

**Lahore Campus**

**Object Oriented Programming**

**Assignment # 1**

**Correction of Midterm lab**

**Submitted to:** Sir Shahid Bhatti

**Submitted by:** Abdullah Saleem

**Roll no:** FA24-BCS-141

**Section:** B

**Date:** 1 May,2025

**Submitted to:** Mr. Abdul-Karim

**Submitted by:** Abdullah Saleem

**Roll no:** FA24-BCS-141

**Section:** B

**Date:** 7 Nov,2024

REPORT

**Mistakes Review**

**Missing Base Class for Shared Attributes**

The original code repeated the name attribute in the Owner, Supervisor, and PermitHolder classes.

**Modification:**

A Person**class** was added as a base class. All roles like Owner, Supervisor, PermitHolder now inherit from Person and reusing the name attribute efficiently.

**2. Singleton Not Implemented**

The ParkingSystem class allowed multiple instances to be created .

**Modification:**

A **singleton pattern** was implemented in ParkingSystem. Now only one instance can ever exist. If someone tries to create a new instance the system warns them and returns the existing one.

**3. Auto-Generated IDs Missing**

The original code did not auto-generate unique IDs for owners, permit holders, or parking zones. For example, all owners had the same ID.

**Static counters** were added to generate unique IDs:

**Owner IDs**: Start at 0001 and increment (e.g., 0002, 0003).

**Permit IDs**: Start at 1000 and increment.

**Zone IDs**: Auto-generated as Z1, Z2, etc.

**4. Incorrect Vehicle-Owner Relationship**

The Vehicle class stored the owner’s name as a String instead of an Owner object. This made it impossible to track owner details properly.

**Modification:**  
The Vehicle class now uses an Owner**object** to link a vehicle to its owner. This ensures every vehicle has valid owner information.

**5. Missing Shallow/Deep Copy for Vehicles**

The original code did not support copying vehicles, which was required to test how modifications to owners affect copies.

**Modification:**

Two copy methods were added to Vehicle:

**Shallow Copy**: Shares the same owner object.

**Deep Copy**: Creates a new, independent owner.

**6. Parking Zone Issues:**

Zones could hold more than five vehicles. Duplicate license plates were allowed.

**Modification:**

A **capacity check** ensures zones cannot exceed five vehicles. A duplicate plate check rejects vehicles with existing plates.

**Corrected Code**

**Owner.java**

public class Owner extends Person {

    private static int nextId = 1;

    private String ownerId;

    public Owner(String name) {

        super(name);

        this.ownerId = String.format("%04d", nextId++);

    }

    public String getOwnerId() {

        return ownerId;

    }

    @Override

    public String toString() {

        return "[Name: " + name + ", Owner ID: " + ownerId + "]";

    }

}

**ParkingSystem.java**

import java.util.ArrayList;

public class ParkingSystem {

    private static ParkingSystem instance;

    private String campusName;

    private Supervisor supervisor;

    private ArrayList<ParkingZone> zones;

    private ArrayList<PermitHolder> permitHolders;

    private ParkingSystem(String campusName, Supervisor supervisor) {

        this.campusName = campusName;

        this.supervisor = supervisor;

        this.zones = new ArrayList<>();

        this.permitHolders = new ArrayList<>();

    }

    public static ParkingSystem getInstance(String campusName, Supervisor supervisor) {

        if (instance == null) {

            if (supervisor == null) {

                throw new IllegalArgumentException("Supervisor is required.");

      }

            instance = new ParkingSystem(campusName, supervisor);

    } else {

            System.out.println("Warning: ParkingSystem instance already exists. Returning existing instance.");

    }

        return instance;

    }

    public void addZone(ParkingZone zone) {

        zones.add(zone);

    }

    public void addPermitHolder(PermitHolder ph) {

        permitHolders.add(ph);

    }

    @Override

    public String toString() {

        return "Campus: " + campusName + "\nSupervisor: " + supervisor + "\nZones: " + zones + "\nPermit Holders: " + permitHolders;

    }

}

**ParkingZone.java**

import java.util.ArrayList;

public class ParkingZone {

    private static int zoneCounter = 1;

    private String zoneId;

    private ArrayList<Vehicle> vehicles;

    public ParkingZone() {

        this.zoneId = "Z" + zoneCounter++;

        this.vehicles = new ArrayList<>(5);

    }

    public boolean addVehicle(Vehicle vehicle) {

        if (vehicles.size() >= 5) {

            System.out.println("Error: Zone is full.");

            return false;

        }

        for (Vehicle v : vehicles) {

            if (v.getPlate().equals(vehicle.getPlate())) {

                System.out.println("Error: Duplicate license plate '" + vehicle.getPlate() + "' is not allowed.");

                return false;

            }

        }

        vehicles.add(vehicle);

        return true;

    }

    @Override

    public String toString() {

        return "Zone ID: " + zoneId + ", Vehicles: " + vehicles;

    }

}

**PermitHolder.java**

public class PermitHolder extends Person {

    private static int nextId = 1000;

    private String permitId;

    public PermitHolder(String name) {

        super(name);

        this.permitId = String.valueOf(nextId++);

    }

    public String getPermitId() {

        return permitId;

    }

    @Override

    public String toString() {

        return "Name: " + name + ", Permit ID: " + permitId;

    }

}

**Supervisor.java**

public class Supervisor extends Person {

    private int experience;

    public Supervisor(String name, int experience) {

        super(name);

        this.experience = experience;

    }

    public int getExperience() {

        return experience;

    }

    @Override

    public String toString() {

        return "Name: " + name + ", Experience: " + experience + " years";

    }

}

**Vehicle.java**

public class Vehicle{

    private String plate;

    private String type;

    private Owner owner;

    public Vehicle(String plate, String type, Owner owner) {

        this.plate = plate;

        this.type = type;

        this.owner = owner;

    }

   // Shallow copy

    public Vehicle shallowCopy() {

        return new Vehicle(plate, type, owner);

    }

    // Deep copy

    public Vehicle deepCopy() {

        Owner newOwner = new Owner(owner.getName());

        return new Vehicle(plate, type, newOwner);

    }

    public String getPlate() {

        return plate;

    }

    @Override

    public String toString() {

        return "License Plate: " + plate + ", Type: " + type + ", Owner: " + owner;

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        Supervisor supervisor = new Supervisor("Dr. Ayesha", 5);

        ParkingSystem system = ParkingSystem.getInstance("CUI Lahore", supervisor);

        ParkingSystem secondAttempt = ParkingSystem.getInstance("ShouldNotWork", supervisor);

        ParkingZone zone1 = new ParkingZone();

        ParkingZone zone2 = new ParkingZone();

        Owner owner1 = new Owner("Ali"); // ownerId 0001

        Owner owner2 = new Owner("Zara"); // ownerId 0002

        Vehicle v1 = new Vehicle("LEA123", "Car", owner1);

        Vehicle v2 = new Vehicle("LEB456", "Bike", owner2);

        Vehicle v3 = new Vehicle("LEA123", "Car", owner1);

        zone1.addVehicle(v1);

        zone1.addVehicle(v2); // This will trigger duplicate error

        zone2.addVehicle(v3);

        system.addZone(zone1);

        system.addZone(zone2);

        PermitHolder ph1 = new PermitHolder("Sara");

        PermitHolder ph2 = new PermitHolder("Hassan");

        system.addPermitHolder(ph1);

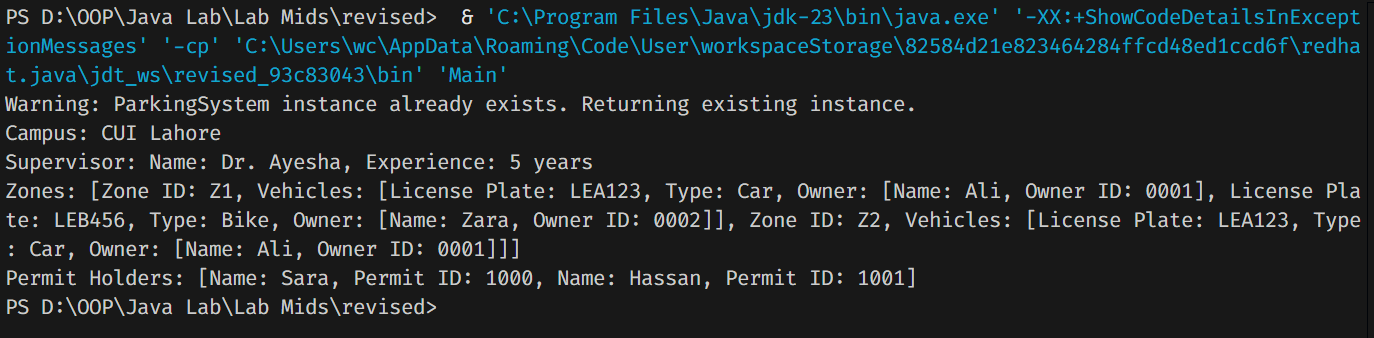
        system.addPermitHolder(ph2);

        System.out.println(system);

    }

}

**Output**

****