**Object Class Methods, Inner Classes Practice Problem**

**Object Modeling & toString()**

public class Vehicle {

private String registrationNo;

private String type;

private double ratePerDay;

public Vehicle(String registrationNo, String type, double ratePerDay) {

this.registrationNo = registrationNo;

this.type = type;

this.ratePerDay = ratePerDay;

}

public String toString() {

return "Vehicle: " + registrationNo + ", Type: " + type + ", Rate: $" + ratePerDay + "/day";

}

public String getRegistrationNo() {

return registrationNo;

}

public String getType() {

return type;

}

public double getRatePerDay() {

return ratePerDay;

}

}

public class VehicleRental {

public static void main(String[] args) {

Vehicle v1 = new Vehicle("MH12AB1234", "Sedan", 1500);

System.out.println(v1);

Vehicle v2 = new Vehicle("DL01XY5678", "SUV", 2000);

System.out.println(v2);

}

}

**equals(),**

**==**

**, and hashCode()**

import java.util.HashSet;

import java.util.Objects;

public class Employee {

private String empCode;

private String name;

public Employee(String empCode, String name) {

this.empCode = empCode;

this.name = name;

}

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Employee employee = (Employee) obj;

return empCode.equals(employee.empCode);

}

public int hashCode() {

return Objects.hash(empCode);

}

public String toString() {

return "Employee: " + empCode + ", Name: " + name;

}

}

public class EmployeeAuth {

public static void main(String[] args) {

Employee e1 = new Employee("BL001", "Ritika");

Employee e2 = new Employee("BL001", "Ritika S.");

System.out.println("e1 == e2: " + (e1 == e2));

System.out.println("e1.equals(e2): " + e1.equals(e2));

HashSet<Employee> employees = new HashSet<>();

employees.add(e1);

employees.add(e2);

System.out.println("HashSet size: " + employees.size());

System.out.println(employees);

}

}

**getClass()**

**Problem: "Payment Gateway"**

public class Payment {

public void pay() {

System.out.println("Generic payment");

}

}

public class CreditCardPayment extends Payment {

public void pay() {

System.out.println("Payment via Credit Card");

}

}

public class WalletPayment extends Payment {

public void pay() {

System.out.println("Payment via Wallet");

}

}

public class PaymentGateway {

public static void main(String[] args) {

Payment[] payments = new Payment[3];

payments[0] = new CreditCardPayment();

payments[1] = new WalletPayment();

payments[2] = new Payment();

for (Payment p : payments) {

System.out.println("Class: " + p.getClass().getSimpleName());

p.pay();

System.out.println();

}

}

}

**clone(), Shallow vs Deep Copy**

class ContactInfo implements Cloneable {

String email;

String phone;

public ContactInfo(String email, String phone) {

this.email = email;

this.phone = phone;

}

public ContactInfo clone() throws CloneNotSupportedException {

return (ContactInfo) super.clone();

}

public String toString() {

return "Email: " + email + ", Phone: " + phone;

}

}

class Student implements Cloneable {

String id;

String name;

ContactInfo contact;

public Student(String id, String name, ContactInfo contact) {

this.id = id;

this.name = name;

this.contact = contact;

}

public Student shallowClone() throws CloneNotSupportedException {

return (Student) super.clone();

}

public Student deepClone() throws CloneNotSupportedException {

Student cloned = (Student) super.clone();

cloned.contact = this.contact.clone();

return cloned;

}

public String toString() {

return "Student ID: " + id + ", Name: " + name + ", " + contact;

}

}

public class Registration {

public static void main(String[] args) throws CloneNotSupportedException {

ContactInfo contact = new ContactInfo("john@email.com", "9876543210");

Student s1 = new Student("S001", "John", contact);

Student s2 = s1.shallowClone();

Student s3 = s1.deepClone();

System.out.println("Original: " + s1);

System.out.println("Shallow: " + s2);

System.out.println("Deep: " + s3);

s1.contact.email = "john.new@email.com";

System.out.println("\nAfter changing original contact:");

System.out.println("Original: " + s1);

System.out.println("Shallow: " + s2);

System.out.println("Deep: " + s3);

}

}

**Member Inner Class**

**Problem: "Hospital Management"**

public class Hospital {

private String name;

public Hospital(String name) {

this.name = name;

}

public class Department {

private String deptName;

public Department(String deptName) {

this.deptName = deptName;

}

public void displayInfo() {

System.out.println("Hospital: " + name + ", Department: " + deptName);

}

}

public Department createDepartment(String deptName) {

return new Department(deptName);

}

}

public class HospitalManagement {

public static void main(String[] args) {

Hospital hospital = new Hospital("City Hospital");

Hospital.Department dept1 = hospital.new Department("Cardiology");

dept1.displayInfo();

Hospital.Department dept2 = hospital.createDepartment("Neurology");

dept2.displayInfo();

Hospital hospital2 = new Hospital("General Hospital");

Hospital.Department dept3 = hospital2.new Department("Emergency");

dept3.displayInfo();

}

}

**Static Nested Class**

**Problem: "App Configuration"**

public class AppConfig {

private String appName;

private static String version = "1.0";

public AppConfig(String appName) {

this.appName = appName;

}

public static class NetworkConfig {

private String host;

private int port;

public NetworkConfig(String host, int port) {

this.host = host;

this.port = port;

}

public void displayConfig() {

System.out.println("Host: " + host);

System.out.println("Port: " + port);

System.out.println("App Version: " + version);

}

}

}

public class AppConfigurator {

public static void main(String[] args) {

AppConfig.NetworkConfig config1 = new AppConfig.NetworkConfig("localhost", 8080);

config1.displayConfig();

System.out.println();

AppConfig.NetworkConfig config2 = new AppConfig.NetworkConfig("192.168.1.1", 3306);

config2.displayConfig();

}

}