Program1:

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

```
public class DefaultValues {
   // Declare fields for each primitive data type
   byte defaultByte;
   short defaultShort;
   int defaultInt;
   long defaultLong;
   float defaultFloat;
   double defaultDouble;
   char defaultChar;
   boolean defaultBoolean;

   public static void main(String[] args) {
      // Create an instance of the DefaultValues class
      DefaultValues defaults = new DefaultValues();

      // Print the default values of each field
      System.out.println("Default byte: " + defaults.defaultByte);
      System.out.println("Default short: " + defaults.defaultShort);
      System.out.println("Default int: " + defaults.defaultInt);
      System.out.println("Default long: " + defaults.defaultLong);
      System.out.println("Default float: " + defaults.defaultFloat);
      System.out.println("Default double: " + defaults.defaultDouble);
      System.out.println("Default char: '" + defaults.defaultChar + "'");
      System.out.println("Default boolean: " + defaults.defaultBoolean);
   }
}
```

Output:

```
Default byte: 0
Default short: 0
Default int: 0
Default long: 0
Default float: 0.0
Default double: 0.0
Default char: '
```

Program 2:

Write a java program that display the roots of a quadratic equation ax2+bx+c=0. Calculate the discriminate D and basing on value of D, describe the nature of root.

```
import java.util.Scanner;

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

      // Input coefficients
      System.out.println("Enter coefficient a: ");
      double a = sc.nextDouble();

      System.out.println("Enter coefficient b: ");
      double b = sc.nextDouble();

      System.out.println("Enter coefficient c: ");
      double c = sc.nextDouble();

      // Calculate the discriminant
      double D = b * b - 4 * a * c;
      System.out.println("The discriminant (D) is: " + D);

      // Determine the nature of the roots
      if (D > 0) {
         // Two distinct real roots
         double root1 = (-b + Math.sqrt(D)) / (2 * a);
         double root2 = (-b - Math.sqrt(D)) / (2 * a);
         System.out.println("The equation has two distinct real roots:");
         System.out.println("Root 1: " + root1);
         System.out.println("Root 2: " + root2);
      } else if (D == 0) {
         // One real root (double root)
         double root = -b / (2 * a);
         System.out.println("The equation has twp equal real roots: " + root);
      } else {
         // Complex roots
         double realPart = -b / (2 * a);
         double imaginaryPart = Math.sqrt(-D) / (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");
      }
```

```
        sc.close();
    }
}
```

Output1:

```
Enter coefficient a:
1
Enter coefficient b:
-5
Enter coefficient c:
6
The discriminant (D) is: 1.0
The equation has two distinct real roots:
Root 1: 3.0
Root 2: 2.0
```

Output2:

```
Enter coefficient a:
1
Enter coefficient b:
-4
Enter coefficient c:
4
The discriminant (D) is: 0.0
The equation has twp equal real roots: 2.0
```

Output3:

```
Enter coefficient a:
3
Enter coefficient b:
5
Enter coefficient c:
6
The discriminant (D) is: -47.0
The equation has complex roots:
Root 1: -0.8333333333333334 + 1.1426091000668406i
Root 2: -0.8333333333333334 - 1.1426091000668406i
```

Program 3:

Write a JAVA program to search for an element in a given list of elements using binary search mechanism

```
import java.util.Scanner;

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

      // Input the size of the array
      System.out.println("Enter the number of elements: ");
      int n = sc.nextInt();

      // Input the elements of the array
      int[] array = new int[n];
      System.out.println("Enter the elements (sorted): ");
      for (int i = 0; i < n; i++) {
         array[i] = sc.nextInt();
      }

      // Input the element to be searched
      System.out.println("Enter the element to search: ");
      int key = sc.nextInt();

      // Perform binary search
      int result = binarySearch(array, key);

      // Display the result
      if (result == -1) {
         System.out.println("Element not found in the array.");
      } else {
         System.out.println("Element found at index: " + result);
      }

      sc.close();
   }

   // Method to perform binary search
   public static int binarySearch(int[] array, int key) {
      int left = 0;
      int right = array.length - 1;

      while (left <= right) {
         int mid = left + (right - left) / 2;
```

```
        // Check if key is present at mid
        if (array[mid] == key) {
            return mid;
        }

        // If key is greater, ignore the left half
        if (array[mid] < key) {
            left = mid + 1;
        }
        // If key is smaller, ignore the right half
        else {
            right = mid - 1;
        }
    }

    // Key not found
    return -1;
  }
}
```

Output1:

```
Enter the number of elements:
6
Enter the elements (sorted):
20 30 40 50 60 70
Enter the element to search:
50
Element found at index: 3
```

Output2:

```
Enter the number of elements:
5
Enter the elements (sorted):
-1 0 2 4 8
Enter the element to search:
10
Element not found in the array.
```

Program4:

Write a JAVA program to sort 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) {
      Scanner sc = new Scanner(System.in);

      // Input the size of the array
      System.out.println("Enter the number of elements: ");
      int n = sc.nextInt();

      // Input the elements of the array
      int[] array = new int[n];
      System.out.println("Enter the elements: ");
      for (int i = 0; i < n; i++) {
         array[i] = sc.nextInt();
      }

      // Perform bubble sort
      bubbleSort(array);

      // Display the sorted array
      System.out.println("Sorted array: ");
      for (int i : array) {
         System.out.print(i + " ");
      }
   }

   // Method to perform bubble sort
   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:
Enter the number of elements:
5
Enter the elements:
1 -5 4 2 89
Sorted array:
-5 1 2 4 89
```

Program 5:

Write a JAVA program using String Buffer to delete, remove character

```
//program to illustrate StringBuffer methods
public class Str5 {
  public static void main(String args[]) {
    //creating StringBuffer object using default constructor
    StringBuffer sb = new StringBuffer("This is Text");

    //insert "a Sample" string after "is"
    //index starts with 0 so the index of s is 6
    sb.insert(7, " a Sample");
    System.out.println("after Inserting:"+sb);

    sb.append(" Book");
    System.out.println("after appending:"+sb);



    //replace "Book" with "Message"
    int index=sb.indexOf("Book");
    sb.replace(index,sb.length(),"Message");
    System.out.println("after replacing:"+sb);

    //deleting the substring
    sb.delete(index,sb.length());
    System.out.println("after deleting:"+sb);

    //deleting the character
    sb.deleteCharAt(0);
    System.out.println("after deleting a character:"+sb);
    //reversing the string
    sb.reverse();
    System.out.println("after reversing:"+sb);
```

```
  }
}
```

Output:

```
after Inserting:This is a Sample Text
after appending:This is a Sample Text Book
after replacing:This is a Sample Text Message
after deleting:This is a Sample Text
after deleting a character:his is a Sample Text
after reversing: txeT elpmaS a si sih
```

Program-6:

Write a JAVA program to implement class mechanism. Create a class, methods and invoke them inside main method.

```
class Motorcycle
{
    String make;
    String color;
    boolean engineState;
    void startEngine()
    {
       if (engineState == true)
          System.out.println("The engine is already on.");
       else
       {
          engineState = true;
          System.out.println("The engine is now on.");
       }
    }
    void showAtts()
    {
      System.out.println("This motorcycle is a " + color + " " + make);
      if (engineState == true)
         System.out.println("The engine is on.");
      else
         System.out.println("The engine is off.");
   }
}
public class Ex1
{

   public static void main (String args[])
```

```
   {
      Motorcycle m = new Motorcycle();
      m.make = "Yamaha RZ350";
      m.color = "yellow";
      System.out.println("Calling showAtts...");
      m.showAtts();
      System.out.println("---------");
      System.out.println("Starting engine...");
      m.startEngine();
      System.out.println("---------");
      System.out.println("Calling showAtts...");
      m.showAtts();
      System.out.println("---------");
      System.out.println("Starting engine...");
      m.startEngine();
    }
}
```

Output:

```
Calling showAtts...
This motorcycle is a yellow Yamaha RZ350
The engine is off.
---------
Starting engine...
The engine is now on.
---------
Calling showAtts...
This motorcycle is a yellow Yamaha RZ350
The engine is on.
---------
Starting engine...
The engine is already on.
```

Program7:

Write a JAVA program implement method overloading

```
//program to illustrate static polymorphism-method overloading
class A {
  void add(int i, int j) {
    System.out.println(i + j);
  }

  void add(float f1, float f2) {
    System.out.println(f1 + f2);
```

```
  }

  void add(String str1, String str2) {
    System.out.println(str1 + str2);
  }
}

public class Test {
  public static void main(String[] args) {
    A a = new A();
    a.add(10, 20);
    a.add(22.22f, 33.33f);
    a.add("abc", "def");
  }
}
```

Output:

```
30
55.550003
abcdef
```

Program8:

Write a JAVA program to implement constructor.

```
//program to illustrate parameterized constructor

public class Employee
{

   int empId;
   String empName;

   //parameterized constructor with two parameters
   Employee(int id, String name)
   {
      empId=id;
      empName = name;
   }
   void info()
   {
       System.out.println("Id: "+empId+" Name: "+empName);
   }

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

```
        Employee obj1 = new Employee(10245,"pavan");
        Employee obj2 = new Employee(92232,"kumar");
        obj1.info();
        obj2.info();
  }
}
```

Output:

```
Id: 10245 Name: pavan
Id: 92232 Name: kumar
```

Program9:

Write a JAVA program to implement constructor overloading.

```
//program to illustrate constructor overloading
public class Demo2 {

  String language;

  // constructor with no parameter
  Demo2() {
    this.language = "Java";
  }

  // constructor with a single parameter
  Demo2(String language) {
    this.language = language;
  }

  public void getName() {
    System.out.println("Programming Langauage: " + this.language);
  }

  public static void main(String[] args) {

    // call constructor with no parameter
    Demo2 obj1 = new Demo2();

    // call constructor with a single parameter
    Demo2 obj2 = new Demo2("Python");

    obj1.getName();
    obj2.getName();
  }
}
```

Output:

```
Programming Langauage: Java
Programming Langauage: Python
```

Program10:

Write a JAVA program to implement Single Inheritance

```
//program to illustrate single Inheritance

class A
{
  public void methodA()
  {
    System.out.println("Base class method");
  }
}

class B extends A
{
  public void methodB()
  {
    System.out.println("Child class method");
  }
}

public class SingleInheritance
{
  public static void main(String args[])
  {
    B obj = new B();
    obj.methodA(); //calling super class method
    obj.methodB(); //calling local method
  }
}
```

Output:

```
Base class method
Child class method
```

Program11:

Write a JAVA program to implement multi level Inheritance

```
//program to illustrate Multilevel Inheritance
class X
{
  public void methodX()
  {
    System.out.println("Class X method");
  }
}
class Y extends X
{
   public void methodY()
  {
     System.out.println("class Y method");
  }
}
class Z extends Y
{
  public void methodZ()
  {
    System.out.println("class Z method");
  }
}

public class MultilevelInheritance
{
  public static void main(String args[])
  {
    Z obj = new Z();
    obj.methodX(); //calling grand parent class method
    obj.methodY(); //calling parent class method
    obj.methodZ(); //calling local method
 }
}
```

Output:

```
Class X method
class Y method
class Z method
```

Program12:

Write a JAVA program for abstract class to find areas of different shapes

```
// Abstract class Shape
abstract class Shape {
   // Abstract method to calculate area
   abstract double calculateArea();

   // Method to display the area
   void displayArea() {
      System.out.println("The area is: " + calculateArea());
   }
}

// Circle class that extends Shape
class Circle extends Shape {
   private double radius;

   // Constructor
   Circle(double radius) {
      this.radius = radius;
   }

   // Implement calculateArea method

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

// Rectangle class that extends Shape
class Rectangle extends Shape {
   private double length;
   private double width;

   // Constructor
   Rectangle(double length, double width) {
      this.length = length;
      this.width = width;
   }
```

```
    // Implement calculateArea method

    double calculateArea() {
        return length * width;
    }
}

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

    // Constructor
    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    // Implement calculateArea method

    double calculateArea() {
        return 0.5 * base * height;
    }
}

// Main class to test the Shape classes
public class Main {
    public static void main(String[] args) {
        // Create objects of different shapes
        Shape circle = new Circle(5.0);
        Shape rectangle = new Rectangle(4.0, 6.0);
        Shape triangle = new Triangle(4.0, 7.0);

        // Display areas of the shapes
        System.out.println("Circle:");
        circle.displayArea();

        System.out.println("Rectangle:");
        rectangle.displayArea();

        System.out.println("Triangle:");
        triangle.displayArea();
    }
}
```

Output:

```
Circle:
The area is: 78.53981633974483
```

```
Rectangle:
The area is: 24.0
Triangle:
The area is: 14.0
```

Program13:

Write a JAVA program to give example for “super” keyword.

```
// Parent class
class Animal {
   String name;

   // Constructor
   Animal(String name) {
      this.name = name;
   }

   // Method to display name
   void display() {
      System.out.println("Animal name: " + name);
   }

   // Method to make sound
   void makeSound() {
      System.out.println("Animal makes a sound");
   }
}

// Child class
class Dog extends Animal {
   String breed;

   // Constructor
   Dog(String name, String breed) {
      super(name); // Call to superclass constructor
      this.breed = breed;
   }

   // Method to display breed
   void display() {
      super.display(); // Call to superclass method
      System.out.println("Dog breed: " + breed);
   }

   // Overriding makeSound method
   void makeSound() {
```

```
        super.makeSound(); // Call to superclass method
        System.out.println("Dog barks");
    }
}

// Main class to test the Dog class
public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog("Buddy", "Golden Retriever");

        // Display the dog's name and breed
        dog.display();

        // Make the dog sound
        dog.makeSound();
    }
}
```

Output:

```
Animal name: Buddy
Dog breed: Golden Retriever
Animal makes a sound
Dog barks
```

Program14:

Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?

```
// Flyable interface
interface Flyable {
    void fly();
}

// Swimmable interface
interface Swimmable {
    void swim();
}

// Walkable interface
interface Walkable {
    void walk();
}

// Class Duck implementing all three interfaces
class Duck implements Flyable, Swimmable, Walkable {
```

```
    public void fly() {
        System.out.println("Duck is flying...");
    }


    public void swim() {
        System.out.println("Duck is swimming...");
    }


    public void walk() {
        System.out.println("Duck is walking...");
    }
}

// Main class to test the implementation
public class Main {
    public static void main(String[] args) {
        Duck duck = new Duck();
        duck.fly();   // Calls fly method from Flyable interface
        duck.swim();  // Calls swim method from Swimmable interface
        duck.walk();  // Calls walk method from Walkable interface
    }
}
```

Output:

```
Duck is flying...
Duck is swimming...
Duck is walking...
```

Program15:

Write a JAVA program that implements Runtime polymorphism

```
// program to illustrate run time polymorphism- method overriding
class Language {
  public void displayInfo() {
    System.out.println("Common English Language");
  }
}

class Java extends Language {

  public void displayInfo() {
    System.out.println("Java Programming Language");
    super.displayInfo();
```

```
  }
}

public class Test4 {
  public static void main(String[] args) {

    // create an object of Java class
    Java j1 = new Java();
    j1.displayInfo();

    // create an object of Language class
    Language l1 = new Language();
    l1.displayInfo();
  }
}
```

Output:

```
Java Programming Language
Common English Language
Common English Language
```