**CAR PARKING SYSTEM BY USING JAVA**

**A PROJECT REPORT**

**CSA0906 - Programming in Java with Full Stack**

**SUBMITTED BY**

**M.Jyothieswar Reddy (192211468)**

**DEPARTMENT OF COMPUTER SCIENCE**

**AND ENGINEERING**



**SAVEETHA SCHOOL OF ENGINEERING SAVEETHA NAGAR, THANDALAM,**

**SIMATS, CHENNAI – 602105**

**SEPTEMBER 2024**

**Bonafide Certificate**

This is to certify that the project report entitled “Car Parking System by using java” submitted by “M.Jyothieswar Reddy (192211468)” to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of bonafide work carried out by him/her under my guidance. The project fulfils the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

## DR. RAMAMOORTHY MADHAN

Department of Applied Machine Learning

Saveetha School of Engineering

SIMATS, Chennai – 602105

 Internal examiner External Examiner

**Table Of Contents**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NO** |
| 1 | ABSTRACT | 4 |
| 2 | INTRODUCTION | 4 |
| 3 | OVERVIEW | 5 |
| 4 | SOURCE CODE | 7 |
| 5 | SCREENSHOTS(OUTPUTS) | 9 |
| 6 | CONCLUSION & FUTURE ENHANCEMENTS | 10 |
| 7 | REFERENCES | 10 |

**Abstract:**

The Parking System is a console-based application developed in Java to manage parking spaces efficiently. It allows users to park, remove, and view cars, while also tracking the availability of parking slots. The system uses an `ArrayList` to dynamically store vehicle license plate numbers, with functionalities for adding and removing cars based on user input. This system provides a real-time view of available slots, helping to organize parking in a streamlined manner.

The core of the application revolves around user interaction through a text-based interface. Users are prompted to choose from various actions: park a car, remove a car, or view all parked cars. If the parking lot is full, the system alerts users and denies further entries. Similarly, users can remove their vehicles by entering the correct license plate, which frees up a slot. The system continuously updates the available parking slots, ensuring the data remains current throughout its operation.

This project demonstrates the basic principles of Java, such as array manipulation, user input handling, and control structures like loops and conditionals. Though simple, the system can be expanded to include more advanced features like graphical user interfaces (GUI) or integration with databases for larger-scale applications.

**Introduction:**

Parking management has become a critical aspect of modern urban planning due to the rapid increase in vehicle ownership. In densely populated areas, finding a suitable parking space is a challenge, especially in commercial zones, malls, airports, and office complexes. The need for efficient and well-organized parking systems has become evident, and technological solutions are emerging to address these challenges.

One such solution is an automated parking management system, which simplifies the parking process by tracking available parking slots, maintaining records of parked vehicles, and streamlining the entry and exit of vehicles. The system ensures that users have quick access to parking slots while providing management teams with a clear view of parking space usage. This project focuses on implementing a basic parking system using Java programming language, leveraging the power of object-oriented design, data structures, and user input handling.

The **Parking System using Java** is a console-based application designed to manage a limited number of parking slots. Users can interact with the system to park their cars, remove parked cars, or view the current list of parked cars. The system dynamically tracks the number of available parking slots and ensures that only valid actions are performed. Java’s standard library classes, such as ArrayList and Scanner, are used to manage and manipulate the list of parked vehicles, making the system both flexible and efficient.

Java was chosen for this project due to its object-oriented nature, portability, and the vast ecosystem of built-in libraries. By utilizing an ArrayList, we can manage the parked cars efficiently without worrying about predefined limits. The flexibility of Java's collections framework allows for easy expansion of the project if needed, making it suitable for real-world parking systems that may involve more complex data structures.

The project follows a simple architecture where a series of menu options allow users to interact with the system. Users can park a car by entering its license plate number, remove a parked car when leaving, and view a list of currently parked cars. The system is equipped with validation checks to ensure that the user cannot park a car if the parking lot is full or remove a car that isn't parked in the system.

One of the main goals of this project is to simulate the basic operations of a parking system while maintaining a focus on usability and ease of interaction. The application provides real-time updates to users about the number of available parking slots and stored cars, making it an ideal solution for smaller parking areas such as those in residential complexes, small businesses, or event venues.

The simplicity of the system design ensures that users, even without technical knowledge, can navigate through the operations easily. Furthermore, the code has been designed to be modular and extensible, allowing for the potential addition of more features such as multi-level parking, parking fees calculation, or integration with a database to store car records persistently.

In conclusion, the **Parking System using Java** is a fundamental approach to understanding how programming can solve real-world problems. It introduces key concepts such as data management, user interaction, and control flow within the scope of parking management. While this project serves as a basic prototype, it opens the doors for future expansions and improvements, making it a scalable solution for different parking management needs.

**Overview:** The Parking System application is a console-based program designed to manage parking slots efficiently. The program enables users to park and remove cars, as well as view the list of currently parked cars. It is implemented using Java, leveraging ArrayList for dynamic storage and Scanner for user input.

**Key Components:**

1. **Data Variables:**
   * totalSlots: Total number of parking slots available.
   * availableSlots: Number of parking slots currently available.
   * parkedCars: An ArrayList to keep track of license plate numbers of parked cars.
2. **Main Method:**
   * Initializes the Scanner to read user input.
   * Prompts the user to enter the total number of parking slots.
   * Enters a loop allowing users to choose different actions: park a car, remove a car, view parked cars, or exit the program.
3. **Functional Methods:**
   * **parkCar() Method:**
     + Checks if there are available parking slots.
     + Prompts the user to enter the license plate number of the car.
     + Adds the license plate to the parkedCars list.
     + Updates the number of available slots.
     + Displays a confirmation message along with the updated number of available slots.
   * **removeCar() Method:**
     + Checks if there are any cars parked (i.e., available slots equal total slots).
     + Prompts the user to enter the license plate number of the car to be removed.
     + Checks if the car's license plate is in the parkedCars list.
     + Removes the car from the list and updates the number of available slots.
     + Displays a message confirming the removal or indicating that the car is not found.
   * **viewParkedCars() Method:**
     + Checks if there are any parked cars (i.e., available slots equal total slots).
     + Prints the list of all parked cars.
     + If no cars are parked, informs the user.

**User Interaction:**

* The user is continuously prompted for input until they choose to exit the program.
* The console provides feedback based on the actions performed, such as successful parking or removal of cars, and current status of parking slots.

**Resource Management:**

* The Scanner is used to read user input for all operations. Proper management involves closing the scanner after its use, which is included in the enhanced code.

**Usage:**

* **Parking a Car**: Allows the user to park a new car by entering its license plate number, provided there are available slots.
* **Removing a Car**: Enables the user to remove a car from the parking system using its license plate, provided the car is parked.
* **Viewing Parked Cars**: Displays a list of all currently parked cars, allowing users to see which cars are in the parking lot.

**Limitations:**

* The current implementation does not include a graphical user interface (GUI), and all interactions are done via the console.
* The system does not persist data between runs; it resets every time the program is restarted.
* More advanced features like handling multiple parking lots, fee calculation, or integration with a database are not implemented.

**Future Improvements:**

* Adding a GUI using Java Swing or JavaFX for a more user-friendly experience.
* Implementing data persistence using file storage or a database to maintain information across sessions.
* Enhancing functionality to support multiple parking lots, dynamic pricing, and real-time monitoring.

**Source Code:**

import java.util.ArrayList;

import java.util.Scanner;

public class ParkingSystem {

static int totalSlots, availableSlots;

static ArrayList<String> parkedCars = new ArrayList<String>();

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the total number of parking slots:");

totalSlots = sc.nextInt();

availableSlots = totalSlots;

while (true) {

System.out.println("\nWhat would you like to do?");

System.out.println("1. Park a car");

System.out.println("2. Remove a car");

System.out.println("3. View parked cars");

System.out.println("4. Exit");

int choice = sc.nextInt();

switch (choice) {

case 1:

parkCar();

break;

case 2:

removeCar();

break;

case 3:

viewParkedCars();

break;

case 4:

System.exit(0);

default:

System.out.println("Invalid choice. Please try again.");

}

}

}

public static void parkCar() {

if (availableSlots == 0) {

System.out.println("Sorry, there are no available parking slots.");

return;

}

Scanner sc = new Scanner(System.in);

System.out.println("Enter the license plate number of the car:");

String licensePlate = sc.nextLine();

parkedCars.add(licensePlate);

availableSlots--;

System.out.println("Car parked successfully. Available slots: " + availableSlots);

}

public static void removeCar() {

if (availableSlots == totalSlots) {

System.out.println("There are no parked cars.");

return;

}

Scanner sc = new Scanner(System.in);

System.out.println("Enter the license plate number of the car to be removed:");

String licensePlate = sc.nextLine();

if (parkedCars.contains(licensePlate)) {

parkedCars.remove(licensePlate);

availableSlots++;

System.out.println("Car removed successfully. Available slots: " + availableSlots);

} else {

System.out.println("The car is not parked here.");

}

}

public static void viewParkedCars() {

if (availableSlots == totalSlots) {

System.out.println("There are no parked cars.");

return;

}

System.out.println("Parked cars:");

for (String licensePlate : parkedCars) {

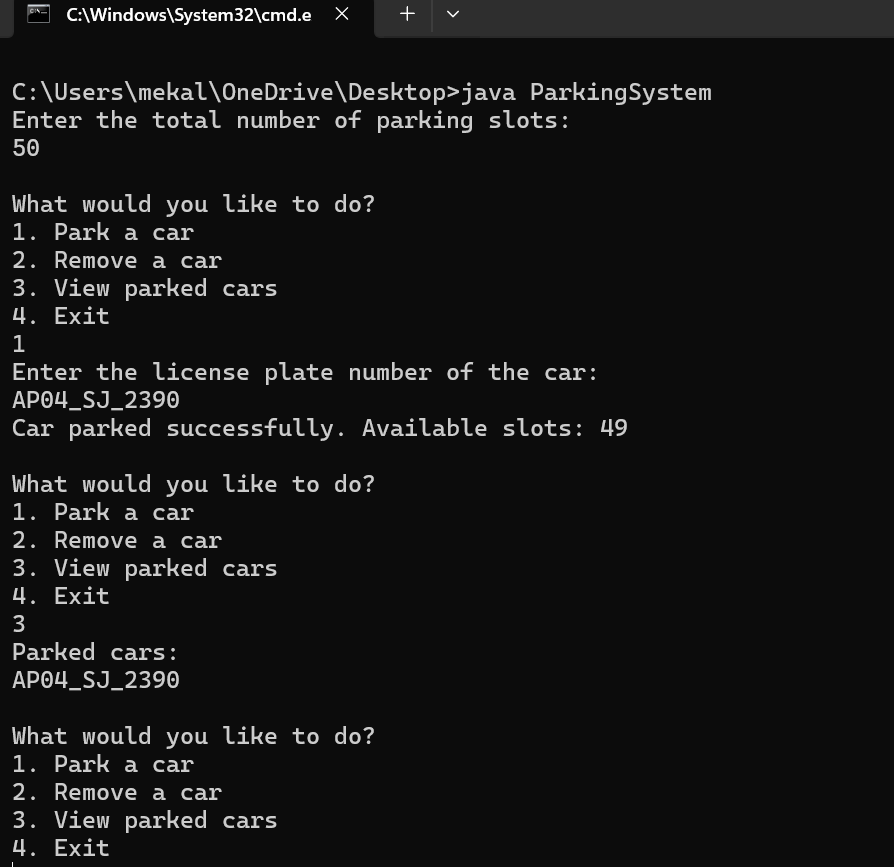
System.out.println(licensePlate);

}

}

}

**Output:**

****

**Conclusion:**

The Parking System implemented in Java offers a straightforward and efficient solution for managing parking slots. Through its console-based interface, users can park and remove cars while keeping track of available parking spaces. The system leverages an ArrayList to maintain a dynamic list of parked vehicles and employs basic control structures to handle user input and perform operations. Despite its simplicity, the system effectively demonstrates core Java concepts such as input handling, list manipulation, and control flow. For future enhancements, adding a graphical user interface (GUI) and integrating a persistent storage mechanism could significantly improve the system's usability and functionality. Overall, this project provides a solid foundation for understanding and developing more complex parking management solutions.

**Future Enhancement:**

 **GUI Implementation**: Develop a graphical user interface using Java Swing or JavaFX for a more user-friendly experience.

 **Database Integration**: Use a database like MySQL or SQLite to persist parking data across sessions.

 **User Authentication**: Add user authentication and authorization to manage parking slots and car details securely.

 **Fee Calculation**: Implement a fee calculation system based on parking duration and slot type.

 **Multi-Level Parking**: Extend the system to support multi-level parking structures.

 **Real-Time Monitoring**: Integrate sensors or real-time tracking to monitor slot occupancy.

 **Mobile App Integration**: Develop a mobile application for users to check parking availability and manage reservations.

 **Automated Ticketing**: Introduce automated ticketing for entry and exit with barcode or QR code scanning.

 **Alerts and Notifications**: Implement alerts for nearing expiry of parking time or slot availability changes.

 **Enhanced Reporting:** Add detailed reporting features for usage statistics and slot occupancy trends.

**Reference:**

**Oracle Java Documentation**

* **Source**: [Oracle Java Documentation](https://docs.oracle.com/javase/8/docs/)
* **Description**: The official Java documentation provides comprehensive information on Java classes and methods, including Scanner, ArrayList, and other core classes used in your project.

** Java Arrays and ArrayLists Tutorial**

* **Source**: Java Arrays and ArrayLists Tutorial
* **Description**: GeeksforGeeks provides tutorials on Java data structures, including ArrayList, which is used in the parking system to manage parked cars.

 **Java Scanner Class Tutorial**

* **Source**: Java Scanner Class Tutorial
* **Description**: W3Schools offers an introduction to the Scanner class, which is used for reading user input in your parking system.

 **Java Programming Basics**

* **Source**: Java Programming Basics
* **Description**: TutorialsPoint provides a broad overview of Java programming, including control statements and basic programming constructs used in your parking system.

 **Effective Java by Joshua Bloch**

* **Source**: [Effective Java by Joshua Bloch](https://www.amazon.com/Effective-Java-Joshua-Bloch/dp/0134685997)
* **Description**: This book is a highly recommended reference for best practices and effective Java programming techniques, offering insights that can help improve and refine your code.