▨ 1. Java Overview & Environment Setup

1. What are the steps to install and set up Java on your system?
2. Write down the difference between JDK, JRE, and JVM.
3. Create a simple "Hello, Java!" program and run it using the terminal.

▨ 2. Your First Java Program

1. Write a Java program that prints your name, age, and favorite programming language.
2. Write a program that adds two integers and prints the result.
3. Write a program to print a multi-line quote using System.out.println.

▨ 3. Java Basics: Syntax, Data Types, Variables, Operators

1. Create a program to demonstrate all Java primitive data types.
2. Write a program that takes two numbers and prints the result of all arithmetic operations (+, -, *, /, %).
3. Declare constants using final and try modifying them—observe what happens.

▨ 4. Object-Oriented Programming (OOP)

a. Classes and Objects

1. Create a class Car with properties: brand, model, year. Create 2 objects.
2. Write a class Student with a method to display student details.
3. Create a class with a constructor that initializes values and displays them.

b. Inheritance

1. Create a base class Animal and a derived class Dog that inherits from Animal.
2. Add a method in the Animal class and override it in the Dog class.
3. Use super to access the parent class method.

c. Polymorphism

1. Demonstrate method overloading with multiple add() methods.
2. Demonstrate method overriding using inheritance.
3. Create an interface and implement it in a class.

d. Encapsulation

1. Create a class with private variables and use getter/setter methods.

2. Try accessing a private field directly—observe what happens.
3. Create a BankAccount class using encapsulation.

e. Abstraction

1. Create an abstract class with one abstract method.
2. Implement the abstract class in a subclass and override the method.
3. Explain the difference between abstract classes and interfaces in code.

---

5. Exception Handling

a. Try-Catch Blocks

1. Write a program that causes a divide-by-zero exception and handles it.
2. Use try-catch-finally blocks to demonstrate final usage.
3. Handle ArrayIndexOutOfBoundsException.

b. Custom Exceptions

1. Create a custom exception InvalidAgeException and throw it if age < 18.
2. Create a program to validate password length; throw exception if invalid.
3. Handle both built-in and custom exceptions together.

---

6. Multithreading

a. Thread Basics

1. Create a class that extends Thread and prints numbers from 1–10.
2. Create a class that implements Runnable and prints a message.
3. Start two threads and run them simultaneously.

b. Synchronization

1. Write a program that prints a table using synchronized method.
2. Use synchronized blocks in a multithreaded class.
3. Create a race condition and fix it using synchronization.

---

7. Collections Framework

a. Lists

1. Create a List of 5 names and display them using a loop.
2. Add, remove, and update elements in an ArrayList.

3. Sort a list of numbers in ascending order.

b. Sets

1. Create a HashSet and demonstrate uniqueness.
2. Try adding duplicate elements and observe the output.
3. Convert a List to Set.

c. Maps

1. Create a HashMap to store student roll numbers and names.
2. Iterate through the map using entrySet().
3. Remove a key-value pair from the map.

▨ 8. JDBC & Advanced Topics

a. JDBC Basics

1. Write JDBC code to connect to a local MySQL database.
2. Write a query to insert a row into a database.
3. Write a query to retrieve and display all rows.

▨ 9. Streams & Lambda Expressions

1. Use a stream to filter even numbers from a list.
2. Use forEach() and lambda to print square of numbers in a list.
3. Convert a list of strings to uppercase using streams.

Solutions

☑ 1. Java Overview & Setup

(No code here, setup instructions)

- Download and install JDK (https://www.oracle.com/java/technologies/javase-downloads.html)
- Set JAVA_HOME and add bin to the PATH variable
- Use IDE: IntelliJ IDEA / VS Code with Java Extension Pack

☑ 2. Your First Java Program

java

CopyEdit

```
// Example 1: Hello World
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Welcome to Java Programming!");
    }
}

// Example 2: Print Your Name
public class MyName {
    public static void main(String[] args) {
        System.out.println("Mahendra Bella");
    }
}

// Example 3: Simple Sum
public class AddTwoNumbers {
    public static void main(String[] args) {
        int a = 10, b = 20;
        int sum = a + b;
        System.out.println("Sum: " + sum);
    }
}
```

☑ **3. Java Basics - Syntax, Data Types, Variables, Operators**

java

CopyEdit

```
// Example 1: Data Types and Operators
public class Basics {
    public static void main(String[] args) {
        int a = 10;
```

```
        float b = 5.5f;
        boolean flag = true;
        char c = 'M';
        System.out.println(a + b);
        System.out.println(a > 5 && b < 6);
    }
}

// Example 2: Type Conversion
public class TypeCasting {
    public static void main(String[] args) {
        int x = 100;
        double y = x; // Implicit casting
        System.out.println("Double: " + y);
    }
}

// Example 3: Arithmetic & Relational Operators
public class OperatorsDemo {
    public static void main(String[] args) {
        int x = 15, y = 10;
        System.out.println("Sum: " + (x + y));
        System.out.println("Is x > y? " + (x > y));
    }
}
```

☑ 4. OOP – Classes, Inheritance, Polymorphism, Encapsulation, Abstraction

```
java
CopyEdit
// Example 1: Inheritance
class Person {
```

```
    private String name;
    public Person(String name) { this.name = name; }
    public void display() { System.out.println("Name: " + name); }
}
class Student extends Person {
    private int id;
    public Student(String name, int id) {
        super(name);
        this.id = id;
    }
    @Override
    public void display() {
        super.display();
        System.out.println("ID: " + id);
    }
}

// Example 2: Encapsulation
class Employee {
    private String empName;
    public void setName(String name) { this.empName = name; }
    public String getName() { return empName; }
}

// Example 3: Abstraction with Abstract Class
abstract class Shape {
    abstract void draw();
}
class Circle extends Shape {
    void draw() {
        System.out.println("Drawing Circle");
```

```
    }
}
```

☑ 5. Exception Handling – Try-Catch, Custom Exception

java

CopyEdit

```
// Example 1: Try-Catch
public class ExceptionDemo {
    public static void main(String[] args) {
        try {
            int a = 5 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero!");
        }
    }
}

// Example 2: Custom Exception
class InvalidAgeException extends Exception {
    public InvalidAgeException(String msg) { super(msg); }
}
class Voting {
    void checkAge(int age) throws InvalidAgeException {
        if (age < 18)
            throw new InvalidAgeException("Not eligible to vote!");
        else
            System.out.println("Eligible to vote");
    }
}

// Example 3: Finally Block
```

```
public class FinallyDemo {
    public static void main(String[] args) {
        try {
            int x = 10 / 2;
        } finally {
            System.out.println("This will always execute");
        }
    }
}
```

☑ 6. Multithreading – Threads, Synchronization

java

CopyEdit

```
// Example 1: Creating a Thread
class MyThread extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++)
            System.out.println("Thread running: " + i);
    }
}

// Example 2: Using Runnable Interface
class RunnableDemo implements Runnable {
    public void run() {
        System.out.println("Runnable thread is running");
    }
}

// Example 3: Synchronization
class SyncDemo {
    synchronized void printTable(int n) {
```

```
        for (int i = 1; i <= 5; i++)
            System.out.println(n * i);
    }
}
```

☑ 7. Collections – List, Set, Map

java

CopyEdit

```
// Example 1: List Example
import java.util.*;
class ListDemo {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        list.add("Java");
        list.add("Python");
        System.out.println("List: " + list);
    }
}

// Example 2: Set Example
class SetDemo {
    public static void main(String[] args) {
        Set<Integer> set = new HashSet<>();
        set.add(1); set.add(2); set.add(1); // No duplicates
        System.out.println("Set: " + set);
    }
}

// Example 3: Map Example
class MapDemo {
    public static void main(String[] args) {
```

```
        Map<Integer, String> map = new HashMap<>();
        map.put(1, "A");
        map.put(2, "B");
        System.out.println("Map: " + map);
    }
}
// ☑ JDBC Basics
import java.sql.*;

class JDBCDemo {
    public static void main(String[] args) throws Exception {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db", "root", "");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM students");
        while (rs.next()) {
            System.out.println(rs.getString(1));
        }
        con.close();
    }
}

class InsertStudent {
    public static void main(String[] args) throws Exception {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db", "root", "");
        String sql = "INSERT INTO students(name, age) VALUES (?, ?)";
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setString(1, "Mahendra");
        pstmt.setInt(2, 22);
        int rows = pstmt.executeUpdate();
        System.out.println(rows + " row(s) inserted.");
```

```
        con.close();
    }
}

class DeleteStudent {
    public static void main(String[] args) throws Exception {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db", "root", "");
        String sql = "DELETE FROM students WHERE name = ?";
        PreparedStatement pstmt = con.prepareStatement(sql);
        pstmt.setString(1, "Mahendra");
        int rows = pstmt.executeUpdate();
        System.out.println(rows + " row(s) deleted.");
        con.close();
    }
}

// ☑ Streams & Lambda Expressions
import java.util.*;

class LambdaFilter {
    public static void main(String[] args) {
        List<Integer> nums = Arrays.asList(10, 20, 30, 40);
        nums.stream().filter(n -> n >= 25).forEach(System.out::println);
    }
}

class LambdaSquare {
    public static void main(String[] args) {
        List<Integer> nums = Arrays.asList(1, 2, 3, 4);
        nums.forEach(n -> System.out.println(n * n));
    }
```

```
}

class LambdaSort {
    public static void main(String[] args) {
        List<String> names = Arrays.asList("Charlie", "Alice", "Bob");
        names.sort((a, b) -> a.compareTo(b));
        names.forEach(System.out::println);
    }
}
```