Java Lab Programs

1. Write a Java program to read an array of integers and sort it in ascending order.

```java
import java.util.Scanner;
public class ArraySort {
// Main method: program entry point
	public static void main(String args[]) {
		int a[], i, j, n;
		Scanner scanner = new Scanner(System.in);
		System.out.println("Enter the number of elements:");
		n = scanner.nextInt();
		a = new int[n];
		System.out.println("Enter " + n + " integers:");
		for (i = 0; i < n; i++) {
			a[i] = scanner.nextInt();
		}
		for (i = 0; i < n - 1; i++) {
			for (j = i + 1; j < n; j++) {
				if (a[i] > a[j]) {
					int temp = a[i];
					a[i] = a[j];
					a[j] = temp;
				}
			}
		}
		System.out.println("Sorted array in ascending order:");
		for (i = 0; i < n; i++) {
			System.out.println(a[i]);
		}
		scanner.close();
	}
}
```

Output:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> ArraySort [Java Application] C:\Users\NIKHITA\.p2\pool\plugins\or
Enter the number of elements:
5
Enter 5 integers:
25
5
7
3
1
Sorted array in ascending order:
1
3
5
7
25
```

2. Write a Java program to demonstrate the concept of a class and object.

```java
public class Student {
        String name;
        int rollNo;
        int age;
        float marks;

        // Method to display student information
        void displayInfo() {
                System.out.println("Student Name : " + name);
                System.out.println("Roll Number    : " + rollNo);
                System.out.println("Age            : " + age);
                System.out.println("Marks          : " + marks);
        }

        // Main method: program entry point
        public static void main(String[] args) {
                Student student1 = new Student();
                student1.name = "Aditya";
                student1.rollNo = 222123;
                student1.age = 20;
                student1.marks = 95.63f;
                student1.displayInfo();
        }
}
```

Output:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> Student [Java Application] C:\Users\NIKHITA\.p2\pool\p
Student Name    : Aditya
Roll Number     : 222123
Age             : 20
Marks           : 95.63
```

3. Write a Java program to demonstrate the concept of constructors.

```java
class Box {
	double width, height, depth;

	// Default Constructor
	Box() {
		width = height = depth = 0;
	}

	// Cube Constructor
	Box(double length) {
		width = height = depth = length;
	}

	// Parameterized Constructor
	Box(double w, double h, double d) {
		width = w;
		height = h;
		depth = d;
	}

	// Copy Constructor
	Box(Box obj) {
		width = obj.width;
		height = obj.height;
		depth = obj.depth;
	}

	// Method to calculate volume
	double volume() {
		return width * height * depth;
	}

	public class Test {
		public static void main(String[] args) {
			Box box1 = new Box();
			Box cube = new Box(7);
			Box box2 = new Box(15, 20, 15);
			Box box3 = new Box(box2); // Copy of box2

			System.out.println("Volume of box1 (default) is: " + box1.volume());
			System.out.println("Volume of cube (cube) is: " + cube.volume());
			System.out.println("Volume of box2 (parameterized) is: " + box2.volume());
			System.out.println("Volume of box3 (copy of box2) is: " + box3.volume());
		}
	}
}
```

Output:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> Student [Java Application] C:\Users\NIKHITA\.p2\pool\plugins\org.eclipse.justj.openjc
Volume of box1 (default) is: 0.0
Volume of cube (cube) is: 343.0
Volume of box2 (parameterized) is: 4500.0
Volume of box3 (copy of box2) is: 4500.0
```

4. Write a Java program to demonstrate method overloading.

```java
class MethodOverloading {
	// Method to find area of Circle
	void area(double radius) {
		System.out.println("Area of Circle is: " + (3.142 * radius * radius));
	}

	// Method to find area of Rectangle
	void area(float length, float breadth) {
		System.out.println("Area of Rectangle is: " + (length * breadth));
	}

	// Method to find area of Triangle
	void area(int base, int height) {
		System.out.println("Area of Triangle is: " + (0.5 * base * height));
	}

	public static void main(String[] args) {
		MethodOverloading mo = new MethodOverloading();
		mo.area(5.5);
		mo.area(5.5f, 6.8f);
		mo.area (10, 20);
	}
}
```

Output:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> Student [Java Application] C:\Users\NIKHITA\.p2\pool\plugins\org.eclipse.ju
Area of Circle is: 95.04549999999999
Area of Rectangle is: 37.4
Area of Triangle is: 100.0
```

5. Write a Java program to demonstrate method overriding.

```
class Employee {          // Parent class
    float salary = 40000;

    // Method to be overridden
    void increment() {
            System.out.println("Employee incremented salary: " + (salary + (salary * 0.2)));
    }
}
// Subclass
class PermanentEmployee extends Employee {
    double hike = 0.5;
    @Override
    void increment() {
            System.out.println("Permanent Employee incremented salary: " + (salary + (salary * hike)));
    }
}
// Subclass
class TemporaryEmployee extends Employee {
    double hike = 0.35;

    @Override
    void increment() {
            System.out.println("Temporary Employee incremented salary: " + (salary + (salary * hike)));
    }
}
// Main class
public class EmployeeSalaryHike {
    public static void main(String[] args) {

            Employee emp= new Employee();
            PermanentEmployee pemp= new PermanentEmployee();
            TemporaryEmployee temp= new TemporaryEmployee();
            emp.increment();
            pemp.increment();
            temp.increment();
    }
}
```

Output:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> EmployeeSalaryHike [Java Application] C:\Users\NIKHITA\.p2\pool\plugins\org.eclipse.justj.openjc
Employee incremented salary: 48000.0
Permanent Employee incremented salary: 60000.0
Temporary Employee incremented salary: 54000.0
```

6. Write a Java Program to demonstrate the working of 'this' and 'super' keyword

```
class Employee {
    private String name;
    private int employeeId;
    // Constructor using 'this' to refer to instance variables
    public Employee(String name, int employeeId) {
        this.name = name;
        this.employeeId = employeeId;
    }
    public void displayDetails() {
        System.out.println("Employee Name: " + name);
        System.out.println("Employee ID: " + employeeId);
    }
}
class Manager extends Employee {            //Subclass
    private String department;
    public Manager(String name, int employeeId, String department) {
        super(name, employeeId); // Call to Employee constructor
        this.department = department; // Refers to current class variable
    }
    public void displayManagerProfile() {
        super.displayDetails();
        System.out.println("Department: " + this.department);
    }
}
// Main class
public class CompanyDemo {
    public static void main(String[] args) {
        Manager mgr = new Manager("John", 2025, "Sales");
        mgr.displayManagerProfile();
    }
}
```

Output :

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> CompanyDemo [Java Application] C:\Users\NIKHITA
Employee Name: John
Employee ID: 2025
Department: Sales
```

7. Write a Java program to implement an inner class and demonstrate its access protection

```java
class Course {
	private String courseName = "Java Programming"; // Private member of outer class

	// Inner class
	public class Student {
		private String studentName;

		public Student(String name) {
			this.studentName = name;
		}

		public void displayDetails() {
			// Inner class accessing private member of outer class
			System.out.println(studentName + " is enrolled in " + courseName);
		}
	}

	// Method to create and use the inner class
	public void enrollStudent(String name) {
		Student s = new Student(name);
		s.displayDetails();
	}
}

// Main class
public class CourseDemo {
	public static void main(String[] args) {
		Course javaCourse = new Course();
		javaCourse.enrollStudent("Disha");
	}
}
```

Output :

```
Disha is enrolled in Java Programming
```

8. Write a Java program to demonstrate the working of multiple catch blocks.

```java
public class MultipleCatchDemo {
	public static void main(String[] args) {
		try {
			int a = 10;
			int b = 0;
			String text = null;
			int result = a / b;
			System.out.println("Result: " + result);
			System.out.println("Text length: " + text.length());
		} catch (ArithmeticException e) {
			System.out.println("Caught ArithmeticException: Cannot divide by zero.");
		} catch (NullPointerException e) {
			System.out.println("Caught NullPointerException: Something is null that shouldn't be.");
		} catch (Exception e) {
			System.out.println("Caught General Exception: " + e.getMessage());
		}
		System.out.println("Program continues after exception handling.");
	}
}
```

Output 1:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> MultipleCatchDemo [Java Application] C:\Users\NIKHITA\.p2\pool\plugins\org.eclipse.justj.openjdk.hots
Caught ArithmeticException: Cannot divide by zero.
Program continues after exception handling.
```

Output 2:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> MultipleCatchDemo [Java Application] C:\Users\NIKHITA\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.
Result: 2
Caught NullPointerException: Something is null that shouldn't be.
Program continues after exception handling.
```

Output 3:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> MultipleCatchDemo [Java Application] C:\Users\NIKHITA\.p2\pool\plugins\org.eclipse.justj.
Result: 2
Text length: 4
Program continues after exception handling.
```