```java
// Write a program to create a Bank Account management system
// without using built-in collection classes
// Hint =>
// a. Create a BankAccount class with private instance variables:
accountNumber
// (String), accountHolderName (String), balance (double), and a static
variable
// totalAccounts (int)
// b. Create a constructor that takes account holder name and initial
deposit,
// automatically generates account number using a static counter
// c. Create instance methods: deposit(double amount), withdraw(double
amount),
// checkBalance() with proper validation for negative amounts and
insufficient funds
// d. Create static methods: getTotalAccounts(), generateAccountNumber()
that
// returns a unique account number like "ACC001", "ACC002"
// e. Create a method displayAccountInfo() to show all account details in
a formatted
// manner
// f.
// In the main method, create an array of BankAccount objects, demonstrate
// creating multiple accounts, performing transactions, and showing the
difference
// between static and instance variables


public class BankAccountManager {

    static class BankAccount {
        private String accountNumber;
        private String accountHolderName;
        private double balance;
        private static int totalAccounts = 0;

        // Constructor
        public BankAccount(String accountHolderName, double
initialDeposit) {
            if (initialDeposit < 0) {
```

```
                throw new IllegalArgumentException("Initial deposit cannot
be negative.");
            }
            this.accountHolderName = accountHolderName;
            this.balance = initialDeposit;
            this.accountNumber = generateAccountNumber();
            totalAccounts++;
        }

        // Static method to generate unique account number
        public static String generateAccountNumber() {
            return String.format("ACC%03d", totalAccounts + 1);
        }

        // Static method to get total accounts
        public static int getTotalAccounts() {
            return totalAccounts;
        }

        // Deposit method
        public void deposit(double amount) {
            if (amount <= 0) {
                System.out.println("Deposit amount must be positive.");
                return;
            }
            balance += amount;
            System.out.println("Deposited ₹" + amount + " to " +
accountNumber);
        }

        // Withdraw method
        public void withdraw(double amount) {
            if (amount <= 0) {
                System.out.println("Withdrawal amount must be positive.");
                return;
            }
            if (amount > balance) {
                System.out.println("Insufficient funds in " +
accountNumber);
                return;
```

```
            }
            balance -= amount;
            System.out.println("Withdrew ₹" + amount + " from " +
accountNumber);
        }

        // Check balance
        public double checkBalance() {
            return balance;
        }

        // Display account info
        public void displayAccountInfo() {
            System.out.println("----- Account Info -----");
            System.out.println("Account Number : " + accountNumber);
            System.out.println("Holder Name    : " + accountHolderName);
            System.out.println("Balance        : ₹" +
String.format("%.2f", balance));
            System.out.println("-------------------------");
        }
    }

    public static void main(String[] args) {
        // Create array of BankAccount objects
        BankAccount[] accounts = new BankAccount[3];

        // Create accounts
        accounts[0] = new BankAccount("Alice", 5000);
        accounts[1] = new BankAccount("Bob", 3000);
        accounts[2] = new BankAccount("Charlie", 7000);

        // Perform transactions
        accounts[0].deposit(1500);
        accounts[1].withdraw(1000);
        accounts[2].withdraw(8000); // Should show insufficient funds

        // Display account info
        for (BankAccount acc : accounts) {
            acc.displayAccountInfo();
        }
```

```
        // Show static vs instance variable
        System.out.println("Total Bank Accounts Created: " +
BankAccount.getTotalAccounts());
    }
}
```

```
Compiling BankAccountManager.java...
Compilation successful. Running program...

Deposited ?1500.0 to ACC001
Withdrew ?1000.0 from ACC002
Insufficient funds in ACC003
----- Account Info -----
Account Number : ACC001
Holder Name    : Alice
Balance        : ?6500.00
------------------------
----- Account Info -----
Account Number : ACC002
Holder Name    : Bob
Balance        : ?2000.00
------------------------
----- Account Info -----
Account Number : ACC003
Holder Name    : Charlie
Balance        : ?7000.00
------------------------
Total Bank Accounts Created: 3

Program finished. Cleaning up...
BankAccountManager.class file deleted successfully.
Press any key to continue . . .
```

```java
// Write a program to create a Library Book management system,
// demonstrating object relationships
// Hint =>
// a. Create a Book class with private variables: bookId (String), title
(String), author
// (String),
// isAvailable (boolean), and static variables totalBooks (int),
// availableBooks (int)
// b. Create a constructor for Book class and methods: issueBook(),
returnBook(),
// displayBookInfo()
// c. Create a Member class with private variables: memberId (String),
memberName
// (String), booksIssued (String array to store book IDs), bookCount (int
to track number
// of books issued)
// d. Create methods in Member class: borrowBook(Book book) which checks
if book
// is available and updates both book and member status, returnBook(String
// bookId, Book[] books) to return a specific book
// 3
// e. Create static methods in both classes to generate unique IDs and
track statistics
// f. In main, create arrays of Book and Member objects, demonstrate
borrowing and
// returning books, showing how objects interact with each other


public class LibraryManagementSystem {

    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;

        // Constructor
        public Book(String title, String author) {
```

```
            this.title = title;
            this.author = author;
            this.bookId = generateBookId();
            this.isAvailable = true;
            totalBooks++;
            availableBooks++;
        }

        // Static method to generate unique book ID
        public static String generateBookId() {
            return String.format("B%03d", totalBooks + 1);
        }

        // Issue book
        public void issueBook() {
            if (isAvailable) {
                isAvailable = false;
                availableBooks--;
            } else {
                System.out.println("Book " + bookId + " is already
issued.");
            }
        }

        // Return book
        public void returnBook() {
            if (!isAvailable) {
                isAvailable = true;
                availableBooks++;
            } else {
                System.out.println("Book " + bookId + " was not issued.");
            }
        }

        // Display book info
        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("----------------------------");
        }

        // Static methods for statistics
        public static int getTotalBooks() {
            return totalBooks;
        }

        public static int getAvailableBooks() {
            return availableBooks;
        }

        public String getBookId() {
            return bookId;
        }

        public boolean isAvailable() {
            return isAvailable;
        }
    }

    static class Member {
        private String memberId;
        private String memberName;
        private String[] booksIssued;
        private int bookCount;
        private static int totalMembers = 0;

        // Constructor
        public Member(String memberName) {
            this.memberName = memberName;
            this.memberId = generateMemberId();
            this.booksIssued = new String[5]; // Max 5 books
            this.bookCount = 0;
            totalMembers++;
        }

        // Static method to generate unique member ID
        public static String generateMemberId() {
            return String.format("M%03d", totalMembers + 1);
```

```
        }

        // Borrow book
        public void borrowBook(Book book) {
            if (bookCount >= booksIssued.length) {
                System.out.println("Member " + memberId + " has reached
book limit.");
                return;
            }
            if (book.isAvailable()) {
                book.issueBook();
                booksIssued[bookCount++] = book.getBookId();
                System.out.println(memberName + " borrowed book " +
book.getBookId());
            } else {
                System.out.println("Book " + book.getBookId() + " is not
available.");
            }
        }

        // Return book
        public void returnBook(String bookId, Book[] books) {
            boolean found = false;
            for (int i = 0; i < bookCount; i++) {
                if (booksIssued[i].equals(bookId)) {
                    booksIssued[i] = null;
                    bookCount--;
                    found = true;
                    break;
                }
            }
            if (!found) {
                System.out.println("Book " + bookId + " not found in
member's issued list.");
                return;
            }

            for (Book book : books) {
                if (book.getBookId().equals(bookId)) {
                    book.returnBook();
```

```
                    System.out.println(memberName + " returned book " +
bookId);
                    return;
                }
            }
            System.out.println("Book " + bookId + " not found in
library.");
        }

        // Display member info
        public void displayMemberInfo() {
            System.out.println("Member ID   : " + memberId);
            System.out.println("Name        : " + memberName);
            System.out.print("Books Issued: ");
            for (String id : booksIssued) {
                if (id != null) System.out.print(id + " ");
            }
            System.out.println("\n----------------------------");
        }

        public static int getTotalMembers() {
            return totalMembers;
        }
    }

    public static void main(String[] args) {
        // Create books
        Book[] books = new Book[3];
        books[0] = new Book("The Alchemist", "Paulo Coelho");
        books[1] = new Book("Clean Code", "Robert C. Martin");
        books[2] = new Book("Java: The Complete Reference", "Herbert
Schildt");

        // Create members
        Member[] members = new Member[2];
        members[0] = new Member("Alice");
        members[1] = new Member("Bob");

        // Borrowing books
        members[0].borrowBook(books[0]);
```

```
        members[1].borrowBook(books[1]);
        members[0].borrowBook(books[1]); // Should show unavailable

        // Returning books
        members[1].returnBook("B002", books);
        members[0].borrowBook(books[1]); // Now available

        // Display info
        for (Book book : books) {
            book.displayBookInfo();
        }

        for (Member member : members) {
            member.displayMemberInfo();
        }

        // Show statistics
        System.out.println("Total Books     : " + Book.getTotalBooks());
        System.out.println("Available Books : " +
Book.getAvailableBooks());
        System.out.println("Total Members   : " +
Member.getTotalMembers());
    }
}
```

```
Compiling LibraryManagementSystem.java...
Compilation successful. Running program...

Alice borrowed book B001
Bob borrowed book B002
Book B002 is not available.
Bob returned book B002
Alice borrowed book B002
Book ID     : B001
Title       : The Alchemist
Author      : Paulo Coelho
Available   : false
---------------------------
Book ID     : B002
Title       : Clean Code
Author      : Robert C. Martin
Available   : false
---------------------------
Book ID     : B003
Title       : Java: The Complete Reference
Author      : Herbert Schildt
Available   : true
---------------------------
Member ID   : M001
Name        : Alice
Books Issued: B001 B002
---------------------------
Member ID   : M002
Name        : Bob
Books Issued:
---------------------------
Total Books     : 3
Available Books : 1
Total Members   : 2
```

```
// Write a program to create an Employee payroll system with different
```

```java
// employee types using method overloading
// Hint =>
// a. Create an Employee class with private variables: empId (String),
empName (String),
// department (String), baseSalary (double), empType (String), and static
variable
// totalEmployees (int)
// b. Create multiple constructors for different employee types:
constructor for full-time
// employees, constructor for part-time employees, constructor for
contract employees
// c. Create overloaded methods calculateSalary(): one for full-time (base
salary +
// bonus), one for part-time (hourly rate × hours), one for contract
(fixed amount)
// d. Create overloaded methods calculateTax() with different tax rates
for different
// employee types
// e. Create methods generatePaySlip() to display employee details with
calculated
// salary and tax, displayEmployeeInfo() for formatted output
// f. Create static methods to track total employees and generate
company-wide payroll
// reports
// g. In main, create different types of employee objects, demonstrate
method overloading
// by calling the same method names with different parameters

public class EmployeePayrollSystem {

    static class Employee {
        private String empId;
        private String empName;
        private String department;
        private double baseSalary;
        private String empType;
        private static int totalEmployees = 0;

        // Constructor for full-time employee
```

```
        public Employee(String empName, String department, double
baseSalary) {
            this.empName = empName;
            this.department = department;
            this.baseSalary = baseSalary;
            this.empType = "Full-Time";
            this.empId = generateEmpId();
            totalEmployees++;
        }

        // Constructor for part-time employee
        public Employee(String empName, String department, double
hourlyRate, int hoursWorked) {
            this.empName = empName;
            this.department = department;
            this.baseSalary = hourlyRate * hoursWorked;
            this.empType = "Part-Time";
            this.empId = generateEmpId();
            totalEmployees++;
        }

        // Constructor for contract employee
        public Employee(String empName, String department, double
fixedAmount, boolean isContract) {
            this.empName = empName;
            this.department = department;
            this.baseSalary = fixedAmount;
            this.empType = "Contract";
            this.empId = generateEmpId();
            totalEmployees++;
        }

        // Static method to generate unique employee ID
        public static String generateEmpId() {
            return String.format("EMP%03d", totalEmployees + 1);
        }

        // Overloaded calculateSalary methods
        public double calculateSalary() {
            if (empType.equals("Full-Time")) {
```

```
                double bonus = 0.10 * baseSalary;
                return baseSalary + bonus;
            } else {
                return baseSalary; // For Part-Time and Contract
            }
        }

        // Overloaded calculateTax methods
        public double calculateTax() {
            switch (empType) {
                case "Full-Time":
                    return 0.20 * calculateSalary();
                case "Part-Time":
                    return 0.10 * calculateSalary();
                case "Contract":
                    return 0.05 * calculateSalary();
                default:
                    return 0.0;
            }
        }

        // Generate payslip
        public void generatePaySlip() {
            double salary = calculateSalary();
            double tax = calculateTax();
            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("Gross Salary  : ₹" + String.format("%.2f",
salary));
            System.out.println("Tax Deducted  : ₹" + String.format("%.2f",
tax));
            System.out.println("Net Salary    : ₹" + String.format("%.2f",
salary - tax));
            System.out.println("--------------------\n");
        }

        // Display employee info
```

```
        public void displayEmployeeInfo() {
            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   : ₹" + String.format("%.2f",
baseSalary));
            System.out.println("--------------------");
        }

        // Static method to get total employees
        public static int getTotalEmployees() {
            return totalEmployees;
        }

        // Static method to generate payroll report
        public static void generatePayrollReport(Employee[] employees) {
            System.out.println("=== Company Payroll Report ===");
            double totalPayroll = 0;
            for (Employee emp : employees) {
                totalPayroll += emp.calculateSalary();
            }
            System.out.println("Total Employees : " +
getTotalEmployees());
            System.out.println("Total Payroll   : ₹" +
String.format("%.2f", totalPayroll));
            System.out.println("==============================\n");
        }
    }

    public static void main(String[] args) {
        // Create employee objects
        Employee[] employees = new Employee[3];
        employees[0] = new Employee("Alice", "Engineering", 50000); //
Full-Time
        employees[1] = new Employee("Bob", "Support", 300, 40);     //
Part-Time
        employees[2] = new Employee("Charlie", "Consulting", 45000, true);
// Contract
```

```
        // Display info and generate payslips
        for (Employee emp : employees) {
            emp.displayEmployeeInfo();
            emp.generatePaySlip();
        }

        // Generate company-wide payroll report
        Employee.generatePayrollReport(employees);
    }
}
```

```
Employee ID   : EMP002
Name          : Bob
Department    : Support
Type          : Part-Time
Base Salary   : ?12000.00
--------------------
----- Pay Slip -----
Employee ID   : EMP002
Name          : Bob
Department    : Support
Type          : Part-Time
Gross Salary  : ?12000.00
Tax Deducted  : ?1200.00
Net Salary    : ?10800.00
--------------------

Employee ID   : EMP003
Name          : Charlie
Department    : Consulting
Type          : Contract
Base Salary   : ?45000.00
--------------------
----- Pay Slip -----
Employee ID   : EMP003
Name          : Charlie
Department    : Consulting
Type          : Contract
Gross Salary  : ?45000.00
Tax Deducted  : ?2250.00
Net Salary    : ?42750.00
--------------------

=== Company Payroll Report ===
Total Employees : 3
Total Payroll   : ?112000.00
==============================
```

```
// Write a program to create a Vehicle rental system demonstrating
// static and instance members
// Hint =>
```

```java
// a. Create a Vehicle class with private instance variables: vehicleId
(String), brand
// (String), model (String), rentPerDay (double), isAvailable (boolean)
// 4
// b. Create static variables: totalVehicles (int), totalRevenue (double),
// companyName (String), rentalDays (int to track total rental days)
// c. Create a constructor and instance methods: rentVehicle(int days)
which
// calculates rent and updates availability, returnVehicle() to make
vehicle available
// again
// d. Create static methods: setCompanyName(String name),
getTotalRevenue(),
// getAverageRentPerDay(), displayCompanyStats()
// e. Create a method calculateRent(int days) that returns rental amount
and
// updates static revenue counter
// f. Create a displayVehicleInfo() method showing all vehicle details
including
// rental history
// g. In main, demonstrate the difference between static and instance
members by creating
// multiple vehicle objects, show how static variables are shared across
all objects while
// instance variables are unique to each object


public class VehicleRentalSystem {

    static class Vehicle {
        private String vehicleId;
        private String brand;
        private String model;
        private double rentPerDay;
        private boolean isAvailable;
        private int totalRentalDays;

        private static int totalVehicles = 0;
        private static double totalRevenue = 0.0;
        private static int rentalDays = 0;
```

```
        private static String companyName = "Default Rentals";

        // Constructor
        public Vehicle(String brand, String model, double rentPerDay) {
            this.brand = brand;
            this.model = model;
            this.rentPerDay = rentPerDay;
            this.isAvailable = true;
            this.vehicleId = generateVehicleId();
            totalVehicles++;
        }

        // Static method to generate vehicle ID
        public static String generateVehicleId() {
            return String.format("V%03d", totalVehicles + 1);
        }

        // Instance method to rent vehicle
        public void rentVehicle(int days) {
            if (!isAvailable) {
                System.out.println("Vehicle " + vehicleId + " is not
available.");
                return;
            }
            double rent = calculateRent(days);
            isAvailable = false;
            totalRentalDays += days;
            System.out.println("Vehicle " + vehicleId + " rented for " +
days + " days. Rent: ₹" + rent);
        }

        // Instance method to return vehicle
        public void returnVehicle() {
            if (isAvailable) {
                System.out.println("Vehicle " + vehicleId + " is already
available.");
                return;
            }
            isAvailable = true;
```

```
            System.out.println("Vehicle " + vehicleId + " has been
returned.");
        }

        // Instance method to calculate rent
        public double calculateRent(int days) {
            double rent = rentPerDay * days;
            totalRevenue += rent;
            rentalDays += days;
            return rent;
        }

        // Display vehicle info
        public void displayVehicleInfo() {
            System.out.println("Vehicle ID     : " + vehicleId);
            System.out.println("Brand          : " + brand);
            System.out.println("Model          : " + model);
            System.out.println("Rent/Day       : ₹" + rentPerDay);
            System.out.println("Available      : " + isAvailable);
            System.out.println("Total Rented   : " + totalRentalDays + "
days");
            System.out.println("-----------------------------");
        }

        // Static methods
        public static void setCompanyName(String name) {
            companyName = name;
        }

        public static double getTotalRevenue() {
            return totalRevenue;
        }

        public static double getAverageRentPerDay() {
            return rentalDays == 0 ? 0 : totalRevenue / rentalDays;
        }

        public static void displayCompanyStats() {
            System.out.println("=== " + companyName + " Stats ===");
            System.out.println("Total Vehicles     : " + totalVehicles);
```

```
            System.out.println("Total Revenue      : ₹" +
String.format("%.2f", totalRevenue));
            System.out.println("Total Rental Days  : " + rentalDays);
            System.out.println("Average Rent/Day   : ₹" +
String.format("%.2f", getAverageRentPerDay()));
            System.out.println("================================");
        }
    }

    public static void main(String[] args) {
        // Set company name
        Vehicle.setCompanyName("Speedy Wheels Rentals");

        // Create vehicle objects
        Vehicle v1 = new Vehicle("Toyota", "Innova", 1500);
        Vehicle v2 = new Vehicle("Honda", "City", 1200);
        Vehicle v3 = new Vehicle("Suzuki", "Swift", 1000);

        // Rent and return operations
        v1.rentVehicle(3);
        v2.rentVehicle(2);
        v1.returnVehicle();
        v3.rentVehicle(5);

        // Display vehicle info
        v1.displayVehicleInfo();
        v2.displayVehicleInfo();
        v3.displayVehicleInfo();

        // Display company stats
        Vehicle.displayCompanyStats();
    }
}
```

```
Compiling VehicleRentalSystem.java...
Compilation successful. Running program...

Vehicle V001 rented for 3 days. Rent: ?4500.0
Vehicle V002 rented for 2 days. Rent: ?2400.0
Vehicle V001 has been returned.
Vehicle V003 rented for 5 days. Rent: ?5000.0
Vehicle ID     : V001
Brand          : Toyota
Model          : Innova
Rent/Day       : ?1500.0
Available      : true
Total Rented   : 3 days
----------------------------
Vehicle ID     : V002
Brand          : Honda
Model          : City
Rent/Day       : ?1200.0
Available      : false
Total Rented   : 2 days
----------------------------
Vehicle ID     : V003
Brand          : Suzuki
Model          : Swift
Rent/Day       : ?1000.0
Available      : false
Total Rented   : 5 days
----------------------------
=== Speedy Wheels Rentals Stats ===
Total Vehicles     : 3
Total Revenue      : ?11900.00
Total Rental Days  : 10
Average Rent/Day   : ?1190.00
============================

Program finished. Cleaning up...
VehicleRentalSystem.class file deleted successfully.
Press any key to continue . . .
```