OOP1 Assignment Report: Car Park Management System

Prepared by: Ahmed Wahba

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:26, 42
Varargs	`printSlotDetails(int... slotNumbers)` accepts variable number of arguments	ParkingService.java:120
LVTI (`var`)	Local Variable Type Inference - compiler infers type from right-hand side	Main.java:61, 92
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:3, Motorbike.java:3
Overriding/Polymorphism	`calculateFee()` abstract in parent, overridden differently in each subclass	Car.java:17-20, Motorbike.java:18-21
`super()` vs `super.`	`super()` calls parent constructor; `super.` accesses parent methods/fields	Car.java:14 (`super()`), Car.java:30 (`super.`)
Checked Exception	ParkingFullException extends Exception - compiler enforces handling	ParkingFullException.java
Unchecked Exception	IllegalArgumentException (extends RuntimeException) for invalid input	ParkingService.java:28, 58, 62
Enums	Type-safe constants with compile-time checking	VehicleType.java, ParkingStatus.java
Arrays	ParkingSlot[] array for managing parking slots	ParkingService.java:15
Java Core API	String, StringBuilder, List/ArrayList, Set/HashSet, LocalDateTime	ParkingService.java:84 (StringBuilder), 16-17 (List/Set), 65-66 (DateTime)

Advanced

Feature	Implementation	Location
Call-by-Value & Defensive Copying	Java passes object references by value; defensive copying prevents external modification of mutable objects	ParkingService.java (copies made when returning mutable state)
Private/Default/Static Interface Methods	All three types demonstrated	Parkable.java:8-11 (static), 13-17 (default), 19-27 (private)

Records	Immutable data carrier with auto-generated constructor, getters, equals, hashCode, toString	Ticket.java (record class)
Custom Immutable Type	Money class is immutable: final class, final field, no setters, methods return new instances	Money.java (documented)
Lambdas (Predicate)	`findVehicles(Predicate<Vehicle>)` accepts lambda expressions	ParkingService.java:104, Main.java:121-124
Final/Effectively Final	Lambdas can only capture final or effectively final variables (never reassigned)	ParkingService.java:97-102 (documented)
Method References	Shorthand for lambdas: `System.out::println` equivalent to `x -> System.out.println(x)`	Main.java:134 (documented)
Switch Expressions	Returns value directly using arrow syntax, no fall-through	ParkingService.java:43-46, Main.java:165-169 (documented)
Pattern Matching	`v instanceof Car car` checks type AND creates typed variable	Main.java:121, 124 (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:24 (with documentation)
Flexible Constructor Bodies (JEP 513)	Validation logic executes BEFORE `super()` call - previously impossible	Car.java:11-14, Motorbike.java:12-15

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 and configure your environment.

pom.xml Configuration:

```
<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>
```

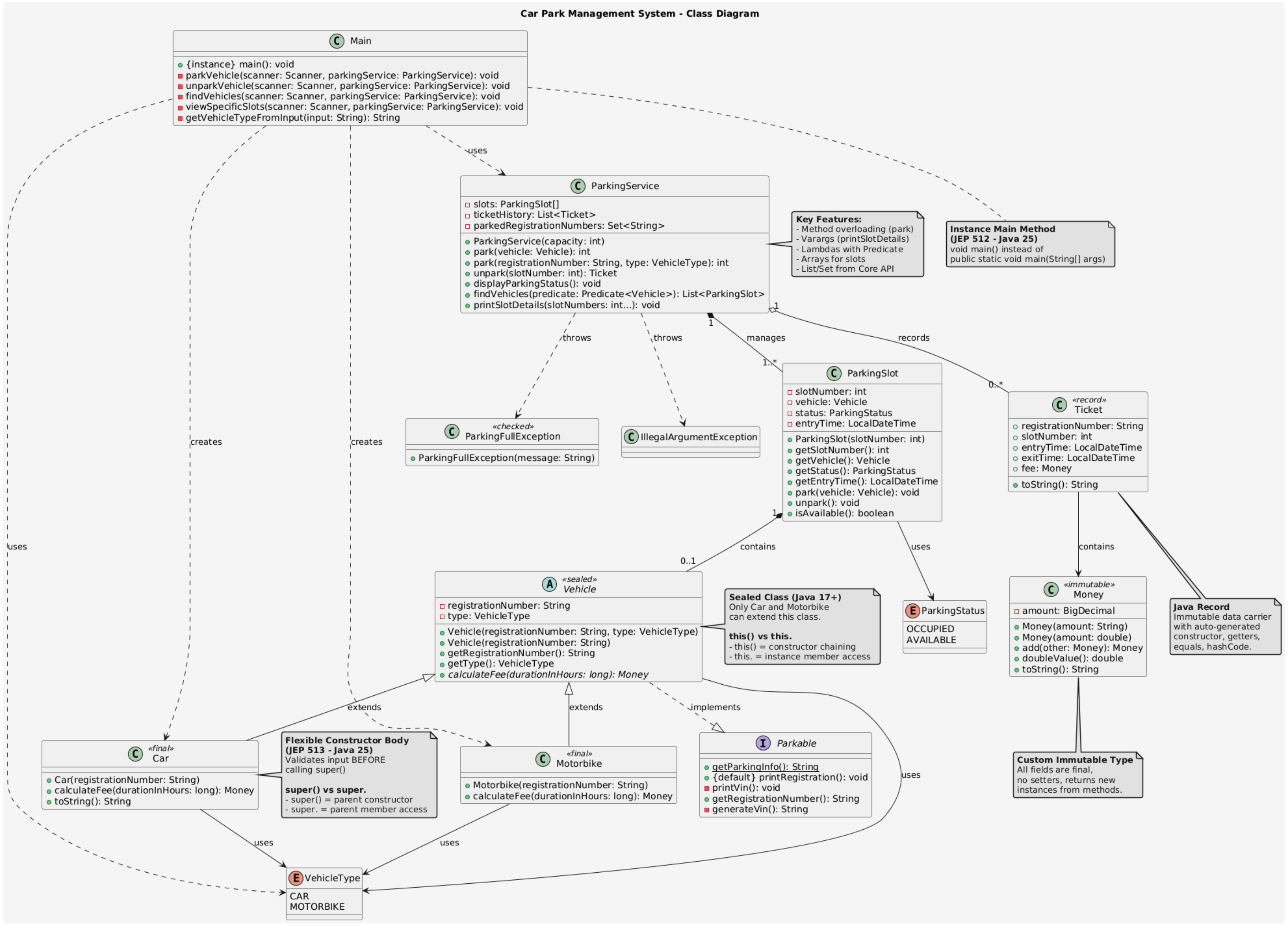
```xml
    <version>3.11.0</version>
    <configuration>
        <compilerArgs>
            <arg>--enable-preview</arg>
        </compilerArgs>
    </configuration>
</plugin>
```

Run Commands:

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

# Run tests
mvn test
```

UML Class Diagram


Car Park Management System - Class Diagram
Main
+ {instance} main(): void
- parkVehicle(scanner: Scanner, parkingService: ParkingService): void
- unparkVehicle(scanner: Scanner, parkingService: ParkingService): void
- findVehicles(scanner: Scanner, parkingService: ParkingService): void
- viewSpecificSlots(scanner: Scanner, parkingService: ParkingService): void
- getVehicleTypeFromInput(input: String): String
uses
ParkingService
- slots: ParkingSlot[]
- ticketHistory: List<Ticket>
- parkedRegistrationNumbers: Set<String>
+ ParkingService(capacity: int)
+ park(vehicle: Vehicle): int
+ park(registrationNumber: String, type: VehicleType): int
+ unpark(slotNumber: int): Ticket
+ displayParkingStatus(): void
+ findVehicles(predicate: Predicate<Vehicle>): List<ParkingSlot>
+ printSlotDetails(slotNumbers: int...): void
Key Features:
- Method overloading (park)
- Varargs (printSlotDetails)
- Lambdas with Predicate
- Arrays for slots
- List/Set from Core API
Instance Main Method
(JEP 512 - Java 25)
void main() instead of
public static void main(String[] args)
throws
throws
1
manages
1..*
1
records
0..*
creates
creates
uses
«checked»
ParkingFullException
+ ParkingFullException(message: String)
IllegalArgumentException
ParkingSlot
- slotNumber: int
- vehicle: Vehicle
- status: ParkingStatus
- entryTime: LocalDateTime
+ ParkingSlot(slotNumber: int)
+ getSlotNumber(): int
+ getVehicle(): Vehicle
+ getStatus(): ParkingStatus
+ getEntryTime(): LocalDateTime
+ park(vehicle: Vehicle): void
+ unpark(): void
+ isAvailable(): boolean
«record»
Ticket
+ registrationNumber: String
+ slotNumber: int
+ entryTime: LocalDateTime
+ exitTime: LocalDateTime
+ fee: Money
+ toString(): String
1
contains
0..1
uses
contains
«sealed»
Vehicle
- registrationNumber: String
- type: VehicleType
+ Vehicle(registrationNumber: String, type: VehicleType)
+ Vehicle(registrationNumber: String)
+ getRegistrationNumber(): String
+ getType(): VehicleType
+ calculateFee(durationInHours: long): Money
Sealed Class (Java 17+)
Only Car and Motorbike
can extend this class.
this() vs this.
- this() = constructor chaining
- this. = instance member access
ParkingStatus
OCCUPIED
AVAILABLE
«immutable»
Money
- amount: BigDecimal
+ Money(amount: String)
+ Money(amount: double)
+ add(other: Money): Money
+ doubleValue(): double
+ toString(): String
Java Record
Immutable data carrier
with auto-generated
constructor, getters,
equals, hashCode.
extends
extends
implements
«final»
Car
+ Car(registrationNumber: String)
+ calculateFee(durationInHours: long): Money
+ toString(): String
Flexible Constructor Body
(JEP 513 - Java 25)
Validates input BEFORE
calling super()
super() vs super.
- super() = parent constructor
- super. = parent member access
«final»
Motorbike
+ Motorbike(registrationNumber: String)
+ calculateFee(durationInHours: long): Money
Parkable
+ getParkingInfo(): String
+ {default} printRegistration(): void
- printVin(): void
+ getRegistrationNumber(): String
- generateVin(): String
uses
Custom Immutable Type
All fields are final,
no setters, returns new
instances from methods.
uses
uses
VehicleType
CAR
MOTORBIKE