Experiment-1.2

Student Name: Vaibhav Kumar
UID: 23BCS13386
Branch: B.E-C.S.E
Section/Group: 23KRG-2B
Semester: 5th
Date of Performance: 18/08/2025
Subject Name: PBLJ
Subject Code: 23CSH-304

Easy Level

1. **Aim:** Write a Java program to create a Product class with attributes id, name, and price. The program should: Demonstrate the use of constructors and methods to display product details

2. **Objective:** Understand the use of classes, constructors, and methods in Java.

3. **Input/Apparatus Used:** Java class definition, constructor, and method usage.

4. **Procedure:**
   Step1: Define a class named `Product` with attributes `id`, `name`, and `price`.
   Step2: Use a parameterized constructor to initialize these attributes.
   Step3: Define a method `displayDetails()` to print product information.
   Step4: In the main method, create an object and display its details.

   Sample Input:
   Product ID: 101
   Name: Laptop
   Price: 75000

   Sample Output:
   Product Details:
   ID: 101
   Name: Laptop
   Price: 75000

5. Code:

```java
class Product {
    int id;
    String name;
    double price;

    public Product(int id, String name, double price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    public void displayDetails() {
        System.out.println(x:"=== Product Details ===");
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("Price: " + price);
    }
}

class ProductDemo {
    Run | Debug
    public static void main(String[] args) {

        Product p1 = new Product(id:101, name:"Laptop", price:75000);

        p1.displayDetails();
    }
}
```

6. Output:

```
=== Product Details ===
ID: 101
Name: Laptop
Price: 75000.0
PS C:\Users\vaibh\OneDrive\Desktop\Java 1>
```

Medium Level

1. **Aim:** Write a Java program to implement a library management system. The program should: Use a base class Book and derived classes Fiction and NonFiction

2. **Objective:** Understand inheritance and dynamic method invocation in Java.

3. **Input/Apparatus Used:** Java inheritance using base and derived classes.

4. **Procedure:**
   Step1: Define a base class `Book` with common attributes like title, author, and price.
   Step2: Create two derived classes: `Fiction` and `NonFiction` extending the `Book` class.
   Step3: Override method in each subclass to display respective book details.
   Step4: Instantiate objects of each subclass and invoke their display methods.

   Sample Input:
   Book 1:
   Type: Fiction
   Title: Harry Potter
   Author: J.K. Rowling
   Price: 500

   Book 2:
   Type: Non-Fiction
   Title: Sapiens
   Author: Yuval Noah Harari
   Price: 700

   Sample Output:
   Fiction Book Details:
   Title: Harry Potter
   Author: J.K. Rowling
   Price: 500

   Non-Fiction Book Details:
   Title: Sapiens
   Author: Yuval Noah Harari
   Price: 700

5. Code:

```java
class Book {
    String title;
    String author;
    double price;

    public Book(String title, String author, double price) {
        this.title = title;
        this.author = author;
        this.price = price;
    }

    public void displayDetails() {
        System.out.println(x:"=== Book Details ===");
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Price: " + price);
    }
}

class Fiction extends Book {
    public Fiction(String title, String author, double price) {
        super(title, author, price);
    }

    @Override
    public void displayDetails() {
        System.out.println(x:"=== Fiction Book ===");
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Price: " + price);
    }
}
```

```java
class NonFiction extends Book {
    public NonFiction(String title, String author, double price) {
        super(title, author, price);
    }

    @Override
    public void displayDetails() {
        System.out.println(x:"=== Non-Fiction Book ===");
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Price: " + price);
    }
}

class LibraryManagement {
    Run | Debug
    public static void main(String[] args) {

        Book b1 = new Fiction(title:"Harry Potter", author:"J.K. Rowling", price:500);
        Book b2 = new NonFiction(title:"Sapiens", author:"Yuval Noah Harari", price:700);

        b1.displayDetails();
        System.out.println();
        b2.displayDetails();
    }
}
```

6. Output:

```
=== Fiction Book ===
Title: Harry Potter
Author: J.K. Rowling
Price: 500.0

=== Non-Fiction Book ===
Title: Sapiens
Author: Yuval Noah Harari
Price: 700.0
PS C:\Users\vaibh\OneDrive\Desktop\Java 1>
```

Hard Level

1. **Aim:** Design a student information system using Java with the following features: Use an abstract class Person with attributes name, age, and methods like displayDetails(). Create derived classes Student and Teacher to override displayDetails() and add unique attributes like rollNumber for students and subject for teachers.

2. **Objective:** Demonstrate abstraction and polymorphism using abstract classes and derived classes.
3. **Input/Apparatus Used:** Abstract classes, inheritance, and overriding in Java.

4. **Procedure:**
   Step1: Define an abstract class `Person` with attributes `name` and `age`, and an abstract method `displayDetails()`.
   Step2: Create a `Student` class extending `Person`, with an additional attribute `rollNumber`, and implement `displayDetails()`.
   Step3: Create a `Teacher` class extending `Person`, with an additional attribute `subject`, and implement `displayDetails()`.
   Step4: In the main method, create objects of `Student` and `Teacher`, and invoke `displayDetails()` on each.

   Sample Input:
   Add Student:
   Name: Alice
   Age: 20
   Roll Number: 101

   Add Teacher:
   Name: Mr. Smith

Age: 40
Subject: Mathematics

Sample Output:
Student Details:
Name: Alice
Age: 20
Roll Number: 101

Teacher Details:
Name: Mr. Smith
Age: 40
Subject: Mathematics

5. Code:

```
abstract class Person {
    String name;
    int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public abstract void displayDetails();
}

class Student extends Person {
    int rollNumber;

    public Student(String name, int age, int rollNumber) {
        super(name, age);
        this.rollNumber = rollNumber;
    }

    @Override
    public void displayDetails() {
        System.out.println(x:"=== Student Details ===");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Roll Number: " + rollNumber);
    }
}
```

```java
class Teacher extends Person {
    String subject;

    public Teacher(String name, int age, String subject) {
        super(name, age);
        this.subject = subject;
    }

    @Override
    public void displayDetails() {
        System.out.println(x:"=== Teacher Details ===");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Subject: " + subject);
    }
}

class StudentInformationSystem {
    Run | Debug
    public static void main(String[] args) {

        Student s1 = new Student(name:"Alice", age:20, rollNumber:101);
        Teacher t1 = new Teacher(name:"Mr. Smith", age:40, subject:"Mathematics");

        Person p1 = s1;
        Person p2 = t1;

        p1.displayDetails();
        System.out.println();
        p2.displayDetails();
    }
}
```

6. Output:

```
=== Student Details ===
Name: Alice
Age: 20
Roll Number: 101

=== Teacher Details ===
Name: Mr. Smith
Age: 40
Subject: Mathematics
PS C:\Users\vaibh\OneDrive\Desktop\Java 1>
```

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.