OOP1 Assignment Report: Car Park Management System

GitHub Repository: https://github.com/ahmedwahba47/oop1-car-park-management

1. Introduction

This report presents a **Car Park Management System** developed as a command-line Java application. The system allows users to:

- Park vehicles (Cars and Motorbikes) by assigning them to available slots
- Unpark vehicles and calculate parking fees based on duration and vehicle type
- View parking status and search for vehicles by type
- Handle edge cases such as full car parks, duplicate registrations, and invalid inputs

---

2. User Stories Completed

#	User Story	Description
1	**Park a vehicle**	User specifies vehicle type (Car/Motorbike) and registration number. System assigns first available slot. Prevents duplicate registrations.
2	**Unpark a vehicle**	User provides slot number. System calculates fee based on duration and vehicle type, returns a ticket.
3	**View parking status**	Displays all slots showing availability and vehicle details (type + registration).
4	**Find vehicles by type**	Search for parked vehicles by type. Returns slot numbers where vehicles are parked.
5	**View specific slot details**	View details of multiple slots using varargs (e.g., slots 1, 3, 5).
6	**Handle full car park**	Gracefully displays "Parking Full" message when capacity is reached.
7	**Handle invalid input**	Validates user input and provides appropriate error messages.

---

3. Evaluation

3.1 Adherence to Project Brief

The application implements **all required language features**:

Fundamentals

Feature	Implementation	Location
Classes	Vehicle, Car, Motorbike, ParkingSlot, Ticket, ParkingService, Money, Main	All .java files
`this()` vs `this.`	`this.` accesses instance variables; `this()` chains to another constructor in same class	Vehicle.java:17-18 (`this.`), Vehicle.java:27 (`this()`)

Method Overloading	Two `park()` methods with different parameter types	ParkingService.java:27, 110
Varargs	`printSlotDetails(int... slotNumbers)` accepts variable number of arguments	ParkingService.java:121
LVTI (`var`)	Local Variable Type Inference - compiler infers type from right-hand side	Main.java:58, 89
Encapsulation	Private fields with public getter methods; state changes through controlled methods	ParkingSlot.java (documented)
Interfaces	Parkable interface with static, default, and private methods	Parkable.java
Inheritance	Car and Motorbike extend abstract Vehicle class	Car.java:5, Motorbike.java:5
Overriding/Polymorphism	`calculateFee()` abstract in parent, overridden differently in each subclass	Car.java:19-22, Motorbike.java:20-23
`super()` vs `super.`	`super()` calls parent constructor; `super.` accesses parent methods/fields	Car.java:16 (`super()`), Car.java:32 (`super.`)
Checked Exception	ParkingFullException extends Exception - compiler enforces handling	ParkingFullException.java
Unchecked Exception	IllegalArgumentException (extends RuntimeException) for invalid input	ParkingService.java:29, 50, 54
Enums	Type-safe constants with compile-time checking	VehicleType.java, ParkingStatus.java
Arrays	ParkingSlot[] array for managing parking slots	ParkingService.java:16
Java Core API	String, StringBuilder, List/ArrayList, Set/HashSet, LocalDateTime	ParkingService.java:76 (StringBuilder), 17-18 (List/Set), 57-58 (DateTime)

Advanced

Feature	Implementation	Location
Call-by-value & Defensive Copying	Primitives passed by value; immutable objects (Record, Money) need no defensive copy	ParkingService.java:42-47 (documented)
Private/Default/Static Interface Methods	All three types demonstrated	Parkable.java:5-7 (static), 10-13 (default), 16-25 (private)
Records	Ticket is a Java record - immutable data carrier with auto-generated methods	Ticket.java (with documentation)
Custom Immutable Type	Money class: final class, final field, no setters, methods return new instances	Money.java (with documentation)

Lambdas (Predicate)	`findVehicles(Predicate<Vehicle>)` accepts lambda expressions	ParkingService.java:96, Main.java:116-119
Final/Effectively Final	Lambdas can only capture final or effectively final variables (never reassigned)	ParkingService.java:89-94 (documented)
Method References	`System.out::println` shorthand for lambda	Main.java:130
Switch Expressions	Returns value directly using arrow syntax, no fall-through	ParkingService.java:111-114, Main.java:160-164 (documented)
Pattern Matching	`v instanceof Car car` checks type AND creates typed variable	Main.java:116, 119 (documented)
Sealed Classes	Vehicle is sealed, permits only Car and Motorbike	Vehicle.java:8

Java 25 Features (Extra Marks)

Feature	Description	Location
Instance Main Methods (JEP 512)	Simplified entry point: `void main()` instead of `public static void main(String[] args)`	Main.java:21 (with documentation)
Flexible Constructor Bodies (JEP 513)	Validation logic executes BEFORE `super()` call - previously impossible	Car.java:13-16, Motorbike.java:14-17

3.2 Problems Encountered

1. **Java 25 Environment Setup**: Required careful `pom.xml` configuration for the new JDK and enabling preview features with `--enable-preview` flag.
2. **JUnit Compatibility**: Default Maven archetype generated incompatible JUnit 4 tests; resolved by migrating to JUnit 5.

3.3 How to Get Java 25 Working

Prerequisites: Install JDK 25 (Early Access) and configure your environment.

pom.xml Configuration:

```xml
<properties>
    <maven.compiler.source>25</maven.compiler.source>
    <maven.compiler.target>25</maven.compiler.target>
</properties>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.11.0</version>
    <configuration>
        <compilerArgs>
            <arg>--enable-preview</arg>
        </compilerArgs>
    </configuration>
</plugin>
```

Run Commands:

```
# Run application
java --enable-preview --source 25 src/main/java/Main.java

# Run tests
mvn test
```

---

4. UML Class Diagram

```
┌────────────────────────────────────────────────────────────────────────────┐
│                             <<interface>>                                  │
│                               Parkable                                     │
├────────────────────────────────────────────────────────────────────────────┤
│ + {static} getParkingInfo(): String                                        │
│ + {default} printRegistration(): void                                      │
│ - printVin(): void {private}                                               │
│ + getRegistrationNumber(): String                                          │
│ - generateVin(): String {private}                                          │
└────────────────────────────────────────────────────────────────────────────┘
                                      △
                                      │ implements
┌─────────────────────────────────────┴──────────────────────────────────────┐
│                         <<abstract, sealed>>                               │
│                              Vehicle                                       │
├────────────────────────────────────────────────────────────────────────────┤
│ - registrationNumber: String {final}                                       │
│ - type: VehicleType {final}                                                │
├────────────────────────────────────────────────────────────────────────────┤
│ + Vehicle(registrationNumber, type)  ← uses this.                          │
│ + Vehicle(registrationNumber)        ← uses this() constructor chaining    │
│ + getRegistrationNumber(): String                                          │
│ + getType(): VehicleType                                                   │
│ + calculateFee(long): Money {abstract}                                     │
└────────────────────────────────────────────────────────────────────────────┘
                         △                       △
           extends       │                       │     extends
              ┌──────────┴──────────┐ ┌─────────┴──────────┐
              │   Car <<final>>     │ │ Motorbike <<final>>│
              ├─────────────────────┤ ├────────────────────┤
              │ + Car(regNum)       │ │ + Motorbike(regNum)│
              │   ↳ JEP 513: code   │ │   ↳ JEP 513: code  │
              │     before super()  │ │     before super() │
              │   ↳ super() used    │ │   ↳ super() used   │
              │ + calculateFee()    │ │ + calculateFee()   │
              │ + toString()        │ └────────────────────┘
              │   ↳ super. used     │
              └─────────────────────┘


┌─────────────────────────┐       ┌─────────────────────────┐
│   <<record>> Ticket     │       │  <<immutable>> Money    │
```

```
├─────────────────────────┤        ├─────────────────────────┤
│ registrationNumber      │◇───────│ - amount: BigDecimal    │
│ slotNumber              │        ├─────────────────────────┤
│ entryTime               │        │ + Money(String)         │
│ exitTime                │        │ + Money(double)         │
│ fee: Money              │        │ + add(Money): Money     │
└─────────────────────────┘        └─────────────────────────┘

┌─────────────────────────┐        ┌─────────────────────────┐
│       ParkingSlot       │        │     ParkingService      │
│     (encapsulation)     │        ├─────────────────────────┤
├─────────────────────────┤        │ - slots: ParkingSlot[]  │ ← Array
│ - slotNumber: int       │        │ - ticketHistory: List   │ ← ArrayList
│ - vehicle: Vehicle      │◇───────│ - parkedRegNums: Set    │ ← HashSet
│ - status: ParkingStatus │        ├─────────────────────────┤
│ - entryTime: DateTime   │        │ + park(Vehicle): int    │ ← overloaded
├─────────────────────────┤        │ + park(String,Type): int│ ← switch expr
│ + getters...            │        │ + unpark(int): Ticket   │
│ + park(Vehicle): void   │        │ + findVehicles(Pred): L │ ← Predicate/lambda
│ + unpark(): void        │        │ + printSlotDetails(...) │ ← varargs
└─────────────────────────┘        └─────────────────────────┘

┌───────────────┐   ┌─────────────────┐   ┌─────────────────────────┐
│   <<enum>>    │   │    <<enum>>     │   │     <<checked>>         │
│  VehicleType  │   │  ParkingStatus  │   │  ParkingFullException   │
├───────────────┤   ├─────────────────┤   │   extends Exception     │
│ CAR           │   │ OCCUPIED        │   └─────────────────────────┘
│ MOTORBIKE     │   │ AVAILABLE       │
└───────────────┘   └─────────────────┘

┌───────────────────────────────────────────────────────────────────────────────┐
│                                   Main                                        │
├───────────────────────────────────────────────────────────────────────────────┤
│ + void main()                           ← JEP 512: Instance Main Method       │
│ - parkVehicle(...)                      ← uses var (LVTI)                     │
│ - unparkVehicle(...)                    ← uses var (LVTI)                     │
│ - findVehicles(...)                     ← pattern matching, method reference │
│ - viewSpecificSlots(...)                │
│ - getVehicleTypeFromInput(String)       ← switch expression                   │
└───────────────────────────────────────────────────────────────────────────────┘
```

---

5. Eclipse Setup Instructions

1. Install JDK 25 on your system
2. In Eclipse: `Window > Preferences > Java > Installed JREs` → Add JDK 25
3. `File > Import > Maven > Existing Maven Projects` → Browse to `car-park-management`
4. Run `Main.java` as Java Application
5. Run `ParkingServiceTest.java` as JUnit Test