**OOPJ Lab Assignment 2**

Problem 1: Counter for Cups Scenario: You are keeping track of how many cups of tea are prepared in your home.

Requirements: 1. Create a class TeaCup with instance variable: teaType (String). 2. Create a static variable totalCups to count all cups created. 3. Constructor should initialize teaType and increment totalCups. 4. Create getter for teaType. 5. Create a static method showTotalCups() to print total cups. Input Example: Cup1: teaType = "Masala Tea" Cup2: teaType = "Green Tea" Cup3: teaType = "Ginger Tea" Expected Output: Cup1 type: Masala Tea Cup2 type: Green Tea Cup3 type: Ginger Tea Total cups made: 3

class TeaCup

{

private String teaType;

private static int totalCups=0;

TeaCup(String teaType)

{

this.teaType=teaType;

totalCups++;

}

public String getData()

{

return teaType;

}

static void showTotalCups()

{

System.out.println("Total cups made:"+totalCups);

}

}

class TotalCup

{

public static void main(String[]args)

{

TeaCup t1=new TeaCup("Masala Tea");

TeaCup t2=new TeaCup("Green Tea");

TeaCup t3=new TeaCup("Ginger Tea");

System.out.println("Cup1 type:"+t1.getData());

System.out.println("Cup2 type:"+t2.getData());

System.out.println("Cup3 type:"+t3.getData());

TeaCup.showTotalCups();

}

}

Problem 2: Simple Mobile Tracker Scenario: A shop wants to count how many mobiles are added to their inventory. Requirements: 1. Create a class Mobile with instance variable: model (String). 2. Create a static variable totalMobiles to count total mobiles added. 3. Constructor should initialize model and increment totalMobiles. 4. Create a getter for model. 5. Create a static method showTotalMobiles() to print total mobiles. Input Example: Mobile1: model = "Samsung Galaxy M32" Mobile2: model = "Redmi Note 12" Expected Output: Mobile1 model: Samsung Galaxy M32 Mobile2 model: Redmi Note 12

Total mobiles in stock: 2

class Mobile

{

private String model;

private static int totalMobiles=0;

Mobile(String model)

{

this.model=model;

totalMobiles++;

}

String getData()

{

return model;

}

static void showToatlMobiles()

{

System.out.println("Total mobiles in stock:"+ totalMobiles);

}

}

class MobileTracker

{

public static void main(String[] args)

{

Mobile m1=new Mobile("Samsung Galaxy M32");

Mobile m2=new Mobile("Redmi Note 12");

System.out.println("Mobile1 model:"+m1.getData());

System.out.println("Mobile2 model:"+m2.getData());

Mobile.showToatlMobiles();

}

}

Problem 3: Library Book Tracker Scenario: A library in Delhi wants to track how many books are issued in total and details of each book. Requirements: 1. Create a Book class with instance variables: title (String), author (String), issued (boolean). 2. Create static variable totalIssuedBooks to keep track of the total number of books issued. 3. Create a constructor to initialize the book details. 4. Create getters and setters for all instance variables. 5. Create a static method showTotalIssued() to print total issued books. 6. Write a main class to create 3 books, issue some of them (issued = true), and show total issued books. Input Example: Book1: "Harry Potter", Author: "J.K. Rowling", Issued: true Book2: "Five Point Someone", Author: "Chetan Bhagat", Issued: false Book3: "Rich Dad Poor Dad", Author: "Robert Kiyosaki", Issued: true Expected Output: Book1 issued? true Book2 issued? false Book3 issued? true Total books issued: 2

class Book

{

private String title;

private String author;

private boolean isissued;

private static int totalIssuedBooks=0;

Book(String title,String author,boolean isissued)

{

this.title=title;

this.author=author;

this.isissued=isissued;

if(isissued)totalIssuedBooks++;

}

String getTitle()

{

return title;

}

String getAuthor()

{

return author;

}

Boolean isIssued()

{

return isissued;

}

void setTitle(String title)

{

this.title=title;

}

void setAuthor(String author)

{

this.author=author;

}

public void setIssued(boolean isissued) {

if (this.isissued != isissued) {

if (isissued) totalIssuedBooks++;

else totalIssuedBooks--;

this.isissued = isissued;

}

}

static void showTotalIssued()

{

System.out.println("Toatl books issued:"+totalIssuedBooks);

}

}

class BookTracker

{

public static void main(String[]args)

{

Book b1=new Book("Harry Potter","J.K.Rowling",true);

Book b2=new Book("Fice Point Someone","Chetan Bhagat",false);

Book b3=new Book("Rich Dad Poor Dad","Robert Kiyosaki",true);

System.out.println("Book1 issued? "+b1.isIssued());

System.out.println("Book2 issued? "+b2.isIssued());

System.out.println("Book3 issued? "+b3.isIssued());

Book.showTotalIssued();

}

}

Problem 4: Employee Salary Manager Scenario: A company in Bengaluru wants to maintain employee details and give a bonus to employees who have worked more than 5 years. Requirements: 1. Create a class Employee with instance variables: name (String), salary (double), yearsOfService (int). 2. Create static variable totalEmployees to store the number of employees created. 3. Constructor should initialize all instance variables and increment totalEmployees. 4. Create getters and setters for all instance variables. 5. Create a method calculateBonus() that returns 5% of salary if yearsOfService > 5, otherwise 0. 6. Create a static method showTotalEmployees() to print total employees created. 7. Write a main class to create 3 employees, print their bonuses, and print total employees. Input Example: Employee1: Name: "Ravi", Salary: 150000, Years of Service: 6 Employee2: Name: "Anita", Salary: 120000, Years of Service: 3 Employee3: Name: "Suresh", Salary: 100000, Years of Service: 5 Expected Output: Employee Ravi Bonus: 7500.0 Employee Anita Bonus: 0.0 Employee Suresh Bonus: 0.0 Total employees: 3

class Employee

{

private String name;

private double salary;

private int yearOfService;

private static int totalEmployees=0;

Employee(String name,double salary,int yearOfService)

{

this.name=name;

this.salary=salary;

this.yearOfService=yearOfService;

totalEmployees++;

}

String getName()

{

return name;

}

double getSalary()

{

return salary;

}

int getYearOfService()

{

return yearOfService;

}

public void setName(String name)

{

this.name = name;

}

public void setSalary(double salary)

{

this.salary = salary;

}

public void setYearsOfService(int yearsOfService)

{

this.yearOfService = yearOfService;

}

public double calculateBonus() {

if (yearOfService > 5) {

return salary \* 0.05;

}

return 0.0;

}

public static void showTotalEmployees() {

System.out.println("Total employees: " + totalEmployees);

}

}

public class Company {

public static void main(String[] args) {

Employee e1 = new Employee("Krushna",50000, 6);

Employee e2 = new Employee("Devesh", 20000, 9);

Employee e3 = new Employee("Darshan", 450000, 5);

System.out.println("Employee " + e1.getName() + " Bonus: " + e1.calculateBonus());

System.out.println("Employee " + e2.getName() + " Bonus: " + e2.calculateBonus());

System.out.println("Employee " + e3.getName() + " Bonus: " + e3.calculateBonus());

Employee.showTotalEmployees();

}

}

Problem 5: Student Marks Calculator Scenario: A school in Mumbai wants to calculate marks of students and also maintain total students in the class. Requirements: 1. Create a class Student with instance variables: name (String), marks (int). 2. Create static variable totalStudents to count total number of students. 3. Constructor to initialize student details and increment totalStudents. 4. Getter and Setter for marks. 5. Method isPassed() returns true if marks >= 35, false otherwise. 6. Static method showTotalStudents() prints total students. 7. In main class, create 3 students, check if they passed, and show total students. Input Example: Student1: Name: "Rahul", Marks: 78 Student2: Name: "Pooja", Marks: 34 Student3: Name: "Amit", Marks: 65 Expected Output: Student Rahul Passed? true Student Pooja Passed? false Student Amit Passed? true Total students: 3

class Student

{

private String name;

private int marks;

private static int totalStudents=0;

Student(String name,int marks)

{

this.name=name;

this.marks=marks;

totalStudents++;

}

public String getName()

{

return name;

}

public int getMarks()

{

return marks;

}

public void setMarks(int marks)

{

this.marks=marks;

}

public boolean isPassed()

{

return marks>=35;

}

public static void showTotalStudents()

{

System.out.println("Total students:"+totalStudents);

}

}

class School

{

public static void main(String[]args)

{

Student s1=new Student("Krushna",78);

Student s2=new Student("Rahul",34);

Student s3=new Student("Pooja",45);

System.out.println("Student " + s1.getName() + " Passed? " + s1.isPassed());

System.out.println("Student " + s2.getName() + " Passed? " + s2.isPassed());

System.out.println("Student " + s3.getName() + " Passed? " + s3.isPassed());

Student.showTotalStudents();

}

}

Problem 6: Indian Railway Ticket Booking Scenario: You are building a mini ticket booking system. A passenger can book a ticket either by giving name and age or name, age, and seat type. The system should also count the total tickets booked using a static counter. Tasks: 1. Create a Passenger class. 2. Implement two constructors (constructor overloading): Constructor 1 → Passenger(String name, int age) Constructor 2 → Passenger(String name, int age, String seatType) 3. Use a static counter to keep track of total passengers booked. 4. Print passenger details and total passengers. Input Example: Passenger1: "Ravi", 25 Passenger2: "Anita", 30, "AC Sleeper" Passenger3: "Suresh", 40 Expected Output: Passenger1: Name: Ravi, Age: 25, Seat: General Passenger2: Name: Anita, Age: 30, Seat: AC Sleeper Passenger3: Name: Suresh, Age: 40, Seat: General Total Passengers Booked: 3

class Passenger {

private String name;

private int age;

private String seatType;

private static int totalPassengers = 0;

public Passenger(String name, int age) {

this.name = name;

this.age = age;

this.seatType = "General";

totalPassengers++;

}

public Passenger(String name, int age, String seatType) {

this.name = name;

this.age = age;

this.seatType = seatType;

totalPassengers++;

}

public void showDetails() {

System.out.println("Name: " + name + ", Age: " + age + ", Seat: " + seatType);

}

public static void showTotalPassengers() {

System.out.println("Total Passengers Booked: " + totalPassengers);

}

}

public class Railway {

public static void main(String[] args) {

Passenger p1 = new Passenger("Ravi", 25);

Passenger p2 = new Passenger("Anita", 30, "AC Sleeper");

Passenger p3 = new Passenger("Suresh", 40);

System.out.print("Passenger1: ");

p1.showDetails();

System.out.print("Passenger2: ");

p2.showDetails();

System.out.print("Passenger3: ");

p3.showDetails();

Passenger.showTotalPassengers();

}

}

Problem 7: Indian Movie Ticket Booking Scenario: A cinema hall offers Normal and Premium tickets. A customer can book just name or name with ticket type. Keep track of total tickets sold using a static counter. Tasks: 1. Create a Customer class. 2. Implement two constructors: Constructor 1 → Customer(String name) Constructor 2 → Customer(String name, String ticketType) 3. Static counter to track tickets sold. 4. Print customer details and total tickets sold. Input Example: Customer1: "Rahul" Customer2: "Pooja", "Premium" Customer3: "Amit" Expected Output: Customer1: Name: Rahul, Ticket: Normal Customer2: Name: Pooja, Ticket: Premium Customer3: Name: Amit, Ticket: Normal Total Tickets Sold: 3

class Customer {

private String name;

private String ticketType;

private static int totalTickets = 0;

Customer(String name) {

this.name = name;

this.ticketType = "Normal";

totalTickets++;

}

Customer(String name, String ticketType) {

this.name = name;

this.ticketType = ticketType;

totalTickets++;

}

public void showDetails() {

System.out.println("Name: " + name + ", Ticket: " + ticketType);

}

public static void showTotalTickets() {

System.out.println("Total Tickets Sold: " + totalTickets);

}

}

public class MovieBooking {

public static void main(String[] args) {

Customer c1 = new Customer("Rahul");

Customer c2 = new Customer("Pooja", "Premium");

Customer c3 = new Customer("Amit");

System.out.print("Customer1: ");

c1.showDetails();

System.out.print("Customer2: ");

c2.showDetails();

System.out.print("Customer3: ");

c3.showDetails();

Customer.showTotalTickets();

}

}

Problem 8: Bank Account Initialization Scenario: A bank wants to initialize the interest rate for all accounts once when the system starts. Each account has account holder name, balance, and interest rate. Students should practice static blocks for initialization and setters/getters to modify and access account details. Tasks: 1. Create a BankAccount class. 2. Use a static block to initialize interest rate to 4%. 3. Create instance variables: name (String) and balance (double). 4. Create setters and getters for name and balance. 5. Print account details including interest rate. Input Example: Account1: Name="Rohit", Balance=5000 Account2: Name="Priya", Balance=15000 Expected Output: Bank Interest Rate Initialized: 4.0% Account1: Name=Rohit, Balance=5000.0, Interest Rate=4.0% Account2: Name=Priya, Balance=15000.0, Interest Rate=4.0%

class BankAccount {

private String name;

private double balance;

private static double interestRate;

static {

interestRate = 4.0;

System.out.println("Bank Interest Rate Initialized: " + interestRate + "%");

}

BankAccount(String name, double balance) {

this.name = name;

this.balance = balance;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getBalance() {

return balance;

}

public void setBalance(double balance) {

this.balance = balance;

}

public static double getInterestRate() {

return interestRate;

}

public static void setInterestRate(double interestRate) {

BankAccount.interestRate = interestRate;

}

public void showDetails() {

System.out.println("Name=" + name + ", Balance=" + balance + ", Interest Rate=" + interestRate + "%");

}

}

class Bank {

public static void main(String[] args) {

BankAccount acc1 = new BankAccount("Rohit", 5000);

BankAccount acc2 = new BankAccount("Priya", 15000);

System.out.print("Account1: ");

acc1.showDetails();

System.out.print("Account2: ");

acc2.showDetails();

}

}

Problem 9: School Fee System Scenario: A school wants to initialize the tuition fee for all students once at program start. Each student has name and class. Use static blocks to set the fee and setters/getters to update/access student information. Tasks: 1. Create a Student class. 2. Use a static block to initialize tuitionFee to 30000. 3. Create instance variables: name (String) and className (String). 4. Create setters and getters for name and className. 5. Print student details including tuition fee. Input Example: Student1: Name="Anjali", Class="10th" Student2: Name="Vikram", Class="12th" Expected Output: School Tuition Fee Initialized: 30000 Student1: Name=Anjali, Class=10th, Tuition Fee=30000 Student2: Name=Vikram, Class=12th, Tuition Fee=30000

class Student {

private String name;

private String className;

private static int tuitionFee;

static {

tuitionFee = 30000;

System.out.println("School Tuition Fee Initialized: " + tuitionFee);

}

Student(String name, String className) {

this.name = name;

this.className = className;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getClassName() {

return className;

}

public void setClassName(String className) {

this.className = className;

}

public static int getTuitionFee() {

return tuitionFee;

}

public static void setTuitionFee(int fee) {

tuitionFee = fee;

}

public void showDetails() {

System.out.println("Name=" + name + ", Class=" + className + ", Tuition Fee=" + tuitionFee);

}

}

class SchoolFeeSystem {

public static void main(String[] args) {

Student s1 = new Student("Anjali", "10th");

Student s2 = new Student("Vikram", "12th");

System.out.print("Student1: ");

s1.showDetails();

System.out.print("Student2: ");

s2.showDetails();

}

}

Problem 10: Student Marks Checker Scenario: Create a Student class with rollNo, name, and marks. ● Use a parameterized constructor to initialize all fields. ● Create a getter and setter for marks. ● In main, create one student, update marks using setter, and print student details.

class Student {

private int rollNo;

private String name;

private int marks;

Student(int rollNo, String name, int marks) {

this.rollNo = rollNo;

this.name = name;

this.marks = marks;

}

public int getMarks() {

return marks;

}

public void setMarks(int marks) {

this.marks = marks;

}

public void showDetails() {

System.out.println("Roll No: " + rollNo + ", Name: " + name + ", Marks: " + marks);

}

}

public class StudentMarksCheckUp {

public static void main(String[] args) {

Student s1 = new Student(101, "Krushna", 50);

s1.setMarks(85);

s1.showDetails();

}

}

Problem 11: Student Grade Calculator Scenario: Extend previous problem. Add method calculateGrade() which returns: ● "A" if marks ≥ 80 ● "B" if marks ≥ 60 ● "C" if marks ≥ 40 ● "Fail" otherwise ● Create 2 students, print marks and grades.

class Student {

private int rollNo;

private String name;

private int marks;

Student(int rollNo, String name, int marks) {

this.rollNo = rollNo;

this.name = name;

this.marks = marks;

}

public int getMarks() {

return marks;

}

public void setMarks(int marks) {

this.marks = marks;

}

public String calculateGrade() {

if (marks >= 80) {

return "A";

} else if (marks >= 60) {

return "B";

} else if (marks >= 40) {

return "C";

} else {

return "Fail";

}

}

public void showDetails() {

System.out.println("Roll No: " + rollNo + ", Name: " + name +

", Marks: " + marks + ", Grade: " + calculateGrade());

}

}

public class StudentGrade {

public static void main(String[] args) {

Student s1 = new Student(101, "Krushna", 85);

Student s2 = new Student(102, "Devesh", 75);

s1.showDetails();

s2.showDetails();

}

}

Problem 12: Bank Account Basic Info Scenario: Create BankAccount class with accountHolder and balance. ● Use parameterized constructor to initialize account. ● Create getter and setter for balance. ● In main, create one account and display details.

class BankAccount {

private String accountHolder;

private double balance;

BankAccount(String accountHolder, double balance) {

this.accountHolder = accountHolder;

this.balance = balance;

}

public double getBalance() {

return balance;

}

public void setBalance(double balance) {

this.balance = balance;

}

public void showDetails() {

System.out.println("Account Holder: " + accountHolder +

", Balance: " + balance);

}

}

public class BankDetail {

public static void main(String[] args) {

BankAccount acc1 = new BankAccount("Krushna", 10000.50);

acc1.showDetails();

}

}

Problem 13: Bank Deposit & Withdrawal Scenario: Extend previous problem. Add methods: ● deposit(double amount) → adds to balance ● withdraw(double amount) → subtracts from balance ● Create two accounts, perform deposit/withdraw, display updated balance

class BankAccount {

private String accountHolder;

private double balance;

BankAccount(String accountHolder, double balance) {

this.accountHolder = accountHolder;

this.balance = balance;

}

public double getBalance() {

return balance;

}

public void setBalance(double balance) {

this.balance = balance;

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

System.out.println(accountHolder + " deposited: " + amount);

} else {

System.out.println("Invalid deposit amount!");

}

}

public void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

System.out.println(accountHolder + " withdrew: " + amount);

} else {

System.out.println("Insufficient balance or invalid withdrawal!");

}

}

public void showDetails() {

System.out.println("Account Holder: " + accountHolder +

", Balance: " + balance);

}

}

public class Bank1 {

public static void main(String[] args) {

BankAccount acc1 = new BankAccount("Gaurav", 10000);

BankAccount acc2 = new BankAccount("Vaibhav", 5000);

acc1.deposit(2000);

acc1.withdraw(3000);

acc2.deposit(1500);

acc2.withdraw(7000);

acc1.showDetails();

acc2.showDetails();

}

}

Problem 14: Bank Name Display Scenario: Add a static variable bankName = "CDAC Bank" and static method displayBankName() to BankAccount. ● Call displayBankName() from main. ● Create one account to verify instance creation.

class BankAccount {

private String accountHolder;

private double balance;

static String bankName = "CDAC Bank";

public BankAccount(String accountHolder, double balance) {

this.accountHolder = accountHolder;

this.balance = balance;

}

public double getBalance() {

return balance;

}

public void setBalance(double balance) {

this.balance = balance;

}

public static void displayBankName() {

System.out.println("Bank Name: " + bankName);

}

public void showDetails() {

System.out.println("Account Holder: " + accountHolder +

", Balance: " + balance);

}

}

public class Bank2 {

public static void main(String[] args) {

BankAccount.displayBankName();

BankAccount acc1 = new BankAccount("Roshan", 10000);

acc1.showDetails();

}

}

Problem 15: Employee Auto-ID Generator Scenario: Create Employee class with id, name, basicSalary. ● Add static counter starting from 1001 for IDs. ● Default constructor → name = "Unknown", salary = 20000 ● Parameterized constructor → accept name and salary ● Getter for all variables ● Create 2 employees and display their IDs, names, salary

class Employee {

private int id;

private String name;

private double basicSalary;

private static int counter = 1001;

public Employee() {

this.id = counter++;

this.name = "Unknown";

this.basicSalary = 20000;

}

public Employee(String name, double basicSalary) {

this.id = counter++;

this.name = name;

this.basicSalary = basicSalary;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public double getBasicSalary() {

return basicSalary;

}

}

class Emp {

public static void main(String[] args) {

Employee e1 = new Employee();

Employee e2 = new Employee("Amit", 35000);

System.out.println("Employee 1 -> ID: " + e1.getId() + ", Name: " + e1.getName() + ", Salary: " + e1.getBasicSalary());

System.out.println("Employee 2 -> ID: " + e2.getId() + ", Name: " + e2.getName() + ", Salary: " + e2.getBasicSalary());

}

}

Problem 16: Employee Net Salary Scenario: Extend previous problem. Add method calculateNetSalary(): ● Add 10% HRA, 5% DA, deduct 2% PF from basicSalary ● Print net salary for 2 employees

class Employee {

private int id;

private String name;

private double basicSalary;

private static int counter = 1001;

public Employee() {

this.id = counter++;

this.name = "Unknown";

this.basicSalary = 20000;

}

public Employee(String name, double basicSalary) {

this.id = counter++;

this.name = name;

this.basicSalary = basicSalary;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public double getBasicSalary() {

return basicSalary;

}

public double calculateNetSalary() {

double hra = 0.10 \* basicSalary;

double da = 0.05 \* basicSalary;

double pf = 0.02 \* basicSalary;

return basicSalary + hra + da - pf;

}

public void showDetails() {

System.out.println("ID: " + id + ", Name: " + name +

", Basic Salary: " + basicSalary +

", Net Salary: " + calculateNetSalary());

}

}

class EmpNetSalary {

public static void main(String[] args) {

Employee e1 = new Employee();

Employee e2 = new Employee("Amit", 35000);

e1.showDetails();

e2.showDetails();

}

}

Problem 17: Library Book Addition Scenario: Create Book class with bookId, title, author. ● Constructor + Getters/Setters ● Create Library class with libraryName and static totalBooks ● Method addBook(Book b) → increments totalBooks ● Method displayTotalBooks() → prints totalBooks ● Add 2 books to library and display total books

class Book {

private int bookId;

private String title;

private String author;

public Book(int bookId, String title, String author) {

this.bookId = bookId;

this.title = title;

this.author = author;

}

public int getBookId() {

return bookId;

}

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

public void setBookId(int bookId) {

this.bookId = bookId;

}

public void setTitle(String title) {

this.title = title;

}

public void setAuthor(String author) {

this.author = author;

}

}

class Library {

private String libraryName;

private static int totalBooks = 0;

public Library(String libraryName) {

this.libraryName = libraryName;

}

public void addBook(Book b) {

totalBooks++;

System.out.println("Book Added: " + b.getTitle() + " by " + b.getAuthor());

}

public void displayTotalBooks() {

System.out.println("Total Books in " + libraryName + ": " + totalBooks);

}

}

class LibraryBook {

public static void main(String[] args) {

Library lib = new Library("City Library");

Book b1 = new Book(101, "Java Basics", "James Gosling");

Book b2 = new Book(102, "Python Fundamentals", "Guido van Rossum");

lib.addBook(b1);

lib.addBook(b2);

lib.displayTotalBooks();

}

}

Problem 18: Vehicle Registration – Static Counter Scenario: Scenario: Create Vehicle class with regNo, ownerName, vehicleType. ● Static variable: vehicleCount ● Constructor → auto-generate regNo as "MH-2025-" + vehicleCount ● Getter methods for all fields ● Create 2 vehicles, display registration details ● Getter methods for all fields ● Create 2 vehicles, display registration details

class Vehicle {

private String regNo;

private String ownerName;

private String vehicleType;

private static int vehicleCount = 0;

Vehicle(String ownerName, String vehicleType) {

vehicleCount++;

this.regNo = "MH-2025-" + vehicleCount;

this.ownerName = ownerName;

this.vehicleType = vehicleType;

}

public String getRegNo() {

return regNo;

}

public String getOwnerName() {

return ownerName;

}

public String getVehicleType() {

return vehicleType;

}

}

class VehicleRegister{

public static void main(String[] args) {

Vehicle v1 = new Vehicle("Ramesh", "Car");

Vehicle v2 = new Vehicle("Suresh", "Bike");

System.out.println("Vehicle 1 -> RegNo: " + v1.getRegNo() +

", Owner: " + v1.getOwnerName() +

", Type: " + v1.getVehicleType());

System.out.println("Vehicle 2 -> RegNo: " + v2.getRegNo() +

", Owner: " + v2.getOwnerName() +

", Type: " + v2.getVehicleType());

}

}

Problem 19: Vehicle Registration – Static Block

Scenario:

Add a static block to Vehicle class:

● Print "Welcome to CDAC Vehicle Registration Portal" when class loads

● Verify that the message prints only once when multiple vehicles are created

class Vehicle {

private String regNo;

private String ownerName;

private String vehicleType;

private static int vehicleCount = 0;

static {

System.out.println("Welcome to CDAC Vehicle Registration Portal");

}

Vehicle(String ownerName, String vehicleType) {

vehicleCount++;

this.regNo = "MH-2025-" + vehicleCount;

this.ownerName = ownerName;

this.vehicleType = vehicleType;

}

public String getRegNo() {

return regNo;

}

public String getOwnerName() {

return ownerName;

}

public String getVehicleType() {

return vehicleType;

}

}

class VehicleType {

public static void main(String[] args) {

Vehicle v1 = new Vehicle("Ramesh", "Car");

Vehicle v2 = new Vehicle("Suresh", "Bike");

System.out.println("Vehicle 1 -> RegNo: " + v1.getRegNo() +

", Owner: " + v1.getOwnerName() +

", Type: " + v1.getVehicleType());

System.out.println("Vehicle 2 -> RegNo: " + v2.getRegNo() +

", Owner: " + v2.getOwnerName() +

", Type: " + v2.getVehicleType());

}

}

Problem 20: Ticket Booking System

Question:

Create a class Ticket with:

● passengerName (instance)

● ticketNo (instance, auto-generated using a static counter starting from 5001)

● Constructor to accept passengerName

● Method displayTicket() to show ticket details

Task:

Create 3 tickets and display their details.

Sample Input:

Passenger 1: Rahul

Passenger 2: Priya

Passenger 3: Amit

Sample Output:

Ticket No: 5001, Passenger: Rahul

Ticket No: 5002, Passenger: Priya

Ticket No: 5003, Passenger: Amit

class Ticket {

private String passengerName;

private int ticketNo;

private static int counter = 5000;

Ticket(String passengerName) {

this.passengerName = passengerName;

counter++;

this.ticketNo = counter;

}

public void displayTicket() {

System.out.println("Ticket No: " + ticketNo + ", Passenger: " + passengerName);

}

}

class TicketBookSystem {

public static void main(String[] args) {

Ticket t1 = new Ticket("Rahul");

Ticket t2 = new Ticket("Priya");

Ticket t3 = new Ticket("Amit");

t1.displayTicket();

t2.displayTicket();

t3.displayTicket();

}

}