

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

**Object Modeling & toString()**

Problem: "Vehicle Rental System"

| 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 getRegistrationNo() {  return registrationNo;  }   public String getType() {  return type;  }   public double getRatePerDay() {  return ratePerDay;  }   @Override  public String toString() {  return "Vehicle: " + registrationNo + ", Type: " + type + ", Rate: $" + ratePerDay + "/day";  } }  public class VehicleRental {  public static void main(String[] args) {  Vehicle v1 = new Vehicle("MH12AB1234", "Sedan", 1500);  System.out.println(v1);  Vehicle v2 = new Vehicle("MH14XY5678", "SUV", 2000);  System.out.println(v2);  System.out.println(v1.equals(v2));  } } |
| --- |

1



**equals(),==, and hashCode()**

Problem: "Employee Authentication System"

| // File: EmployeeAuth.java import java.util.HashSet;  public class Employee {  private String empCode;  private String name;   public Employee(String empCode, String name) {  this.empCode = empCode;  this.name = name;  }   @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (obj == null || getClass() != obj.getClass()) return false;  Employee e = (Employee) obj;  return empCode.equals(e.empCode);  }   @Override  public int hashCode() {  return empCode.hashCode();  }   @Override  public String toString() {  return "Employee Code: " + 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);  System.out.println(e1.equals(e2));  HashSet<Employee> set = new HashSet<>();  set.add(e1);  set.add(e2);  System.out.println(set);  } } |
| --- |

2

**getClass()**

Problem: "Payment Gateway"



| class Payment {  public void pay() {  System.out.println("Generic payment");  } }  class CreditCardPayment extends Payment {  public void pay() {  System.out.println("Paid using Credit Card");  } }  class WalletPayment extends Payment {  public void pay() {  System.out.println("Paid using Wallet");  } }  public class PaymentGateway {  public static void main(String[] args) {  Payment[] payments = { new CreditCardPayment(), new WalletPayment() };  for (Payment p : payments) {  System.out.println(p.getClass().getSimpleName());  p.pay();  }  } } |
| --- |

3



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

Problem: "Course Registration System"

| // File: Registration.java class ContactInfo implements Cloneable {  String email;  String phone;   public ContactInfo(String email, String phone) {  this.email = email;  this.phone = phone;  }   @Override  protected Object clone() throws CloneNotSupportedException {  return super.clone();  }   @Override  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;  }   protected Student shallowCopy() throws CloneNotSupportedException {  return (Student) super.clone();  }   protected Student deepCopy() throws CloneNotSupportedException {  Student cloned = (Student) super.clone();  cloned.contact = (ContactInfo) contact.clone();  return cloned;  }   @Override  public String toString() {  return "ID: " + id + ", Name: " + name + ", " + contact;  } }  public class Registration {  public static void main(String[] args) throws CloneNotSupportedException {  ContactInfo c1 = new ContactInfo("student@mail.com", "9999999999");  Student s1 = new Student("S101", "Aarav", c1);   Student shallow = s1.shallowCopy();  Student deep = s1.deepCopy();   s1.contact.email = "changed@mail.com";   System.out.println("Original: " + s1);  System.out.println("Shallow Copy: " + shallow);  System.out.println("Deep Copy: " + deep);  } } |
| --- |

**Member Inner Class**

Problem: "Hospital Management"

| // File: Hospital.java 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 h = new Hospital("CityCare Hospital");  Hospital.Department d1 = h.createDepartment("Cardiology");  Hospital.Department d2 = h.createDepartment("Neurology");  d1.displayInfo();  d2.displayInfo();  } } |
| --- |

**Static Nested Class**

Problem: "App Configuration"

| // File: AppConfig.java public class AppConfig {  private String appName = "MyApplication";   public static class NetworkConfig {  private String host;  private int port;   public NetworkConfig(String host, int port) {  this.host = host;  this.port = port;  }   public void display() {  System.out.println("Network Config -> Host: " + host + ", Port: " + port);  }  } }  public class AppConfigurator {  public static void main(String[] args) {  AppConfig.NetworkConfig config = new AppConfig.NetworkConfig("localhost", 8080);  config.display();  } } |
| --- |

5

**Local Inner Class** Problem: "Voting System"



| public class VotingSystem {  public void processVote(String voterId, String candidate) {  class VoteValidator {  public boolean validate(String id) {  return id != null && id.startsWith("V") && id.length() == 5;  }  }   VoteValidator validator = new VoteValidator();  if (validator.validate(voterId))  System.out.println("Vote accepted for " + candidate + " by voter " + voterId);  else  System.out.println("Invalid voter ID: " + voterId);  }   public static void main(String[] args) {  VotingSystem vs = new VotingSystem();  vs.processVote("V1234", "Alice");  vs.processVote("12345", "Bob");  vs.processVote("V12", "Charlie");  } } |
| --- |

**Anonymous Inner Class**

Problem: "Notification Service"

| interface Notifier {  void send(String message); }  public class Service {  public void triggerAlert() {  Notifier notifier = new Notifier() {  public void send(String message) {  System.out.println("Alert: " + message);  }  };  notifier.send("System overload detected!");  }   public static void main(String[] args) {  new Service().triggerAlert();  } } |
| --- |

6