**1.a. Write a JAVA program to display default value of all primitive data type of JAVA**

```
public class DefaultValues
{
	// Declare all primitive data types
	static byte byteVar;
	static short shortVar;
	static int intVar;
	static long longVar;
	static float floatVar;
	static double doubleVar;
	static char charVar;
	static boolean booleanVar;
	public static void main(String[] args)
	{
		// Display the default values
		System.out.println("Default value of byte: " + byteVar);
		System.out.println("Default value of short: " + shortVar);
		System.out.println("Default value of int: " + intVar);
		System.out.println("Default value of long: " + longVar);
		System.out.println("Default value of float: " + floatVar);
		System.out.println("Default value of double: " + doubleVar);
		System.out.println("Default value of char: '"+ charVar+" ' ");
		System.out.println("Default value of boolean: " + booleanVar);
	}
}
```

**Output:**

```
C:\javap>javac DefaultValues.java
C:\javap>java DefaultValues
Default value of byte: 0
Default value of short: 0
Default value of int: 0
Default value of long: 0
```

Default value of float: 0.0

Default value of double: 0.0

Default value of char: ' '

Default value of boolean: false

**1.b. Write a JAVA program that display the roots of a quadratic equation $ax^2+bx=0$. Calculate the discriminate D basing on value of D, describe the nature of root.**

```
public class QuadraticEquation
{
	public static void main(String[] args)
	{
		// Create a Scanner object for input
		Scanner scanner = new Scanner(System.in);
		// Input coefficients a, b, and c
		System.out.print("Enter the coefficient a: ");
		double a = scanner.nextDouble();
		System.out.print("Enter the coefficient b: ");
		double b = scanner.nextDouble();
		System.out.print("Enter the coefficient c: ");
		double c = scanner.nextDouble();
		// Calculate the discriminant D
		double discriminant = b * b - 4 * a * c;
		// Determine the nature of the roots based on the discriminant
		if (discriminant > 0)
		{
			// Two real and distinct roots
			double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
			double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
			System.out.println("The equation has two real and distinct roots.");
			System.out.println("Root 1: " + root1);
			System.out.println("Root 2: " + root2);
		}
		else if (discriminant == 0)
		{
			// Two real and equal roots
			double root = -b / (2 * a);
			System.out.println("The equation has two real and equal roots.");
			System.out.println("Root: " + root);
		}
		else
		{
			// Complex roots
			double realPart = -b / (2 * a);
			double imaginaryPart = Math.sqrt(-discriminant) / (2 * a);
			System.out.println("The equation has complex roots.");
			System.out.println("Root 1: " + realPart + " + " + imaginaryPart + "i");
			System.out.println("Root 2: " + realPart + " - " + imaginaryPart + "i");
		}
		// Close the scanner
		scanner.close();
	}
}
```

**Output:**

```
C:\javap>javac QuadraticEquation.java
```

```
C:\javap>java QuadraticEquation
Enter the coefficient a: 1
Enter the coefficient b: -3
Enter the coefficient c: 2
The equation has two real and distinct roots.
Root 1: 2.0
Root 2: 1.0
```

**2.a. Write a JAVA program tosearch for an element in a given list of elements using binary search mechanism.**

```
import java.io.*;
import java.util.*;
public class Binss
{
	public static void main(String args[]) throws Exception
	{
		int a[]=new int[10];
		int i,k,n,f;
		Scanner sc = new Scanner(System.in);
		System.out.println("enter no.of elements:");
		n = sc.nextInt();
		System.out.println("enter elements:");
		for(i=0;i<n;i++)
			a[i] = sc.nextInt();
		for(i=0;i<n;i++)
		{
			for(int j=i;j<n;j++)
			{
				if(a[i]>a[j])
				{
					int t=a[i];
					a[i]=a[j];
					a[j]=t;
				}
			}
		}
		System.out.println("sorted elements:");
		for(i=0;i<n;i++)
			System.out.println(a[i]);
		System.out.println("enter key element:");
		k = sc.nextInt();
		bins(a,n,k);
	}
	static int bins(int a[],int n,int k)
	{
		int last = n-1,f=0;
		int first = 0;
		int mid;
		while(first <= last)
		{
			mid=(first+last)/2;
			if(a[mid]<k)
				first=mid+1;
			else
				if(a[mid]>k)
					last=mid-1;
			else
```

```
                        if(a[mid]==k)
                        {
                                System.out.println("the element is found at "+(mid+1));
                                f=1;
                                break;
                        }
                }
                if(f==0)
                        System.out.println("the element is not found ");
                        return 0;
        }
}
```

**Output:**

```
C:\javap>javac Binss.java
C:\javap>java Binss
enter no.of elements:
5
enter elements:
8
9
6
4
7
sorted elements:
4
6
7
8
9
enter key element:
7
the element is found at3
```

**2.b. Write a JAVA program tosort for an element in a given list of elements using bubble sort.**

```
import java.util.Scanner;
public class BubbleSort
{
	public static void main(String[] args)
	{
		// Create a Scanner object for input
		Scanner scanner = new Scanner(System.in);
		// Input the number of elements in the array
		System.out.print("Enter the number of elements in the array: ");
		int n = scanner.nextInt();
		// Create an array and input elements
		int[] array = new int[n];
		System.out.println("Enter " + n + " elements: ");
		for (int i = 0; i< n; i++)
		{
			array[i] = scanner.nextInt();
		}
		// Perform bubble sort
		bubbleSort(array);
		// Display the sorted array
		System.out.println("Sorted array:");
		for (int i = 0; i< n; i++)
		{
			System.out.print(array[i] + " ");
		}
		// Close the scanner
		scanner.close();
	}
	// Bubble sort function
	public static void bubbleSort(int[] array)
	{
		int n = array.length;
		for (int i = 0; i< n - 1; i++)
		{
			for (int j = 0; j < n - 1 - i; j++)
			{
				if (array[j] >array[j + 1])
				{
					// Swap array[j] and array[j + 1]
					int temp = array[j];
					array[j] = array[j + 1];
					array[j + 1] = temp;
				}
			}
		}
	}
}
```

**Output:**

```
C:\javap>javac BubbleSort.java
C:\javap>java BubbleSort
Enter the number of elements in the array: 5
Enter 5 elements:
5
9
7
2
3
Sorted array:
2 3 5 7 9
```

**2.c. Write a JAVA program using StringBuffer to delete, remove character.**

```
public class StringBufferEx
{
	public static void main(String[] args)
	{
		// Create a StringBuffer object with an initial string
		StringBuffer sb = new StringBuffer("Hello, World!");

		// Display the original string
		System.out.println("Original string: " + sb);

		// Delete a character at a specific index
		sb.deleteCharAt(5); // Delete the comma at index 5
		System.out.println("After deleting character at index 5: " + sb);

		// Delete a substring from the string
		sb.delete(6, 11); // Delete the substring "World" (index 6 to 10)
		System.out.println("After deleting substring from index 6 to 10: " + sb);

		// Insert a new substring
		sb.insert(6, "Java"); // Insert "Java" at index 6
		System.out.println("After inserting 'Java' at index 6: " + sb);

		// Replace a substring
		sb.replace(0, 5, "Hi"); // Replace "Hello" with "Hi"
		System.out.println("After replacing 'Hello' with 'Hi': " + sb);
	}
}
```

**Output:**

```
C:\javap>javac StringBufferEx.java
C:\javap>java StringBufferEx
Original string: Hello, World!
After deleting character at index 5: Hello World!
After deleting substring from index 6 to 10: Hello !
After inserting 'Java' at index 6: Hello Java!
After replacing 'Hello' with 'Hi': Hi Java!
```

## EXCERCISE-3

3A)WRITE A JAVA PROGRAM TO IMPLEMENT CLASS MECHANISM. CREATE A CLASS , METHODS AND INVOKE THEM INSIDE MAIN METHOD,

### Program:

```
class Example
 {

   String name;
   int age;

   // Constructor to initialize the Person object
   Example(String name, int age)
         {
      this.name = name;
      this.age = age;
   }

   // Method to display the person's information
   void displayInfo()
         {
      System.out.println("Name: " + name);
      System.out.println("Age: " + age);
   }


   void introduce()
         {
      System.out.println("Hello, my name is " + name + " and I am " + age + " years old.");
   }
}


public class Main
 {
   public static void main(String[] args)
         {
      Example person1 = new Example("Alice", 30);
      person1.displayInfo();
      person1.introduce();
          }
 }
```

### OUTPUT:

```
Name: Alice
Age: 30
Hello, my name is Alice and I am 30 years old.
```

## 3 b) Write a java program implement method overloading

### PROGRAM:

```
public class Overloading
{

   // Method with one integer parameter
   public void display(int a)
         {
      System.out.println("Integer: " + a);
   }

   // Method with two integer parameters
   public void display(int a, int b)
         {
      System.out.println("Integers: " + a + ", " + b);
   }

   // Method with one double parameter
   public void display(double a)
         {
      System.out.println("Double: " + a);
   }

   // Method with one string parameter
   public void display(String a)
         {
      System.out.println("String: " + a);
   }

   public static void main(String[] args)
         {
      Overloading  obj = new Overloading();

      // Calling the overloaded methods
      obj.display(10);              // Calls the method with one int parameter
      obj.display(10, 20);          // Calls the method with two int parameters
      obj.display(10.5);            // Calls the method with one double parameter
      obj.display("Hello, World!");  // Calls the method with one String parameter
   }
}
```

### OUTPUT:

```
Integer: 10
Integers: 10, 20
Double: 10.5
String: Hello, World!
```

3C) WRITE A JAVA PROGRAM TO IMPLEMENT CONSTRUCTOR.

## PROGRAM:

```
public class Person
 {
   // Fields
   private String name;
   private int age;

   // Default Constructor
   public Person()
         {
      this.name = "Unknown";
      this.age = 0;
      System.out.println("Default constructor called.");
   }

   // Parameterized Constructor
   public Person(String name, int age)
         {
      this.name = name;
      this.age = age;
      System.out.println("Parameterized constructor called.");
   }

   // Method to display person's details
   public void displayInfo()
         {
      System.out.println("Name: " + name);
      System.out.println("Age: " + age);
   }

   public static void main(String[] args)
         {
      // Creating objects using different constructors
      Person person1 = new Person(); // Calls the default constructor
      Person person2 = new Person("Alice", 30); // Calls the parameterized constructor

      // Displaying information
      System.out.println("Person 1:");
      person1.displayInfo();

      System.out.println("\nPerson 2:");
      person2.displayInfo();
   }
}
```

## Output:

```
Default constructor called.
Parameterized constructor called.
Person 1:
Name: Unknown
Age: 0

Person 2:
Name: Alice
Age: 30
```

## 3D)WRITE A JAVA PROGRAM TO IMPLEMENT CONSTRUCTOR OVERLOADING.

### Program:

```
class Book
 {

   String title;
   String author;
   double price;
   int pages;

   // Constructor with all parameters
   Book(String title, String author, double price, int pages)
         {
      this.title = title;
      this.author = author;
      this.price = price;
      this.pages = pages;
   }

   // Constructor with title and author only
   Book(String title, String author)
         {
      this.title = title;
      this.author = author;
      this.price = 0.0; // default price
      this.pages = 0;   // default pages
   }

   // Constructor with title only
   Book(String title)
         {
      this.title = title;
      this.author = "Unknown"; // default author
      this.price = 0.0;       // default price
      this.pages = 0;         // default pages
   }

   // Method to display book information
   void displayInfo()
         {
      System.out.println("Title: " + title);
      System.out.println("Author: " + author);
      System.out.println("Price: $" + price);
      System.out.println("Pages: " + pages);
```

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

public class Constructoroverloading
 {
    public static void main(String[] args)
            {
        // Create Book objects using different constructors
        Book book1 = new Book("Effective Java", "Joshua Bloch", 45.50, 416);
        Book book2 = new Book("Clean Code", "Robert C. Martin");
        Book book3 = new Book("Java: The Complete Reference");

        // Display information of each book
        book1.displayInfo();
        book2.displayInfo();
        book3.displayInfo();
    }
}
```

## OUTPUT:

```
Title: Effective Java
Author: Joshua Bloch
Price: $45.5
Pages: 416

Title: Clean Code
Author: Robert C. Martin
Price: $0.0
Pages: 0

Title: Java: The Complete Reference
Author: Unknown
Price: $0.0
Pages: 0
```

# EXCERCISE-4

## WRITE A JAVA PROGRAM TO IMPLEMENT SINGLE INHERITANCE.

### PROGRAM:

```
// Parent class (Super Class)
class Animal
 {
   // Method in the parent class
   void eat()
         {
      System.out.println("This animal eats food.");
   }

   void sleep()
         {
      System.out.println("This animal sleeps.");
   }
}

// Child class (Sub Class) that inherits from Animal
class Dog extends Animal
 {

   void bark()
         {
      System.out.println("The dog barks.");
   }
}

// Main class to test inheritance
public class SingleInheritance
 {
   public static void main(String[] args)
         {
      Dog myDog = new Dog();
      myDog.eat();
      myDog.sleep();
      myDog.bark();
   }
}
```

## OUTPUT:

```
This animal eats food.
This animal sleeps.
The dog barks
```

## 4B)WRITE A JAVA PROGRAM TO IMPLEMENT MULTI LEVEL INHERITANCE.

### PROGRAM:

```
// Grandparent class
class Grandparent
{
   void showGrandparent()
                                {
      System.out.println("This is the grandparent class.");
   }
}

// Parent class that inherits from Grandparent
class Parent extends Grandparent
 {
   void showParent()
                                {
      System.out.println("This is the parent class.");
   }
}

// Child class that inherits from Parent
class Child extends Parent
 {
   void showChild()
                                {
      System.out.println("This is the child class.");
   }
}
```

```
// Main class to test multilevel inheritance
public class MultilevelInheritance
{
    public static void main(String[] args)
{
   Child myChild = new Child();
   myChild.showGrandparent();
    myChild.showParent();
    myChild.showChild();
}
}
```

**OUTPUT:**

```
This is the grandparent class.
This is the parent class.
This is the child class.
```

## 4c) write a JAVA program for abstract class to find areas of different shapes.

### Program:

```
// Abstract class Shape
abstract class Shape
{
   abstract double area();
}

class Circle extends Shape
 {
   private double radius;

   public Circle(double radius)
{
      this.radius = radius;
   }

   double area()
{
      return Math.PI * radius * radius;
   }
}

class Rectangle extends Shape
{
   private double width;
   private double height;
   public Rectangle(double width, double height)
```

```
{
        this.width = width;
        this.height = height;
    }

    double area()
{
        return width * height;
    }
}

// Concrete class Triangle that extends Shape
class Triangle extends Shape
 {
    private double base;
    private double height;

    // Constructor for Triangle
    public Triangle(double base, double height)
{
        this.base = base;
        this.height = height;
    }
    double area()
{
        return 0.5 * base * height;
    }
}

// Main class to test the area calculation
public class DifferentShapes
```

```
{
   public static void main(String[] args)
{
      // Create instances of different shapes
      Shape circle = new Circle(5.0);
      Shape rectangle = new Rectangle(4.0, 6.0);
      Shape triangle = new Triangle(3.0, 7.0);

      // Calculate and display the area of each shape
      System.out.println("Area of the Circle: " + circle.area());
      System.out.println("Area of the Rectangle: " + rectangle.area());
      System.out.println("Area of the Triangle: " + triangle.area());
   }
}
```

**OUTPUT:**

Area of the Circle: 78.53981633974483

Area of the Rectangle: 24.0

Area of the Triangle: 10.5

## EXPERIMENT-5

**AIM: WRITE A JAVA PROGRAM GIVE EXAMPLE FOR"SUPER"KEYWORD.**

**PROGRAM:**

```
class Animal
{
   String name;
   Animal(String name)
   {
      this.name = name;
    }
 // Method of the superclass
   void speak()
    {
      System.out.println("Animal speaks");
    }
}
class Dog extends Animal
 {
   String breed;
   Dog(String name, String breed)
     {
      // Call the superclass constructor
      super(name);
      this.breed = breed;
      }
   void speak()
      {
      // Call the superclass method
      super.speak();
      System.out.println("Dog barks");
    }
```

```
    void displayInfo()
     {
       System.out.println("Name: " + name);
       System.out.println("Breed: " + breed);
     }
}
public class TestSuper
 {
    public static void main(String[] args)
     {
       Dog myDog = new Dog("Buddy", "Golden Retriever");
       myDog.speak();
       myDog.displayInfo();
     }
}
```

**OUTPUT:**

```
Animal speaks
Dog barks
Name: Buddy
Breed: Golden Retriever
```

## 5B)WRITE A JAVA PROGRAM GIVE EXAMPLE OF "FINAL"KEYWORD.

### PROGRAM:

```
class FinalVariableExample
{
   // Declare a final variable
   public static final int MAX_ATTEMPTS = 5;
      public void showMaxAttempts()
    {
       System.out.println("Maximum attempts allowed: " + MAX_ATTEMPTS);
     }
}
class Parent
 {
   // Declare a final method
   public final void display()
      {
      System.out.println("This is a final method from the Parent class.");
     }
}

final class Main
{
   public void printMessage()
    {
       System.out.println("This is a final class.");
     }
}
public class TestFinal
{
   public static void main(String[] args)
{
```

```
        // Test final variable
        FinalVariableExample example = new FinalVariableExample();
        example.showMaxAttempts();

        // Test final method
        Parent parent = new Parent();
        parent.display();

        // Test final class
        Main finalClass = new Main();
        finalClass.printMessage();
    }
}
```

**OUTPUT:**

```
Maximum attempts allowed: 5
This is a final method from the Parent class.
This is a final class.
```

## AIM:WRITE A JAVA PROGRAM TO IMPLEMENT INTERFACE.

## PROGRAM:

```
interface Student
{
   void year();
}


class College implements Student
{
   public void year()
{
      System.out.println("II year student");
   }
}
class University  implements Student
{
   public void year()
   {
      System.out.println("CSE  students");
    }
}
public class Example1
 {
   public static void main(String[] args)
{
      Student college = new College();
      Student university = new University();
       college.year();
       university.year();
   }
}
```

OUTPUT:

II year student

CSE students.

## 5D)WRITE A JAVA PROGRAM TO IMPLEMENT RUNTIME POLYMORPHISUM PROGRAM:

```
class Animal
{
   void sound()
{
      System.out.println("Animal makes a sound");
   }
}
class Dog extends Animal
 {
   void sound()
{
      System.out.println("Dog barks");
   }
}
class Cat extends Animal
{
   void sound()
  {
 System.out.println("Cat meows");
   }
}
public class RuntimePolymorphisumExample
{
```

```
    public static void main(String[] args)
{
        Animal myDog = new Dog();
        Animal myCat = new Cat();
        myDog.sound();
        myCat.sound();
        Animal[] animals = {myDog, myCat};

        for (Animal animal : animals)
 {
            animal.sound();
        }
    }
}
```

OUTPUT:

```
Dog barks
Cat meows
Dog barks
Cat meows
```

## Excercise-6

### A) AIM:Write a JAVA program that describes exception handling mechanisum.

### Program:

```
public class ExceptionHandlingExample
 {
   public static void main(String[] args)
{
       ExceptionHandlingExample example = new ExceptionHandlingExample();
       example.divideNumbers(10, 0);
       example.divideNumbers(10, 2);
   }

   public void divideNumbers(int numerator, int denominator)
{
       try
     {
          int result = numerator / denominator;
          System.out.println("Result: " + result);
       }
       catch (ArithmeticException e)
     {
        System.out.println("Error: Cannot divide by zero.");
       }
     finally
     {
          System.out.println("Execution of divideNumbers method completed.");
       }
   }
}
```

OUTPUT:

Result: 2

Execution of divideNumbers method completed.

Result: 5

Execution of divideNumbers method completed

## B)Write a JAVA Program Illustrate Multiple catch clauses.

### Program:

```
import java.util.Scanner;

public class MultipleCatchExample
{
   public static void main(String[] args)
{
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter two numbers to divide:");
    try
  {
         System.out.print("Numerator: ");
         int numerator = scanner.nextInt();
         System.out.print("Denominator: ");
         int denominator = scanner.nextInt();
         int result = numerator / denominator;
         System.out.println("Result: " + result);
      }
     catch (ArithmeticException e)
      {
         System.out.println("Error: Cannot divide by zero.");
      }
```

```
     catch (java.util.InputMismatchException e)
      {
          System.out.println("Error: Please enter valid integers.");
      }
    catch (Exception e)
      {
          System.out.println("Error: An unexpected error occurred.");
      }
      finally
       {
          System.out.println("Execution completed.");
          scanner.close();
      }
   }
}
```

## OUTPUT:

```
Enter two numbers to divide:
Numerator: 80
Denominator: 4
Result: 20
Execution completed
Enter two numbers to divide:
Numerator: 89
Denominator: 0
Error: Cannot divide by zero.
Execution completed.
Enter two numbers to divide:
Numerator: 89.000
Error: Please enter valid integers.
Execution completed.
```

## c) Write a JAVA Program for creation of java Built-in Exceptions.

### Program:

```
public class BuiltInExceptionExample
{
public static void main(String[] args)
{
  try
    {
        String str = null;
        System.out.println("Length of the string: " + str.length());
     }
   catch (NullPointerException e)
    {
        System.out.println("Error: NullPointerException - Attempted to call a method on a null object.");
     }
  try
    {
        int[] numbers = {1, 2, 3};
        System.out.println("Accessing invalid index: " + numbers[5]); // This will throw ArrayIndexOutOfBoundsException
     }
  catch (ArrayIndexOutOfBoundsException e)
     {
        System.out.println("Error: ArrayIndexOutOfBoundsException - Invalid index accessed.");
     }

     try
     {
        String invalidNumber = "abc";
        int number = Integer.parseInt(invalidNumber);
        System.out.println("Parsed number: " + number);
     }
```

```
    catch (NumberFormatException e)
      {
          System.out.println("Error: NumberFormatException - Unable to convert string to an integer.");
       }
          System.out.println("Execution completed.");
    }
}
```

**Output:**

Error: NullPointerException - Attempted to call a method on a null object.

Error: ArrayIndexOutOfBoundsException - Invalid index accessed.

Error: NumberFormatException - Unable to convert string to an integer.

Execution completed.

## D) Write a JAVA Program for creation of User Defined Exception.

**Program:**

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

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

```
            checkAge(151);
        }
        catch (InvalidAgeException e)
        {
            System.out.println("Caught Exception: " + e.getMessage());
        }

        try
        {
            checkAge(25);
            System.out.println("Age is valid.");
        }
        catch (InvalidAgeException e)
        {
            System.out.println("Caught Exception: " + e.getMessage());
        }

        System.out.println("Execution completed.");
    }
    public static void checkAge(int age) throws InvalidAgeException
{
        if (age < 0 || age > 150)
        {
            throw new InvalidAgeException("Age must be between 0 and 150.");
        }
    }
}
```

**OUTPUT:**

Caught Exception: Age must be between 0 and 150.

Age is valid.

Execution completed.