Unit 1

Classes & Objects: Classes, objects and methods: defining a class, Access Control, Method overloading, constructors, constructor overloading, use of this and static.

Question 1: Implement a Banking System

Problem Statement

Design a BankAccount class that supports:

1. **Creating a bank account** with an account number and initial balance.
2. **Depositing money** into the account.
3. **Withdrawing money** (but ensure sufficient balance).
4. **Checking the account balance**.

Expected Features

- **Constructor** to initialize account number and balance.
- **Methods**: deposit(), withdraw(), getBalance().
- **Balance validation**: Withdrawal should not exceed available funds.

```
class BankAccount {
  private String accountNumber;
  private double balance;

  // Constructor
  public BankAccount(String accountNumber, double initialBalance) {
    this.accountNumber = accountNumber;
    this.balance = initialBalance;
  }

  // Deposit method
  public void deposit(double amount) {
    if (amount > 0) {
      balance += amount;
      System.out.println("Deposited: $" + amount);
    } else {
      System.out.println("Deposit amount must be positive.");
    }
  }

  // Withdraw method
  public void withdraw(double amount) {
    if (amount > 0 && amount <= balance) {
      balance -= amount;
      System.out.println("Withdrawn: $" + amount);
    } else {
      System.out.println("Insufficient balance or invalid amount.");
    }
  }

  // Get balance method
  public void getBalance() {
        System.out.println("Current Balance: $" + balance);
    }
}

public class Main {
    public static void main(String[] args) {
        BankAccount myAccount = new BankAccount("123456", 500);
        myAccount.getBalance();
        myAccount.deposit(200);
        myAccount.withdraw(100);
        myAccount.getBalance();
    }
}
```

Question 2: Implement a Library Management System
Problem Statement
Create a Book class that contains:

1. **Book title, author, and availability status**.
2. **Methods**:
    - borrowBook() → Mark as borrowed (if available).
    - returnBook() → Mark as available.
    - displayDetails() → Show book details.

Expected Features

- **Constructor** to initialize title, author, and set availability to true.
- **Methods**: borrowBook(), returnBook(), displayDetails().
- **Check Availability** before borrowing.

```
class Book {
    private String title;
    private String author;
    private boolean isAvailable;

    // Constructor
    public Book(String title, String author) {
        this.title = title;
        this.author = author;
        this.isAvailable = true;
    }

    // Borrow book method
    public void borrowBook() {
        if (isAvailable) {
            isAvailable = false;
            System.out.println(title + " has been borrowed.");
        } else {
            System.out.println(title + " is already borrowed.");
        }
    }


    // Return book method
    public void returnBook() {
        if (!isAvailable) {
            isAvailable = true;
            System.out.println(title + " has been returned.");
        } else {
            System.out.println(title + " is already in the library.");
        }
    }

    // Display book details
    public void displayDetails() {
        System.out.println("Title: " + title + ", Author: " + author + ", Available: " + isAvailable);
    }
}

public class Main {
    public static void main(String[] args) {
        Book book1 = new Book("1984", "George Orwell");
        book1.displayDetails();
        book1.borrowBook();
        book1.displayDetails();
        book1.returnBook();
        book1.displayDetails();
    }
}
```

Question 3
Student Management System Implementation
Features Covered
Constructor initializes student details
Methods: calculateTotalMarks(), calculateAverageMarks(), displayStudentDetails()
Marks validation (between 0-100)

```
class Student {
    private String name;
    private int rollNumber;
    private int marks1, marks2, marks3;

    // Constructor
    public Student(String name, int rollNumber, int marks1, int marks2, int marks3) {
        this.name = name;
        this.rollNumber = rollNumber;
        if (isValidMarks(marks1) && isValidMarks(marks2) && isValidMarks(marks3)) {
            this.marks1 = marks1;
            this.marks2 = marks2;
            this.marks3 = marks3;
        } else {
            System.out.println("Invalid marks! Marks should be between 0 and 100.");
        }
    }

    // Method to validate marks
    private boolean isValidMarks(int marks) {
        return marks >= 0 && marks <= 100;
    }

    // Calculate total marks
    public int calculateTotalMarks() {
        return marks1 + marks2 + marks3;
    }

    // Calculate average marks
    public double calculateAverageMarks() {
        return calculateTotalMarks() / 3.0;
    }

    // Display student details
    public void displayStudentDetails() {
        System.out.println("Name: " + name);
        System.out.println("Roll Number: " + rollNumber);
        System.out.println("Marks: " + marks1 + ", " + marks2 + ", " + marks3);
        System.out.println("Total Marks: " + calculateTotalMarks());
        System.out.println("Average Marks: " + calculateAverageMarks());
    }
}

public class Main {
    public static void main(String[] args) {
        Student student1 = new Student("Alice", 101, 85, 90, 78);
        student1.displayStudentDetails();
    }
}
```

Mcq

Question 1

Static block instance block and constructor

```
class Test {
    static {
        System.out.println("Static Block");
    }

    {
        System.out.println("Instance Block");
    }

    Test() {
        System.out.println("Constructor");
    }

    public static void main(String[] args) {
        Test t1 = new Test();
        Test t2 = new Test();
    }
}
```

Question 2

```
class Test {
    {
        System.out.println("Instance Block");
    }

    Test() {
        System.out.println("Constructor");
    }

    public static void main(String[] args) {
        new Test();
    }
}
```

Question 3 Example of this

```
class Test {
    Test() {
        this(10);
        System.out.println("Default Constructor");
    }

    Test(int x) {
        System.out.println("Parameterized Constructor: " + x);
    }
    public static void main(String[] args) {
        Test obj = new Test();
    }
}
```

Example 4

```
class Test {
    static {
        System.out.println("Static Block");
    }

    public static void main(String[] args) {
        System.out.println("Main Method");
    }
}
```

Example 5

```
class Test {
    static {
        System.out.println("Static Block 1");
    }

    {
        System.out.println("Instance Block 1");
    }

    Test() {
        System.out.println("Constructor");
    }

    static {
        System.out.println("Static Block 2");
    }

    {
        System.out.println("Instance Block 2");
    }

    public static void main(String[] args) {
        Test obj1 = new Test();
        Test obj2 = new Test();
    }
}
```

Example 7
```
class Test {
    int x = 10;

    static {
        System.out.println(x);
        System.out.println("Static Block");
    }

    public static void main(String[] args) {
        Test obj = new Test();
    }
}
```

Example 8
```
class Test {
    int x = 10;

    static {
        //System.out.println(x);
        System.out.println("Static Block");
    }
    {

     System.out.println(x);
    }

    public static void main(String[] args) {
        Test obj1 = new Test();
        Test obj2 = new Test();
    }
}
```

Example 9
```
class Test {
    private int num = 10;
}

public class Main {
    public static void main(String[] args) {
        Test obj = new Test();
        System.out.println(obj.num);
    }
}
```
Ans: compilation error