**WEEK 3 – LAB PROBLEM**

Q1. Write a program to create a Bank Account management system without using built-in collection classes

Soln.

public class BankAccountSystem {

static class BankAccount {

private String accountNumber;

private String accountHolderName;

private double balance;

private static int totalAccounts = 0;

public BankAccount(String accountHolderName, double initialDeposit) {

if (initialDeposit < 0) {

throw new IllegalArgumentException("Initial deposit cannot be negative");

}

this.accountHolderName = accountHolderName;

this.accountNumber = generateAccountNumber();

this.balance = initialDeposit;

totalAccounts++;

}

public void deposit(double amount) {

if (amount <= 0) {

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

return;

}

balance += amount;

System.out.println("Deposited Rs." + amount + " into " + accountNumber);

}

public void withdraw(double amount) {

if (amount <= 0) {

System.out.println("Invalid withdrawal amount");

return;

}

if (amount > balance) {

System.out.println("Insufficient funds in " + accountNumber);

return;

}

balance -= amount;

System.out.println("Withdrawn Rs." + amount + " from " + accountNumber);

}

public double checkBalance() {

return balance;

}

public void displayAccountInfo() {

System.out.println("----- Account Info -----");

System.out.println("Account Number: " + accountNumber);

System.out.println("Holder Name : " + accountHolderName);

System.out.println("Balance : Rs." + balance);

System.out.println("------------------------");

}

public static int getTotalAccounts() {

return totalAccounts;

}

private static String generateAccountNumber() {

return String.format("ACC%03d", totalAccounts + 1);

}

}

public static void main(String[] args) {

BankAccount[] accounts = new BankAccount[3];

accounts[0] = new BankAccount("Alice", 5000);

accounts[1] = new BankAccount("Bob", 3000);

accounts[2] = new BankAccount("Charlie", 7000);

accounts[0].deposit(2000);

accounts[1].withdraw(1000);

accounts[2].withdraw(8000); // insufficient funds

for (BankAccount acc : accounts) {

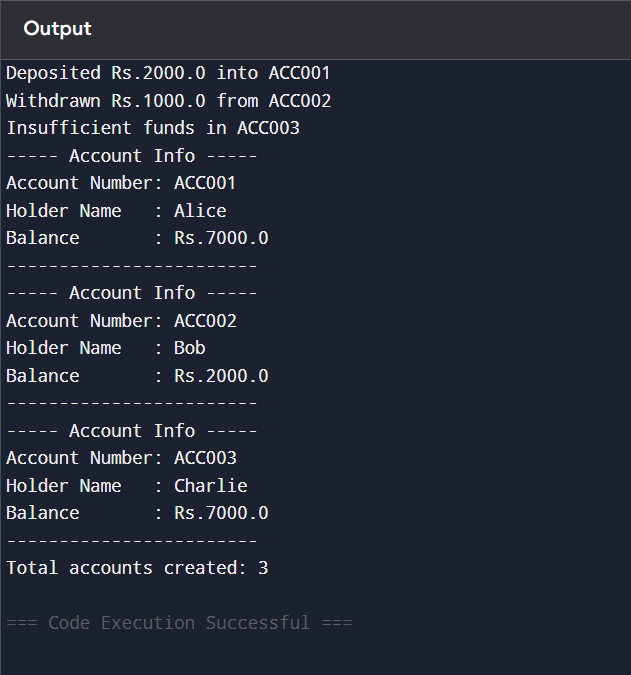
acc.displayAccountInfo();

}

System.out.println("Total accounts created: " + BankAccount.getTotalAccounts());

}

}



Q2. Write a program to create a Library Book management system, demonstrating object relationships

Soln.

public class LibrarySystem {

static class Book {

private String bookId;

private String title;

private String author;

private boolean isAvailable;

private static int totalBooks = 0;

private static int availableBooks = 0;

public Book(String title, String author) {

this.bookId = generateBookId();

this.title = title;

this.author = author;

this.isAvailable = true;

totalBooks++;

availableBooks++;

}

public void issueBook() {

if (isAvailable) {

isAvailable = false;

availableBooks--;

System.out.println("Book " + bookId + " issued.");

} else {

System.out.println("Book " + bookId + " is already issued.");

}

}

public void returnBook() {

if (!isAvailable) {

isAvailable = true;

availableBooks++;

System.out.println("Book " + bookId + " returned.");

} else {

System.out.println("Book " + bookId + " was not issued.");

}

}

public void displayBookInfo() {

System.out.println("Book ID : " + bookId);

System.out.println("Title : " + title);

System.out.println("Author : " + author);

System.out.println("Available : " + isAvailable);

System.out.println("-----------------------------");

}

public boolean isAvailable() { return isAvailable; }

public String getBookId() { return bookId; }

private static String generateBookId() {

return String.format("B%03d", totalBooks + 1);

}

public static int getTotalBooks() { return totalBooks; }

public static int getAvailableBooks() { return availableBooks; }

}

static class Member {

private String memberId;

private String memberName;

private String[] booksIssued;

private int bookCount;

private static int totalMembers = 0;

public Member(String memberName) {

this.memberId = generateMemberId();

this.memberName = memberName;

this.booksIssued = new String[5]; // limit: 5 books

this.bookCount = 0;

totalMembers++;

}

public void borrowBook(Book book) {

if (book.isAvailable() && bookCount < booksIssued.length) {

book.issueBook();

booksIssued[bookCount] = book.getBookId();

bookCount++;

System.out.println(memberName + " borrowed " + book.getBookId());

} else {

System.out.println("Cannot borrow book " + book.getBookId() + " by " + memberName);

}

}

public void returnBook(String bookId, Book[] books) {

for (int i = 0; i < bookCount; i++) {

if (booksIssued[i].equals(bookId)) {

for (Book b : books) {

if (b.getBookId().equals(bookId)) {

b.returnBook();

break;

}

}

// shift array left

for (int j = i; j < bookCount - 1; j++) {

booksIssued[j] = booksIssued[j + 1];

}

booksIssued[bookCount - 1] = null;

bookCount--;

System.out.println(memberName + " returned " + bookId);

return;

}

}

System.out.println(memberName + " does not have book " + bookId);

}

public void displayMemberInfo() {

System.out.println("Member ID : " + memberId);

System.out.println("Name : " + memberName);

System.out.print("Books Issued: ");

for (int i = 0; i < bookCount; i++) {

System.out.print(booksIssued[i] + " ");

}

if (bookCount == 0) System.out.print("None");

System.out.println("\n-----------------------------");

}

private static String generateMemberId() {

return String.format("M%03d", totalMembers + 1);

}

}

public static void main(String[] args) {

Book[] books = new Book[3];

books[0] = new Book("1984", "George Orwell");

books[1] = new Book("The Alchemist", "Paulo Coelho");

books[2] = new Book("Harry Potter", "J.K. Rowling");

Member[] members = new Member[2];

members[0] = new Member("Alice");

members[1] = new Member("Bob");

members[0].borrowBook(books[0]);

members[0].borrowBook(books[1]);

members[1].borrowBook(books[1]); // already issued

members[0].displayMemberInfo();

members[1].displayMemberInfo();

books[0].displayBookInfo();

books[1].displayBookInfo();

members[0].returnBook("B001", books);

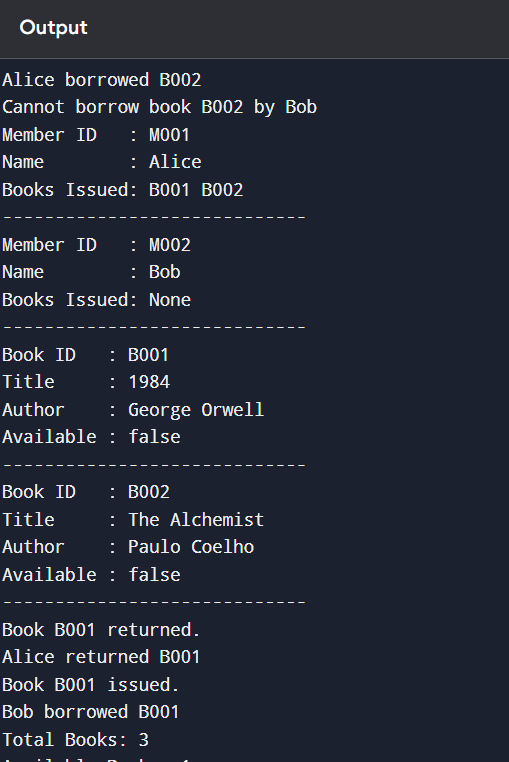
members[1].borrowBook(books[0]); // now available

System.out.println("Total Books: " + Book.getTotalBooks());

System.out.println("Available Books: " + Book.getAvailableBooks());

}

}



Q3. Write a program to create an Employee payroll system with different employee types using method overloading

Soln.

public class PayrollSystem {

static class Employee {

private String empId;

private String empName;

private String department;

private double baseSalary;

private String empType;

private static int totalEmployees = 0;

public Employee(String empName, String department, double baseSalary) {

this.empId = generateEmpId();

this.empName = empName;

this.department = department;

this.baseSalary = baseSalary;

this.empType = "Full-Time";

totalEmployees++;

}

public Employee(String empName, String department, double hourlyRate, int hoursWorked) {

this.empId = generateEmpId();

this.empName = empName;

this.department = department;

this.baseSalary = hourlyRate \* hoursWorked;

this.empType = "Part-Time";

totalEmployees++;

}

public Employee(String empName, String department, double contractAmount, boolean isContract) {

this.empId = generateEmpId();

this.empName = empName;

this.department = department;

this.baseSalary = contractAmount;

this.empType = "Contract";

totalEmployees++;

}

public double calculateSalary() {

return baseSalary;

}

public double calculateSalary(double bonus) {

if (empType.equals("Full-Time")) {

return baseSalary + bonus;

}

return baseSalary;

}

public double calculateSalary(int hours, double rate) {

if (empType.equals("Part-Time")) {

return hours \* rate;

}

return baseSalary;

}

public double calculateSalary(boolean isContract) {

if (empType.equals("Contract")) {

return baseSalary;

}

return baseSalary;

}

public double calculateTax() {

if (empType.equals("Full-Time")) return baseSalary \* 0.2;

if (empType.equals("Part-Time")) return baseSalary \* 0.1;

if (empType.equals("Contract")) return baseSalary \* 0.15;

return 0;

}

public double calculateTax(double customRate) {

return baseSalary \* customRate;

}

public void generatePaySlip() {

System.out.println("----- Pay Slip -----");

System.out.println("Employee ID : " + empId);

System.out.println("Name : " + empName);

System.out.println("Department : " + department);

System.out.println("Type : " + empType);

System.out.println("Base Salary : " + baseSalary);

System.out.println("Calculated Tax: " + calculateTax());

System.out.println("Net Salary : " + (calculateSalary() - calculateTax()));

System.out.println("---------------------\n");

}

public void displayEmployeeInfo() {

System.out.println("Employee ID: " + empId + " | Name: " + empName + " | Type: " + empType + " | Department: " + department);

}

private static String generateEmpId() {

return String.format("E%03d", totalEmployees + 1);

}

public static int getTotalEmployees() {

return totalEmployees;

}

public static void generateCompanyReport(Employee[] employees) {

System.out.println("=== Company Payroll Report ===");

double totalSalary = 0, totalTax = 0;

for (Employee e : employees) {

totalSalary += e.calculateSalary();

totalTax += e.calculateTax();

e.displayEmployeeInfo();

}

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

System.out.println("Total Salary Paid: " + totalSalary);

System.out.println("Total Tax Collected: " + totalTax);

System.out.println("==============================");

}

}

public static void main(String[] args) {

Employee e1 = new Employee("Alice", "HR", 50000); // Full-Time

Employee e2 = new Employee("Bob", "IT", 200, 80); // Part-Time

Employee e3 = new Employee("Charlie", "Finance", 60000, true); // Contract

e1.generatePaySlip();

e2.generatePaySlip();

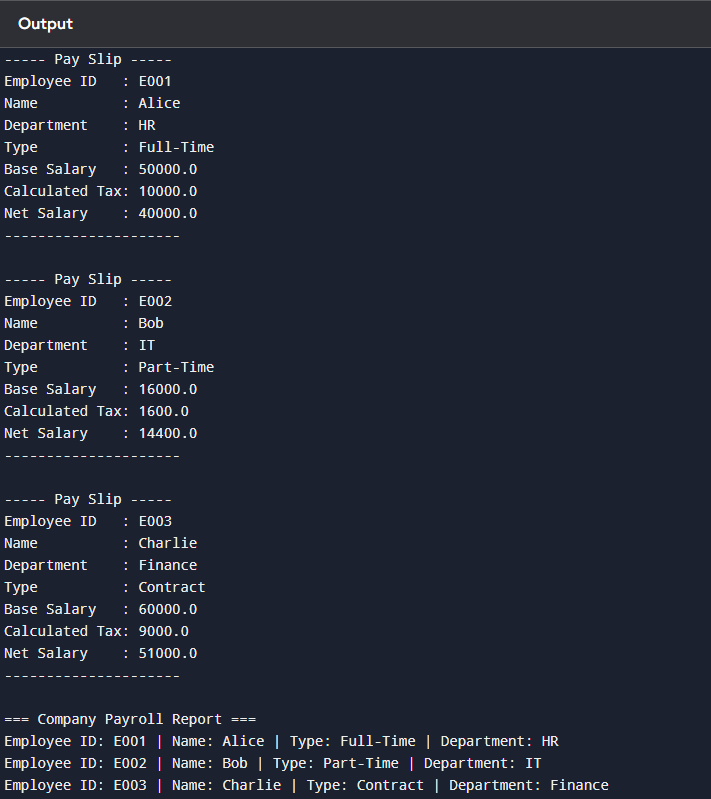
e3.generatePaySlip();

Employee[] employees = { e1, e2, e3 };

Employee.generateCompanyReport(employees);

}

}



Q4. Write a program to create a Vehicle rental system demonstrating static and instance members

Soln.

public class VehicleRentalSystem {

static class Vehicle {

private String vehicleId;

private String brand;

private String model;

private double rentPerDay;

private boolean isAvailable;

private int totalRentedDays;

private static int totalVehicles = 0;

private static double totalRevenue = 0;

private static int rentalDays = 0;

private static String companyName = "Default Rentals";

public Vehicle(String brand, String model, double rentPerDay) {

this.vehicleId = generateVehicleId();

this.brand = brand;

this.model = model;

this.rentPerDay = rentPerDay;

this.isAvailable = true;

this.totalRentedDays = 0;

totalVehicles++;

}

public double rentVehicle(int days) {

if (!isAvailable) {

System.out.println("Vehicle " + vehicleId + " is already rented.");

return 0;

}

double rent = calculateRent(days);

isAvailable = false;

totalRentedDays += days;

System.out.println("Vehicle " + vehicleId + " rented for " + days + " days. Rent = Rs." + rent);

return rent;

}

public void returnVehicle() {

if (isAvailable) {

System.out.println("Vehicle " + vehicleId + " is already available.");

} else {

isAvailable = true;

System.out.println("Vehicle " + vehicleId + " returned and available for rent.");

}

}

public double calculateRent(int days) {

double rent = rentPerDay \* days;

totalRevenue += rent;

rentalDays += days;

return rent;

}

public void displayVehicleInfo() {

System.out.println("Vehicle ID : " + vehicleId);

System.out.println("Brand : " + brand);

System.out.println("Model : " + model);

System.out.println("Rent/Day : Rs." + rentPerDay);

System.out.println("Available : " + isAvailable);

System.out.println("Rented Days: " + totalRentedDays);

System.out.println("-----------------------------");

}

public static void setCompanyName(String name) {

companyName = name;

}

public static double getTotalRevenue() {

return totalRevenue;

}

public static double getAverageRentPerDay() {

if (rentalDays == 0) return 0;

return totalRevenue / rentalDays;

}

public static void displayCompanyStats() {

System.out.println("===== " + companyName + " =====");

System.out.println("Total Vehicles : " + totalVehicles);

System.out.println("Total Revenue : Rs." + totalRevenue);

System.out.println("Total Rental Days: " + rentalDays);

System.out.println("Average Rent/Day : Rs." + getAverageRentPerDay());

System.out.println("===============================");

}

private static String generateVehicleId() {

return String.format("V%03d", totalVehicles + 1);

}

}

public static void main(String[] args) {

Vehicle.setCompanyName("Zoomy Rentals");

Vehicle v1 = new Vehicle("Toyota", "Corolla", 1200);

Vehicle v2 = new Vehicle("Honda", "Civic", 1500);

Vehicle v3 = new Vehicle("Tesla", "Model 3", 3000);

v1.rentVehicle(3);

v2.rentVehicle(2);

v1.returnVehicle();

v3.rentVehicle(5);

v1.displayVehicleInfo();

v2.displayVehicleInfo();

v3.displayVehicleInfo();

Vehicle.displayCompanyStats();

}

}

