How to build a simple parking system in Java

Java can help you create a simple parking system that you can then design and integrate with the database, add authentication, develop a GUI to implement in the real world. Here are detailed instructions.

Java can help you create a simple parking system that you can then design and integrate with the database, add authentication, develop a GUI to implement in the real world. Here are detailed instructions.

How to build a parking system in Java

The example in the article uses ArrayList - a common and customizable array of sizes. You can access elements with an index, implement CRUD operations, and much more. To use ArrayList, you need to import the ArrayList class from the Java standard library. Similarly, for the input-input operation, enter the Scanner class. Define a public class, **VehiceParkingSystem** as the main class.

Define 3 static variables: **totalSlots** , **availableSlots** , and **parkedCars** . **totalSlots** is the total number of parking slots, **availableSlots** keeps track of the number of spaces left. **parkedCars** is **an ArrayList** that contains the license plates of currently parked cars.

Define the main() function and create an object of the Scanner class. Ask the user to enter the total number of parking spaces & save it in **totalSlots** . Initially, available slots are equal to total parking spaces so initialize **availableSlots** to the same value as **totalSlots** .

```
import java.util.ArrayList; import java.util.Scanner; public class VehicleParking
```

Use while loop to run indefinitely. Ask the user if he wants to park the car, remove it, see if the car is parked, or exit the program. Depending on the selection, call the corresponding method. If the user wants to end this program, use **System.exit(0)** to terminate immediately.

```
while (true) { System.out.println("n What would you like to do?"); System.out.pr
```

The public static method definition, **parkCar()** , has no return type. If the parking location is not available, notify the user and return. If not, ask the user to enter the license plate and use the **add()** function to insert it into the ArrayList. Reduce the number of available slots to 1 and display the parking program and the number of available slots.

```
public static void parkCar() { if (availableSlots == 0) { System.out.println("So
```

Define a function, **removeCar()** . If the vacancies value and the total number of slots match, there is no parking available and go back. If not, ask the user for the license plate number. Check the number plate the user entered in **the ArrayList** with **contains()** .

If successful, use the remove() method to remove that data from the ArrayList and increment the number of available parking slots. It is now announced that this program has removed that vehicle along with the current number of vacancies. In case you can't find the license plate, it means that the car is not currently parked in the garage.

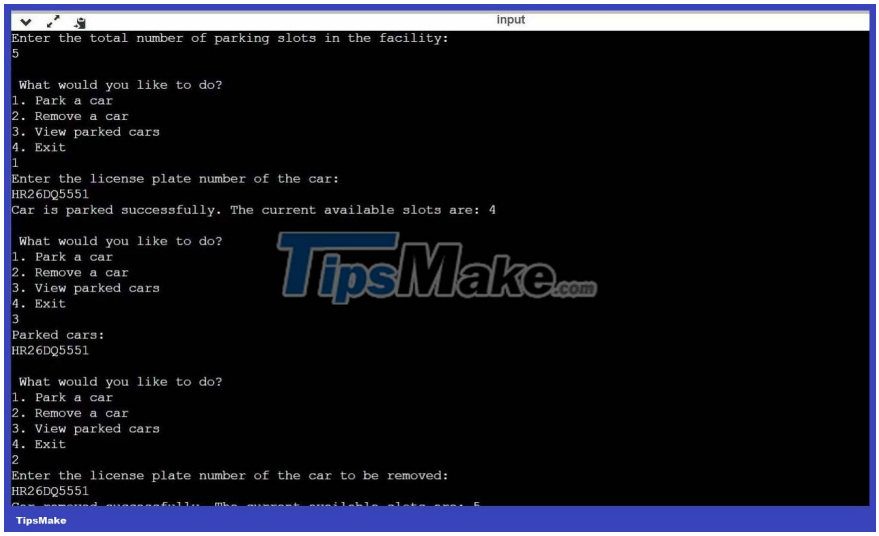
```
public static void removeCar() { if (availableSlots == totalSlots) { System.out
```

Define the method, **viewParkedCars()** . Same as above, check to see if the parking space is full. If not, show parked license plates. To do this, use lopp for-each to iterate over the items in the ArrayList and display each number plate in turn.

```
public static void viewParkedCars() { if (availableSlots == totalSlots) { Syster
```

A simple parking system management program is ready for you to use.

And this is the result:



Good luck!