class.

Discussions and Design Implementation:

The team came up with an initial idea of creating the classes Building, Floor and Car, Truck and Bike implementing the abstract class Floor. But gradually, we realized that there was a need for another class that constitutes the driver code and interacts with the actors directly by calling the appropriate methods in a menu-driven process. The creation of the class was also supported by the fact that it would improve the secure nature of the code against the idea of implementing this in the main() method. Therefore, DisplayBoard class was created.

The Floor class was implemented as an interface in the early stages of development. But, it was pointed out by the team that there are a few fields and methods that are common to all the classes that implement/extend Floor class. Since interfaces cannot have a method body and non-static variables, interface Floor was converted into an abstract class.

The methods that calculate the amount and accept money from the user were originally implemented in each of the Truck, Car and Bike classes. But it proved to be redundant and inefficient. So, they were decided to be put inside the Building class and DisplayBoard class, since they can be directly accessed from the DisplayBoard class.

Since there was a requirement for another actor, the administrator was decided to be the most sensible actor other than the customer. Hence, a separate Admin class was created which prompts the admin to login by entering the correct password in 5 chances.

Java Collections was thought to be the most suitable way of storing the user's data by using a list of map of array list in a readable format. But, the team noticed that it would make the code inefficient since it has to store large amounts of data. Hence, file handling was used by invoking the FileWriter class.

Many fields and arrays related to different floors have been made static since single instances must be created or shared and updated simultaneously.

To simulate a real world parking lot system, it was decided to provide the user a unique token number on entering the building, which the user has to remember and

enter the number while exiting. This is convenient to the user compared to remembering the floor number and spot number.

The need for validation was evident in many places in the driver code. Hence, instead of creating different methods for validating different types of user inputs, a common validate method was created and overloaded.

Contribution:

1. Bobba Ruthvik - implemented Building class and its methods. Helped in creating Admin class and DisplayBoard class. Helped in implementing file handling system. Gave the idea of creating a Bank class that accepts money and provides change. Helped in code testing. Improved some methods in Car, Truck and Bike classes.
2. G Vaibhav - implemented DisplayBoard class and its methods. Gave the idea of using file handling. Helped in implementing file handling system. Implemented polymorphism. Gave the idea of giving a limited number of chances to the admin to enter the correct password. Implemented encapsulation. Helped in code testing.
3. Shreetesh M - implemented Floor class, Car, Truck and Bike classes. Implemented abstraction and encapsulation. Gave the idea of storing the user's details in a list of map of array list using Java Collections. Helped in implementing Building class. Used the idea of static arrays to create one instance and share it among all methods and classes. Gave the idea of displaying the floor plan in the form of a matrix. Helped in code testing.
4. G Yashwanth - implemented Admin class. Gave the idea of using abstract class instead of interface. Helped in implementing DisplayBoard and Building classes. Suggested some changes in Car, Truck and Bike classes.
5. N Deepika - Helped in implementing payment methods (Implemented a prototype of Payment class, which has been eventually converted into a method). Helped in implementing DisplayBoard class. Gave the idea of implementing a VIP space.

Note: It took our collective effort to implement each class completely.