1. Write a program to check number is prime or not

```java
import java.util.Scanner;

class Prime{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number : ");
        int n = scanner.nextInt();
        scanner.close();

        int p = 0;
        for (int i = 2; i <= n; i++) {
            if (n % i == 0) {
                p++;
            }
        }

        if (p == 1) {
            System.out.println("\nPrime number");
        } else {
            System.out.println("Not prime number");
        }
    }
}
```

2 .Write a program to print fibonicci series

```java
import java.util.Scanner;

public class FibonacciSequence {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number: ");
        int n = scanner.nextInt();

        int a = 0, b = 1;
        System.out.print(a + " " + b + " ");

        for (int i = 1; i <= n; i++) {
```

```
            int c = a + b;
            a = b;
            b = c;
            System.out.print(" " + c);
        }
    }
}
```

3.write a program to print rational number

```
import java.util.Scanner;

public class RationalNumberPrinter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the numerator: ");
        int numerator = scanner.nextInt();
        System.out.print("Enter the denominator: ");
        int denominator = scanner.nextInt();
        scanner.close();
        System.out.println("The rational number is: " + numerator + "/" +
denominator);
    }
}
```

4.write a program to convert smaller data type to larger data type

```
public class DataTypeConversion {
    public static void main(String[] args) {
        byte smallerDataType = 10;
        int largerDataType = smallerDataType; // Implicit conversion from byte
to int
        System.out.println("Value after conversion: " + largerDataType);
    }
}
```

5.write a program to demonstrate narrowing type casting

```java
public class NarrowingTypeCasting {
    public static void main(String[] args) {
        double a = 10.5;
        int b = (int) a; // Narrowing conversion: double to int with explicit
casting

        System.out.println("Original double value: " + a);
        System.out.println("Narrowed int value: " + b);
    }
}
```

6.write a program swap two numbers without third variable.

```java
import java.util.Scanner;

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

        // Taking input from the user
        System.out.print("Enter the value of a: ");
        int a = scanner.nextInt();

        System.out.print("Enter the value of b: ");
        int b = scanner.nextInt();

        // Printing the values before swapping
        System.out.println("\nBefore swapping:");
        System.out.println("a = " + a);
        System.out.println("b = " + b);

        // Swapping without using a third variable
        a = a + b;
        b = a - b;
        a = a - b;

        // Printing the values after swapping
        System.out.println("\nAfter swapping:");
```

```
        System.out.println("a = " + a);
        System.out.println("b = " + b);

        scanner.close();
    }
}
```

7.write a program to take the input from command line argument.

```
public class CommandLineInput {
    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Usage: java CommandLineInput <value1>
<value2>");
            return;
        }

        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);

        System.out.println("Value of a: " + a);
        System.out.println("Value of b: " + b);
    }
}
```

8.write a program to access class data members.

```
public class test {
    int a; // Class data member 'a'
    int b; // Class data member 'b'

    public static void main(String[] args) {
        test obj = new test(); // Creating an object of the class

        // Accessing and setting values of class data members
        obj.a = 10;
        obj.b = 20;
```

```
        // Accessing and printing values of class data members
        System.out.println("Value of a: " + obj.a);
        System.out.println("Value of b: " + obj.b);
    }
}
```

9.write a program to demonstrate single level inheritance

```
class Parent {
    int a = 10; // Variable in the parent class

    void displayA() {
        System.out.println("Value of a: " + a);
    }
}

class Child extends Parent {
    int b = 20; // Variable in the child class

    void displayB() {
        System.out.println("Value of b: " + b);
    }
}

class SingleLevelInheritance {
    public static void main(String[] args) {
        Child obj = new Child(); // Creating an object of the child class

        // Accessing variables from parent and child classes using methods
        obj.displayA();
        obj.displayB();
    }
}
```

10.write a program to demonstrate multilevel inheritance.

```java
class Grandparent {
    int a = 10; // Variable in the grandparent class

    void displayA() {
        System.out.println("Value of a: " + a);
    }
}

class Parent extends Grandparent {
    int b = 20; // Variable in the parent class

    void displayB() {
        System.out.println("Value of b: " + b);
    }
}

class Child extends Parent {
    int c = 30; // Variable in the child class

    void displayC() {
        System.out.println("Value of c: " + c);
    }
}

class MultiLevelInheritance {
    public static void main(String[] args) {
        Child obj = new Child(); // Creating an object of the child class

        // Accessing variables from all three classes using methods
        obj.displayA();
        obj.displayB();
        obj.displayC();
    }
}
```

11.write a program to demonstrate hierarchical inheritance.

```
class Parent {
    int a = 10; // Variable in the parent class

    void displayParentVariable() {
        System.out.println("Value of a: " + a);
    }
}

class Child1 extends Parent {
    int b = 20; // Variable in the first child class

    void displayChild1Variable() {
        System.out.println("Value of b: " + b);
    }
}

class Child2 extends Parent {
    int c = 30; // Variable in the second child class

    void displayChild2Variable() {
        System.out.println("Value of c: " + c);
    }
}

class HierarchicalInheritance {
    public static void main(String[] args) {
        Child1 obj1 = new Child1(); // Creating an object of the first child
class
        Child2 obj2 = new Child2(); // Creating an object of the second child
class

        // Accessing variables from parent and child classes using methods
        obj1.displayParentVariable();
        obj1.displayChild1Variable();

        System.out.println();

        obj2.displayParentVariable();
        obj2.displayChild2Variable();
    }
}
```

12.write a program to demonstrate method overloading.

```java
class SimpleMethodOverloading {

    // Method to add two integers
    static int add(int a, int b) {
        return a + b;
    }

    // Method to add two doubles
    static double add(double a, double b) {
        return a + b;
    }

    public static void main(String[] args) {
        // Adding two integers
        int sum1 = add(5, 7);
        System.out.println("Sum of 5 and 7: " + sum1);

        // Adding two doubles
        double sum2 = add(5.5, 7.3);
        System.out.println("Sum of 5.5 and 7.3: " + sum2);
    }
}
```

13.write a program to demonstrate method overriding.

```java
class Parent {
    void displayMessage() {
        System.out.println("This is the parent class.");
    }
}

class Child extends Parent {
    @Override
    void displayMessage() {
        System.out.println("This is the child class.");
    }
}
```

```java
public class MethodOverridingDemo {
    public static void main(String[] args) {
        Parent parentObj = new Parent();
        Child childObj = new Child();

        parentObj.displayMessage();
        childObj.displayMessage();
    }
}
```

14.write a program to achieve run time polymorphism.

```java
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 RuntimePolymorphismDemo {
    public static void main(String[] args) {
        Animal animal1 = new Dog();
        Animal animal2 = new Cat();

        animal1.sound(); // Output: Dog barks
        animal2.sound(); // Output: Cat meows
    }
}
```

15.write a program to display the execution of constructors sequences in multilevel inheritance.

```
class Grandparent {
    Grandparent() {
        System.out.println("Constructor of Grandparent class");
    }
}

class Parent extends Grandparent {
    Parent() {
        System.out.println("Constructor of Parent class");
    }
}

class Child extends Parent {
    Child() {
        System.out.println("Constructor of Child class");
    }
}

class ConstructorSequenceDemo {
    public static void main(String[] args) {
        Child childObj = new Child();
    }
}
```

16.write a program to demonstrate any three uses of this keyword.

```
public class Test {
    int x; // Instance variable

    // Constructor to initialize instance variable
    Test(int x) {
        this.x = x; // Use of this to differentiate between instance variable
and parameter
    }

    // Method to display instance variable
    void display() {
```

```
        System.out.println("Value of x: " + this.x); // Use of this to access
instance variable
    }

    // Method to invoke current class method
    void callMethod() {
        this.display(); // Use of this to call method within the same class
    }

    public static void main(String[] args) {
        Test obj1 = new Test(10);

        obj1.display(); // Use of this to call method within the same object

        obj1.callMethod(); // Use of this to invoke method within the same
object
    }
}
```

17.write a program to demonstrate any three uses of super keyword.

```
class Parent {
    int x;

    // Constructor with parameter
    Parent(int x) {
        this.x = x;
    }

    // Method to display value of x
    void display() {
        System.out.println("Value of x in Parent: " + x);
    }
}

class Child extends Parent {
    int y;

    // Constructor with parameters
    Child(int x, int y) {
        super(x); // Use of super to call superclass constructor
        this.y = y;
    }
```

```
    // Method to display values of x and y
    void display() {
        super.display(); // Use of super to call superclass method
        System.out.println("Value of y in Child: " + y);
    }

    // Method to access superclass method
    void callSuperMethod() {
        super.display(); // Use of super to call superclass method
    }
}

class Test {
    public static void main(String[] args) {
        Child child = new Child(10, 20);
        child.display(); // Use of super to access superclass method
        child.callSuperMethod(); // Use of super to call superclass method
    }
}
```

18.write a program to achieve interface concept

```
interface Animal {
    void sound();
    void eat();
}

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

    @Override
    public void eat() {
        System.out.println("Dog eats bones");
    }
}

class Cat implements Animal {
    @Override
    public void sound() {
```

```
        System.out.println("Cat meows");
    }

    @Override
    public void eat() {
        System.out.println("Cat eats fish");
    }
}

class InterfaceDemo {
    public static void main(String[] args) {
        Animal dog = new Dog();
        Animal cat = new Cat();

        dog.sound();
        dog.eat();

        cat.sound();
        cat.eat();
    }
}
```

19.write a program to display matrix of array.

```
class MatrixDisplay {
    public static void main(String[] args) {
        // Define a 2D array (matrix) with 3 rows and 3 columns
        int[][] matrix = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };

        // Display the matrix
        System.out.println("Matrix:");
        // Assuming a 3x3 matrix, iterate over rows and columns
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println(); // Move to the next row after printing
elements of current row
        }
    }
}
```

20.write a program to display ascending order of an array.

```java
public class AscendingOrder {
    public static void main(String[] args) {
        // Define an array
        int[] array = {5, 2, 9, 1, 7};

        // Sort the array in ascending order
        Arrays.sort(array);

        // Display the array in ascending order
        System.out.println("Array in ascending order:");
        for (int num : array) {
            System.out.print(num + " ");
        }
    }
}
```

21.write a program to display rate of intrest of 4 banks using interface concept.

```java
interface Bank {
    double getInterestRate();
}

// Implement the Bank interface for different banks
class BankA implements Bank {
    @Override
    public double getInterestRate() {
        return 7.5; // Example interest rate for Bank A
    }
}

class BankB implements Bank {
    @Override
    public double getInterestRate() {
        return 8.0; // Example interest rate for Bank B
    }
}
```

```java
class BankC implements Bank {
    @Override
    public double getInterestRate() {
        return 7.25; // Example interest rate for Bank C
    }
}

class BankD implements Bank {
    @Override
    public double getInterestRate() {
        return 7.75; // Example interest rate for Bank D
    }
}

public class BankInterestRateDemo {
    public static void main(String[] args) {
        Bank bankA = new BankA();
        Bank bankB = new BankB();
        Bank bankC = new BankC();
        Bank bankD = new BankD();

        // Display interest rates for different banks
        System.out.println("Interest rates for different banks:");
        System.out.println("Bank A: " + bankA.getInterestRate() + "%");
        System.out.println("Bank B: " + bankB.getInterestRate() + "%");
        System.out.println("Bank C: " + bankC.getInterestRate() + "%");
        System.out.println("Bank D: " + bankD.getInterestRate() + "%");
    }
}
```

22.write a program to demonstrate abstract class and abstract method.

```java
abstract class Test {
    // Abstract method
    abstract void display();
}

// Concrete subclass extending abstract class
class Test1 extends Test {
    // Implementation of abstract method
    void display() {
        System.out.println("Implementation of abstract method in Test1");
```

```
    }
}

public class AbstractDemo {
    public static void main(String[] args) {
        // Cannot instantiate an abstract class directly
        // Test obj = new Test(); // Compilation error

        // Creating object of subclass
        Test1 obj1 = new Test1();

        // Calling the abstract method
        obj1.display();
    }
}
```

23.write a program to achieve multithreading using Thread class.

```
class MyThread extends Thread {
    // Override the run method to define the task for the thread
    public void run() {
        // Task to be executed by the thread
        for (int i = 1; i <= 5; i++) {
            System.out.println("Thread: " + i);
            try {
                // Introduce some delay to simulate a time-consuming task
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

public class MultithreadingDemo {
    public static void main(String[] args) {
        // Create objects of MyThread class
        MyThread thread1 = new MyThread();
        MyThread thread2 = new MyThread();

        // Start the threads
        thread1.start();
        thread2.start();
    }
}
```

24.write a program to hold a thread using join method.

```
class MyThread extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println("Thread: " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

public class JoinDemo {
    public static void main(String[] args) {
        MyThread thread1 = new MyThread();
        MyThread thread2 = new MyThread();

        // Start the first thread
        thread1.start();

        try {
            // Join the first thread
            thread1.join();
        } catch (InterruptedException e) {
            System.out.println(e);
        }

        // Start the second thread after the first thread completes execution
        thread2.start();
    }
}
```

25.write a program to stop the thread.

```
class threadDemo extends Thread {
    public void run() {
        String name = Thread.currentThread().getName();
        for (int i = 0; i < 3; i++)
            System.out.println(name);
    }
}

class Demo {
    public static void main(String[] args) {
        threadDemo th1 = new threadDemo();
        threadDemo th2 = new threadDemo();
        threadDemo th3 = new threadDemo();
        th1.setName("thread 1");
        th2.setName("thread 2");
        th3.setName("thread 3");
        th1.start();
        th2.start();
        th3.start();
        th2.stop();
        for (int i = 0; i < 3; i++)
            System.out.println("main");
    }
}
```

26.write a program to achive multithreading concept using Runnable interface.

```
class thread1 implements Runnable {
    public void run() {
        for (int i = 0; i < 5; i++)
            System.out.println("child thread");
    }
}

class ThreadDemo {
    public static void main(String[] args) {
        thread1 th = new thread1();
        Thread t = new Thread(th);
```

```
        t.start();
        for (int i = 0; i < 5; i++)
            System.out.println("main thread");
    }
}
```

27.write a program to import one package to another package and display it's class members.

```
package package1;

public class Class1 {
    public void display() {
        System.out.println("Class1 method called");
    }
}
```

```
package package2;

import package1.Class1;

public class Main {
    public static void main(String[] args) {
        // Create an instance of Class1
        Class1 obj = new Class1();

        // Call the display method from Class1
        obj.display();
    }
}
```

28.write a program to handle arithmetic exception.

```
class ArithmeticExceptionDemo {
    public static void main(String[] args) {
        try {
            int dividend = 10;
            int divisor = 0;
            int result = dividend / divisor; // This line may throw
ArithmeticException
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            // Handle the exception
            System.out.println("Error: Division by zero");
        }
    }
}
```

29.write a program to use multiple catch blocks.

```
class MultipleCatchBlocksDemo {
    public static void main(String[] args) {
        try {
            int[] numbers = {1, 2, 3};
            System.out.println(numbers[4]); // This line may throw
ArrayIndexOutOfBoundsException
            int result = 10 / 0; // This line may throw ArithmeticException
        } catch (ArrayIndexOutOfBoundsException e) {
            // Handle the ArrayIndexOutOfBoundsException
            System.out.println("Error: Array index out of bounds");
        } catch (ArithmeticException e) {
            // Handle the ArithmeticException
            System.out.println("Error: Division by zero");
        } catch (Exception e) {
            // Handle any other exception
            System.out.println("Error: Unknown exception occurred");
        }
    }
}
```

30.write a program to display important content(code) using finally block.

```
class FinallyBlockDemo {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero");
        } finally {
            System.out.println("Finally block executed - Important content
(code)");
        }
    }
}
```

31.write a program to achive multiple inheritance using interface concept.

```
interface Parent1 {
    void method1();
}

// Interface for second parent class
interface Parent2 {
    void method2();
}

// Child class implementing both interfaces
class Child implements Parent1, Parent2 {
    public void method1() {
        System.out.println("Method 1 implemented by Child");
    }

    public void method2() {
        System.out.println("Method 2 implemented by Child");
    }
}
```

```java
class MultipleInheritanceDemo {
    public static void main(String[] args) {
        // Creating an object of Child class
        Child child = new Child();

        // Calling methods inherited from Parent1 and Parent2
        child.method1();
        child.method2();
    }
}
```

32.write a program of file handling to display name of file, size and address.

```java
import java.io.File;

class FileHandlingDemo {
    public static void main(String[] args) {
        // Create a File object representing the file
        File file = new File("example.txt");

        // Display file name
        String fileName = file.getName();
        System.out.println("File Name: " + fileName);

        // Display file size
        long fileSize = file.length();
        System.out.println("File Size: " + fileSize + " bytes");

        // Display file address
        String fileAddress = file.getAbsolutePath();
        System.out.println("File Address: " + fileAddress);
    }
}
```