

SMART PARKING SYSTEM

```
package parking;
import java.util.*;
class ParkingSlot {
    private final int slotId;
    private boolean isOccupied;
    private String vehicleNumber;
    private long startTime;
    public ParkingSlot(int slotId) {
        this.slotId = slotId;
        this.isOccupied = false;
    }
    public boolean isOccupied() {
        return isOccupied;
    }
    public void parkVehicle(String vehicleNumber) {
        this.vehicleNumber = vehicleNumber;
        this.isOccupied = true;
        this.startTime = System.currentTimeMillis();
    }
    public double releaseVehicle() {
        if (!isOccupied) {
            System.out.println("Slot " + slotId + " is already empty.");
            return 0.0;
        }
        this.isOccupied = false;
        long endTime = System.currentTimeMillis();
        long duration = (endTime - startTime) / 1000; // seconds
        double fee = calculateFee(duration);
        System.out.println("Vehicle " + vehicleNumber + " parked for " + duration + " seconds. Fee: ₹" + fee);
        this.vehicleNumber = null;
        return fee;
    }
    private double calculateFee(long duration) {
        // ₹2 per second, minimum ₹2
        return Math.max(2.0, duration * 2.0);
    }
    public int getSlotId() {
        return slotId;
    }
}
class SmartParkingSystem {
    private final List<ParkingSlot> slots;
    public SmartParkingSystem(int totalSlots) {
        slots = new ArrayList<>();
        for (int i = 1; i <= totalSlots; i++) {
            slots.add(new ParkingSlot(i));
        }
    }
    public void parkVehicle(String vehicleNumber) {
        for (ParkingSlot slot : slots) {
            if (!slot.isOccupied()) {
                slot.parkVehicle(vehicleNumber);
                System.out.println("Vehicle " + vehicleNumber + " parked at slot " + slot.getSlotId());
                return;
            }
        }
        System.out.println("Parking full! No slots available.");
    }
    public void releaseVehicle(int slotId) {
        for (ParkingSlot slot : slots) {
            if (slot.getSlotId() == slotId) {
                slot.releaseVehicle();
                return;
            }
        }
        System.out.println("Invalid slot ID: " + slotId);
    }
    public void displayStatus() {
        System.out.println("\n--- Parking Status ---");
        for (ParkingSlot slot : slots) {
            System.out.println("Slot " + slot.getSlotId() + ": " + (slot.isOccupied() ? "Occupied" : "Empty"));
        }
    }
}
public class parking {
    public static void main(String[] args) {
        SmartParkingSystem parking = new SmartParkingSystem(5); // 5 slots
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("\n1. Park Vehicle\n2. Release Vehicle\n3. Display Status\n4. Exit");
            System.out.print("Enter choice: ");
            if (!sc.hasNextInt()) {
                System.out.println("Invalid input! Please enter a number.");
                sc.next(); // clear invalid input
                continue;
            }
            int choice = sc.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter vehicle number: ");
                    String vehicle = sc.next();
                    parking.parkVehicle(vehicle);
                    break;
                case 2:
                    System.out.print("Enter slot ID to release: ");
                    if (!sc.hasNextInt()) {
                        System.out.println("Invalid input! Slot ID must be a number.");
                        sc.next();
                        break;
                    }
                    int slotId = sc.nextInt();
                    parking.releaseVehicle(slotId);
                    break;
                case 3:
                    parking.displayStatus();
                    break;
                case 4:
                    System.out.println("Exiting...");
                    sc.close();
                    return;
                default:
                    System.out.println("Invalid choice!");
            }
        }
    }
}
```

edlps workspace - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

parking.java

Console

```
<terminated> parking [Java Application] C:\Users\91953\p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_25.0.1.v20251108-1451\jre\bin\java.exe (30 Dec 2025, 12:14:39 pm - 12:15:22 pm elapsed)

1. Park Vehicle
2. Release Vehicle
3. Display Status
4. Exit
Enter choice: 1
Enter vehicle number: ka07
Vehicle ka07 parked at slot 1

1. Park Vehicle
2. Release Vehicle
3. Display Status
4. Exit
Enter choice: 1
Enter vehicle number: kao9
Vehicle kao9 parked at slot 2

1. Park Vehicle
2. Release Vehicle
3. Display Status
4. Exit
Enter choice: 2
Enter slot ID to release: 2
Vehicle kao9 parked for 12 seconds. Fee: ₹24.0

1. Park Vehicle
2. Release Vehicle
3. Display Status
4. Exit
Enter choice: 3
```

1216 30-12-2025

edlps workspace - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

parking.java

Console

```
<terminated> parking [Java Application] C:\Users\91953\p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_25.0.1.v20251108-1451\jre\bin\java.exe (30 Dec 2025, 12:14:39 pm - 12:15:22 pm elapsed)

Vehicle kao9 parked at slot 2

1. Park Vehicle
2. Release Vehicle
3. Display Status
4. Exit
Enter choice: 2
Enter slot ID to release: 2
Vehicle kao9 parked for 12 seconds. Fee: ₹24.0

1. Park Vehicle
2. Release Vehicle
3. Display Status
4. Exit
Enter choice: 3

--- Parking Status ---
Slot 1: Occupied
Slot 2: Empty
Slot 3: Empty
Slot 4: Empty
Slot 5: Empty

1. Park Vehicle
2. Release Vehicle
3. Display Status
4. Exit
Enter choice: 4
Exiting...
```

1217 30-12-2025

