DEWAN VS INSTITUTE OF ENGINEERING AND TECHNOLOGY, MEERUT

(Affiliated to Dr APJ Abdul Kalam Technical University, Lucknow)

NH58Bypass, Partapur, Meerut, UP, India

(DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING)

DEWAN VS GROUP OF INSTITUTIONS INDIA

OOPS with JAVA Lab File

(BCS-452)

B.TECH–2nd YEAR (EVEN SEM,2025-2026)

Name	
RollNo.	
Section-Batch	

Submitted to

(Assistant Professor)

Index

1. WAP in java to print Hello World

```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello, World");
   }
}
```

Output:
Hello, World

2. WAP to Add Two Numbers (User Input)

```
import java.util.Scanner;

class AddTwoNumbers {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the first number: ");
      int num1 = scanner.nextInt();

      System.out.print("Enter the second number: ");
      int num2 = scanner.nextInt();

      int sum = num1 + num2;
      System.out.println("The sum is: " + sum);
      scanner.close();
   }
}
```

Output:
Enter the first number: 10
Enter the second number: 25
The sum is: 35

3. WAP to Find Greatest of Three Numbers,take the input from user.

```
import java.util.Scanner;

class GreatestOfThree {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter first number: ");
      int a = scanner.nextInt();
      System.out.print("Enter second number: ");
      int b = scanner.nextInt();
      System.out.print("Enter third number: ");
      int c = scanner.nextInt();

      int greatest;
      if (a >= b && a >= c) {
         greatest = a;
      } else if (b >= a && b >= c) {
         greatest = b;
      } else {
         greatest = c;
      }
      System.out.println("The greatest number is: " + greatest);
      scanner.close();
   }
}
```

Output:
Enter first number: 12
Enter second number: 25
Enter third number: 17
The greatest number is: 25

4. Take a Matrix as input form user . Search for given X and print the indices at which it occurs.

```
import java.util.Scanner;

class SearchInMatrix {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter number of rows: ");
      int rows = scanner.nextInt();
      System.out.print("Enter number of columns: ");
      int cols = scanner.nextInt();

      int[][] matrix = new int[rows][cols];
      System.out.println("Enter the elements of the matrix:");
      for (int i = 0; i < rows; i++) {
         for (int j = 0; j < cols; j++) {
            matrix[i][j] = scanner.nextInt();
         }
      }

      System.out.print("Enter the number to search (X): ");
      int x = scanner.nextInt();

      boolean found = false;
      System.out.println("X found at positions:");
      for (int i = 0; i < rows; i++) {
         for (int j = 0; j < cols; j++) {
            if (matrix[i][j] == x) {
               System.out.println("Row: " + i + ", Column: " + j);
               found = true;
            }
         }
      }
      if (!found) {
         System.out.println("X not found in the matrix.");
      }
      scanner.close();
   }
}
```

Output:

```
Enter number of rows: 2
Enter number of columns: 3
1 2 3
4 5 3
Enter the number to search (X): 3
X found at positions:
Row: 0, Column: 2
Row: 1, Column: 2
```

5. WAP to explain Inheritance in Java (Single, Multi-level, Hierarchical, Multiple, Hybrid)

```
// Single Inheritance
class Animal {
   void eat() {
      System.out.println("Animal eats.");
   }
}

class Dog extends Animal {
   void bark() {
      System.out.println("Dog barks.");
   }
}

class SingleInheritance {
   public static void main(String[] args) {
      Dog dog = new Dog();
      dog.eat();
      dog.bark();
   }
}

// Multilevel Inheritance
class Puppy extends Dog {
   void weep() {
      System.out.println("Puppy weeps.");
   }
}

class MultilevelInheritance {
   public static void main(String[] args) {
      Puppy puppy = new Puppy();
```

```
        puppy.eat();
        puppy.bark();
        puppy.weep();
    }
}

// Hierarchical Inheritance
class Cat extends Animal {
    void meow() {
        System.out.println("Cat meows.");
    }
}

class HierarchicalInheritance {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();
        dog.bark();
        Cat cat = new Cat();
        cat.eat();
        cat.meow();
    }
}

// Multiple Inheritance via Interface
interface Printable {
    void print();
}

interface Showable {
    void show();
}

class A implements Printable, Showable {
    public void print() {
        System.out.println("Printing...");
    }
    public void show() {
        System.out.println("Showing...");
    }
}

class MultipleInheritance {
```

```
    public static void main(String[] args) {
        A obj = new A();
        obj.print();
        obj.show();
    }
}

// Hybrid Inheritance
interface B extends Printable {
    void methodB();
}

class C {
    void methodC() {
        System.out.println("Class C method.");
    }
}

class D extends C implements B {
    public void print() {
        System.out.println("Interface A method.");
    }
    public void methodB() {
        System.out.println("Interface B method.");
    }
}

class HybridInheritance {
    public static void main(String[] args) {
        D obj = new D();
        obj.print();
        obj.methodB();
        obj.methodC();
    }
}
```

Output:

```
Animal eats.
Dog barks.
Puppy weeps.
Animal eats.
Dog barks.
Animal eats.
```

Cat meows.
Printing...
Showing...
Interface A method.
Interface B method.
Class C method.

6. WAP to explain Polymorphism in Java (Overloading & Overriding)

```
// Method Overloading
class Calculator {
   int add(int a, int b) {
      return a + b;
   }

   int add(int a, int b, int c) {
      return a + b + c;
   }

   double add(double a, double b) {
      return a + b;
   }
}

class MethodOverloadingExample {
   public static void main(String[] args) {
      Calculator calc = new Calculator();
      System.out.println("Sum of 2 and 3: " + calc.add(2, 3));
      System.out.println("Sum of 2, 3 and 4: " + calc.add(2, 3, 4));
      System.out.println("Sum of 2.5 and 3.5: " + calc.add(2.5, 3.5));
   }
}

// Method Overriding
class Animal {
   void sound() {
      System.out.println("Animal makes a sound");
   }
}

class Dog extends Animal {
   @Override
   void sound() {
      System.out.println("Dog barks");
   }
}

class Cat extends Animal {
   @Override
   void sound() {
      System.out.println("Cat meows");
```

```
    }
}

class MethodOverridingExample {
    public static void main(String[] args) {
        Animal a;
        a = new Dog();
        a.sound();
        a = new Cat();
        a.sound();
    }
}
```

Output:
Sum of 2 and 3: 5
Sum of 2, 3 and 4: 9
Sum of 2.5 and 3.5: 6.0
Dog barks
Cat meows

7. WAP to explain Encapsulation in Java

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

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public int getAge() {
      return age;
   }

   public void setAge(int age) {
      if (age >= 0) {
         this.age = age;
      } else {
         System.out.println("Age cannot be negative!");
      }
   }
}

class EncapsulationExample {
   public static void main(String[] args) {
      Person p = new Person();
      p.setName("Alice");
      p.setAge(25);
      System.out.println("Name: " + p.getName());
      System.out.println("Age: " + p.getAge());
      p.setAge(-5);
   }
}
```

Output:
Name: Alice
Age: 25
Age cannot be negative!

8. WAP to explain Abstraction in Java

```
abstract class Phone {
   abstract void google();
   abstract void selfie();

   void call() {
      System.out.println("Calling...");
   }
}

class Samsung extends Phone {
   @Override
   void google() {
      System.out.println("Using Google Assistant on Samsung");
   }

   @Override
   void selfie() {
      System.out.println("Taking a selfie with Samsung's front camera");
   }
}

class PhoneTest {
   public static void main(String[] args) {
      Phone myPhone = new Samsung();
      myPhone.call();
      myPhone.google();
      myPhone.selfie();
   }
}
```

Output:
Calling...
Using Google Assistant on Samsung
Taking a selfie with Samsung's front camera

9. Create a Custom Exception Handling program ,take the input from user and if age is less than 18 ,then throw exception.

```
import java.util.Scanner;

class InvalidAgeException extends Exception {
  public InvalidAgeException(String message) {
    super(message);
  }
}

class AgeValidation {
  static void validateAge(int age) throws InvalidAgeException {
    if (age < 18) {
      throw new InvalidAgeException("Age must be 18 or above to proceed.");
    } else {
      System.out.println("Age is valid. Access granted!");
    }
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter your age: ");
    int age = scanner.nextInt();

    try {
      validateAge(age);
    } catch (InvalidAgeException e) {
      System.out.println("Exception caught: " + e.getMessage());
    }

    scanner.close();
  }
}
```

Output:
Enter your age: 16
Exception caught: Age must be 18 or above to proceed.

10. WAP to create ArrayList Operations in Java

```
import java.util.*;

class Example {
   public static void main(String[] args) {
      ArrayList<String> fruits = new ArrayList<>();
      fruits.add("apple");
      fruits.add("banana");
      fruits.add("mango");
      fruits.add("orange");

      System.out.println(fruits);

      fruits.add(2, "grapes");
      System.out.println(fruits);

      System.out.println(fruits.get(1));

      fruits.set(1, "kiwi");
      System.out.println(fruits);

      fruits.remove("kiwi");
      System.out.println(fruits);

      fruits.remove(0);
      System.out.println(fruits);

      System.out.println(fruits.contains("apple"));
      System.out.println(fruits.contains("orange"));

      for (String fruit : fruits) {
         System.out.println(fruit);
      }

      Iterator<String> it = fruits.iterator();
      while (it.hasNext()) {
         System.out.println(it.next());
      }

      Collections.sort(fruits);
      System.out.println(fruits);

      Collections.reverse(fruits);
```

```
        System.out.println(fruits);

        fruits.clear();
        System.out.println(fruits);

        System.out.println(fruits.isEmpty());
    }
}
```

Output:
[apple, banana, mango, orange]...
[orange, mango, grapes] []
true

1. WAP in java to print Hello World

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

```
Output:
Hello, World
```

2. WAP to Add Two Numbers (User Input)

```
import java.util.Scanner;

class AddTwoNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the first number: ");
        int num1 = scanner.nextInt();

        System.out.print("Enter the second number: ");
        int num2 = scanner.nextInt();

        int sum = num1 + num2;
        System.out.println("The sum is: " + sum);
        scanner.close();
    }
}
```

```
Output:
Enter the first number: 10
Enter the second number: 25
The sum is: 35
```

3. WAP to Find Greatest of Three Numbers,take the input from user.

```
import java.util.Scanner;

class GreatestOfThree {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter first number: ");
        int a = scanner.nextInt();
        System.out.print("Enter second number: ");
        int b = scanner.nextInt();
        System.out.print("Enter third number: ");
        int c = scanner.nextInt();

        int greatest;
        if (a >= b && a >= c) {
            greatest = a;
        } else if (b >= a && b >= c) {
            greatest = b;
        } else {
            greatest = c;
        }
        System.out.println("The greatest number is: " + greatest);
        scanner.close();
    }
}


Output:
Enter first number: 12
Enter second number: 25
Enter third number: 17
The greatest number is: 25
```

4. Take a Matrix as input form user . Search for given X and print the indices at which it occurs.

```
import java.util.Scanner;

class SearchInMatrix {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter number of rows: ");
        int rows = scanner.nextInt();
        System.out.print("Enter number of columns: ");
        int cols = scanner.nextInt();

        int[][] matrix = new int[rows][cols];
        System.out.println("Enter the elements of the matrix:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }

        System.out.print("Enter the number to search (X): ");
        int x = scanner.nextInt();

        boolean found = false;
        System.out.println("X found at positions:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (matrix[i][j] == x) {
                    System.out.println("Row: " + i + ", Column: " + j);
                    found = true;
                }
            }
        }
        if (!found) {
            System.out.println("X not found in the matrix.");
        }
        scanner.close();
    }
}
```

```
Output:
Enter number of rows: 2
Enter number of columns: 3
1 2 3
4 5 3
Enter the number to search (X): 3
X found at positions:
```

```
Row: 0, Column: 2
Row: 1, Column: 2
```

5. WAP to explain Inheritance in Java (Single, Multi-level, Hierarchical, Multiple, Hybrid)

```
// Single Inheritance
class Animal {
    void eat() {
        System.out.println("Animal eats.");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println("Dog barks.");
    }
}

class SingleInheritance {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();
        dog.bark();
    }
}

// Multilevel Inheritance
class Puppy extends Dog {
    void weep() {
        System.out.println("Puppy weeps.");
    }
}

class MultilevelInheritance {
    public static void main(String[] args) {
        Puppy puppy = new Puppy();
        puppy.eat();
        puppy.bark();
        puppy.weep();
    }
}

// Hierarchical Inheritance
class Cat extends Animal {
    void meow() {
        System.out.println("Cat meows.");
    }
}
```

```
class HierarchicalInheritance {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();
        dog.bark();
        Cat cat = new Cat();
        cat.eat();
        cat.meow();
    }
}

// Multiple Inheritance via Interface
interface Printable {
    void print();
}

interface Showable {
    void show();
}

class A implements Printable, Showable {
    public void print() {
        System.out.println("Printing...");
    }
    public void show() {
        System.out.println("Showing...");
    }
}

class MultipleInheritance {
    public static void main(String[] args) {
        A obj = new A();
        obj.print();
        obj.show();
    }
}

// Hybrid Inheritance
interface B extends Printable {
    void methodB();
}

class C {
    void methodC() {
        System.out.println("Class C method.");
    }
}
```

```
class D extends C implements B {
    public void print() {
        System.out.println("Interface A method.");
    }
    public void methodB() {
        System.out.println("Interface B method.");
    }
}

class HybridInheritance {
    public static void main(String[] args) {
        D obj = new D();
        obj.print();
        obj.methodB();
        obj.methodC();
    }
}
```

```
Output:
Animal eats.
Dog barks.
Puppy weeps.
Animal eats.
Dog barks.
Animal eats.
Cat meows.
Printing...
Showing...
Interface A method.
Interface B method.
Class C method.
```

6. WAP to explain Polymorphism in Java (Overloading & Overriding)

```
// Method Overloading
class Calculator {
    int add(int a, int b) {
        return a + b;
    }

    int add(int a, int b, int c) {
        return a + b + c;
    }

    double add(double a, double b) {
        return a + b;
    }
}

class MethodOverloadingExample {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("Sum of 2 and 3: " + calc.add(2, 3));
        System.out.println("Sum of 2, 3 and 4: " + calc.add(2, 3, 4));
        System.out.println("Sum of 2.5 and 3.5: " + calc.add(2.5,
3.5));
    }
}

// Method Overriding
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}

class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}

class MethodOverridingExample {
    public static void main(String[] args) {
```

```
        Animal a;
        a = new Dog();
        a.sound();
        a = new Cat();
        a.sound();
    }
}
```

```
Output:
Sum of 2 and 3: 5
Sum of 2, 3 and 4: 9
Sum of 2.5 and 3.5: 6.0
Dog barks
Cat meows
```

7. WAP to explain Encapsulation in Java

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        if (age >= 0) {
            this.age = age;
        } else {
            System.out.println("Age cannot be negative!");
        }
    }
}

class EncapsulationExample {
    public static void main(String[] args) {
        Person p = new Person();
        p.setName("Alice");
        p.setAge(25);
        System.out.println("Name: " + p.getName());
        System.out.println("Age: " + p.getAge());
        p.setAge(-5);
    }
}
```

```
Output:
Name: Alice
Age: 25
Age cannot be negative!
```

8. WAP to explain Abstraction in Java

```
abstract class Phone {
    abstract void google();
    abstract void selfie();

    void call() {
        System.out.println("Calling...");
    }
}

class Samsung extends Phone {
    @Override
    void google() {
        System.out.println("Using Google Assistant on Samsung");
    }

    @Override
    void selfie() {
        System.out.println("Taking a selfie with Samsung's front
camera");
    }
}

class PhoneTest {
    public static void main(String[] args) {
        Phone myPhone = new Samsung();
        myPhone.call();
        myPhone.google();
        myPhone.selfie();
    }
}


Output:
Calling...
Using Google Assistant on Samsung
Taking a selfie with Samsung's front camera
```

9. Create a Custom Exception Handling program ,take the input from user and if age is less than 18 ,then throw exception.

```
import java.util.Scanner;

class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}

class AgeValidation {
    static void validateAge(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("Age must be 18 or above to
proceed.");
        } else {
            System.out.println("Age is valid. Access granted!");
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        try {
            validateAge(age);
        } catch (InvalidAgeException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }

        scanner.close();
    }
}


Output:
Enter your age: 16
Exception caught: Age must be 18 or above to proceed.
```

10. WAP to create ArrayList Operations in Java

```
import java.util.*;

class Example {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("apple");
        fruits.add("banana");
        fruits.add("mango");
        fruits.add("orange");

        System.out.println(fruits);

        fruits.add(2, "grapes");
        System.out.println(fruits);

        System.out.println(fruits.get(1));

        fruits.set(1, "kiwi");
        System.out.println(fruits);

        fruits.remove("kiwi");
        System.out.println(fruits);

        fruits.remove(0);
        System.out.println(fruits);

        System.out.println(fruits.contains("apple"));
        System.out.println(fruits.contains("orange"));

        for (String fruit : fruits) {
            System.out.println(fruit);
        }

        Iterator<String> it = fruits.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }

        Collections.sort(fruits);
        System.out.println(fruits);

        Collections.reverse(fruits);
        System.out.println(fruits);

        fruits.clear();
        System.out.println(fruits);

        System.out.println(fruits.isEmpty());
```

```
    }
}
```

```
Output:
[apple, banana, mango, orange]...
[orange, mango, grapes] []
true
```