## Introduction

The ParkingSpotSystemGUI is a graphical user interface (GUI) developed to manage a parking lot system with functionalities such as adding and removing parking slots, parking vehicles, and monitoring parking durations. The system consists of several components, each fulfilling a specific role such as displaying parking slots, gathering user input, and providing real-time updates. This user-friendly interface utilizes GUI elements like JPanel, JButton, and JLabel for visual display and user interaction. The system also incorporates event-driven programming with ActionListener, MouseAdapter, and Timer to ensure smooth interactions and timely updates.

## GUI Components

### 1.1 Main Window (JFrame)

* **Name:** frame  
  **Description:** This is the primary window of the application, containing all other GUI components such as panels, labels, and buttons. It sets the layout and controls the overall interface.

### 1.2 Panels (JPanel)

* **Name:** panel  
  **Description:** This panel organizes user interaction buttons like "Add Parking Slot," "Park Car," and others in a vertical layout on the left side of the interface.
* **Name:** slotsPanel  
  **Description:** A grid layout panel used to display parking slots as labels. Each slot is clickable and shows details such as the slot identifier, status, and parked car details.
* **Name:** legendPanel  
  **Description:** A panel that holds the legend for slot types, helping users identify staff and visitor slots by color.

### 1.3 Labels (JLabel)

* **Name:** staffLegend  
  **Description:** Displays a blue label in the legend panel to represent staff parking slots.
* **Name:** visitorLegend  
  **Description:** Displays a yellow label in the legend panel to represent visitor parking slots.
* **Name:** slotLabels  
  **Description:** These labels represent individual parking slots. They display the slot identifier, parking status, and additional details such as time parked and fare. The labels change color depending on the slot’s occupation status.

### 1.4 Buttons (JButton)

**1.4.1 addSlotButton**

* **Description:** Allows the user to add a new parking slot. The button opens a dialog box for inputting the slot identifier and specifying whether it is a staff or visitor slot.

**1.4.2 parkCarButton**

* **Description:** Lets the user park a car into a selected slot. Upon clicking, a dialog prompts the user to enter the car registration number, owner’s name, and whether the car belongs to a staff member.

**1.4.3 removeCarButton**

* **Description:** Enables the removal of a car from a slot. The user is asked to provide the car registration number, after which the car is removed, and the slot is freed.

**1.4.4 findCarButton**

* **Description:** Allows the user to find a car based on its registration number. Displays the parking slot details and the duration for which the car has been parked if the car is found.

**1.4.5 removeByRegButton**

* **Description:** Removes a car from the parking lot by registration number. The car is removed from its slot, and the user receives confirmation.

**1.4.6 deleteUnoccupiedSlotsButton**

* **Description:** Deletes all unoccupied parking slots from the system. When clicked, this button clears any slots that are not currently holding a car.

**1.4.7 deleteSlotButton**

* **Description:** Deletes a specific parking slot, provided it is not currently in use. The user must provide the slot ID.

**1.4.8 exitButton**

* **Description:** Closes the application after confirming with the user. It ensures that the user doesn't accidentally exit without saving any necessary information.

### 1.5 Dialogs and Input Handling (JOptionPane)

* **Name:** JOptionPane  
  **Description:** Handles input from the user via pop-up dialogs. It prompts for information like car registration numbers and slot IDs. It is also used to display error or confirmation messages.

### 1.6 Timer (Timer)

* **Name:** Timer  
  **Description:** A background timer that triggers the regular updating of parking slot details, ensuring that the time parked and parking fees are updated in real-time. It refreshes every second.

## Event handling functions

### 1. addSlotButton ActionListener

**Event Triggered:** Clicking the "Add Parking Slot" button.

**Function Description:**  
The addParkingSlot() method is invoked when the "Add Parking Slot" button is clicked. This method prompts the user to enter a slot ID (e.g., "A01" for staff slots, "B01" for visitor slots) and whether the slot is for staff or visitors. The new slot is added to the CarPark object, and the GUI is updated to display the new slot.

### 2. parkCarButton ActionListener

**Event Triggered:** Clicking the "Park Car" button.

**Function Description:**  
The parkCarInSlot() method is called when the "Park Car" button is clicked. It prompts the user for the slot ID, car registration number, and owner’s name. It checks if the car can be parked in the specified slot, verifying if the slot is empty and whether the car matches the slot type (staff or visitor). If valid, the car is parked, and the display is updated.

### 3. removeCarButton ActionListener

**Event Triggered:** Clicking the "Remove Car from Slot" button.

**Function Description:**  
The removeCarFromSlot() method is executed when the "Remove Car from Slot" button is clicked. It prompts the user for the car’s registration number and, if found in the parking system, removes the car from the corresponding slot. The GUI is refreshed to show the slot as empty.

### 4. findCarButton ActionListener

**Event Triggered:** Clicking the "Find Car by Registration" button.

**Function Description:**  
The findCarByRegNumber() method is triggered when the "Find Car by Registration" button is clicked. The user is prompted to enter the car’s registration number. If the car is found in a slot, its details (slot ID, owner’s name, parking duration, and current fare) are displayed in a message dialog.

### 5. removeByRegButton ActionListener

**Event Triggered:** Clicking the "Remove Car by Registration" button.

**Function Description:**  
The removeCarByRegNumber() method is executed when the "Remove Car by Registration" button is clicked. The user is prompted for the car’s registration number. If the car is found, it is removed from the corresponding slot, and the display is updated to reflect the removal.

### 6. deleteUnoccupiedSlotsButton ActionListener

**Event Triggered:** Clicking the "Delete Unoccupied Slots" button.

**Function Description:**  
The deleteAllUnoccupiedParkingSlots() method is called when the "Delete Unoccupied Slots" button is clicked. This function removes all parking slots that are currently empty. After deletion, the GUI is updated to reflect the changes.

### 7. deleteSlotButton ActionListener

**Event Triggered:** Clicking the "Delete Parking Slot" button.

**Function Description:**  
The deleteParkingSlot() method is invoked when the "Delete Parking Slot" button is clicked. The user is asked to provide the slot ID. If the slot exists and is unoccupied, it is deleted from the system, and the display is refreshed to reflect the removal.

### 8. exitButton ActionListener

**Event Triggered:** Clicking the "Exit" button.

**Function Description:**  
When the "Exit" button is clicked, an ActionListener calls System.exit(0) to terminate the program after displaying a message dialog confirming the exit. This ensures that the user is notified before the application closes.

### 9. MouseListener for slotLabels

**Event Triggered:** Clicking on any parking slot label.

**Function Description:**  
A MouseAdapter is added to each JLabel representing a parking slot. When a slot is clicked, a JOptionPane is displayed with two options: "Park a Car" or "Remove Car." Depending on the user's choice, it either calls parkCarInSlot() or removeCarFromSlot() to handle the action. This allows interactive slot management directly from the slot labels.

### 10. Timer ActionListener

**Event Triggered:** A timer refreshes the parking slot display every second.

**Function Description:**  
The Timer object runs every 1000 milliseconds (1 second) and calls the updateSlotsDisplay() method to refresh the parking slots display. This ensures the parking duration and fare information is updated in real-time for each occupied slot.

## Conclusion

The ParkingSpotSystemGUI delivers an effective and easy-to-use solution for managing parking operations. Users can effortlessly add slots, park or remove cars, and track real-time parking durations and fees. By incorporating various GUI components and event-handling mechanisms, the system offers a seamless and responsive user experience. The application uses event-driven programming with ActionListener for button actions and MouseAdapter for interactive slot labels, making it responsive and efficient. This system is ideal for environments where organized and real-time management of parking facilities is essential.