**Java**

**[app2022 and javacode folder]**

Taking input from user by object of scanner class.

**Scanner :-** Scanner is a built-in class which is available in java.util package.

**Package :** package is a container of classes, interfaces and sub- packages.

// Write a java code to print a message.

// Class is a blueprint of object.

// Class is a container of variables and methods.

class p1{

public static void main(String[] args)

{

System.out.println("Hello world");

}

}

Sol :-



6-feb-2023

**Revision JAVA**

**Java Programming**

Author : James Gosling

Vender : SUN Microsoft

Old Name : OAK

Present Name : Java

Symbol : Coffee Cup with Saucer

Slogan : WORA ( Write once run anywhere)

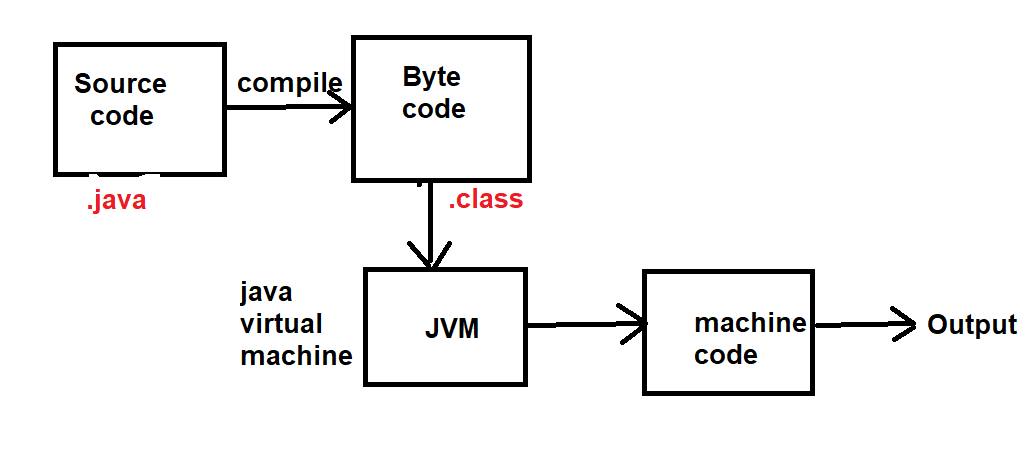
Language Type : Open Source

Operating System : Any Operating System

Developed From : C and C++

Extensions : .java , .class , .jar

**Execution Scenario Of JAVA**



**Features of Java :**

1. **Simple :-**  Java is Simple for C and C++ learner , because all complexities of C and C++ reduced in java .
2. **Open Source :-** Java is open source that means source code of java is available for user and user can modify source code .
3. **Platform Independent :**- Java is Platform Independent , that means you can run java program on any Operating System .
4. **Object Oriented :-** Java Programming Language is an Object Oriented programming language .
5. **High Performance :**- The performance of java programming is better than C and C++ .
6. **Write Once Run Anywhere .**
7. **Architecture Neutral :-**  Java program is not executed under operating system. Java program is executed under java runtime environment (JRE).
8. **Multithreaded :-** Java programming Language supports concept of multithreading .
9. **Web Application Development :** You can develop web application by using java programming language .

**Applications Developed Using Java :-**

1. **Desktop Application :**

e.g. Media Player , Anti-Virus ….. etc .

1. **Web Application :**

e.g. irctc.co.in.

1. **ERP ( Enterprise Resource Planning ) like banking solution**

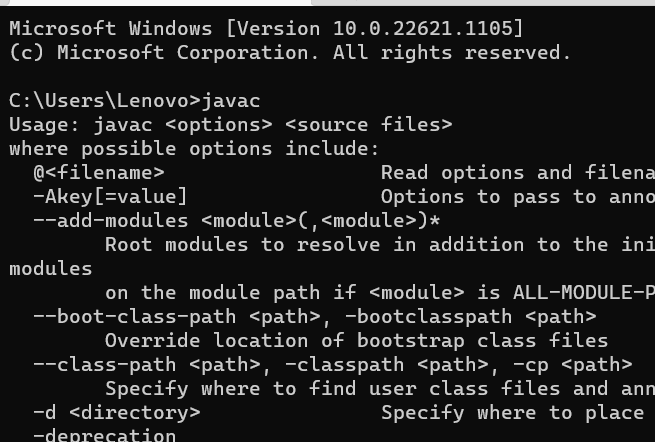
e.g. Finacle .

1. **Mobile Application Development .**
2. **Embedded System and Robotics .**
3. **Game Development ……**

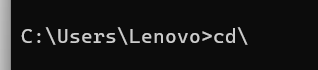
**JDK ( Java Development Kit )**

**Commands :**

**Javac :-** For checking JDK is available in our system .

****

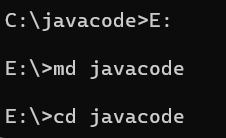
**Cd :- It is used for that Directory .**

****

**E : For change the Directory .**

**Md : Create new folder .**

**Cd :** Usi folder me rahane ke liye.



**Code file :-**



**Program ….**

**Compile the program :** filename se compile hoti hai .

**javac filename**

**javac p1.java**

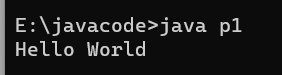


**For running the program :** program me jo class bnate tym classname dete hai

usse run hota hai .

**java className**

**Java p1**



**dir 🡪** It is used for know the, how much files in the folder .

**Class :** Class is a blueprint of object.

**Example :**

**class p1**

**{**

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

**{**

**System.out.println("Hello World");**

**}**

**}**

🡪public is a keyword , that is work as Access modifier .

Main method can make public .

🡪 Static is a keyword , that is a Modifier .

Static method ko call karne ke liye Object ki jarurt nahi parti .

🡪Main is a method , our program start from a main method .

🡪void is a keyword , that is work as Return type .

(agr kisi method me , void use kar lete hai to vo koi value return nahi karta hai.) void use kiya gaya hai to vo

🡪String[] args ---- It is use to take input from command line .

System.out.println("Hello World");

🡪System is a class

🡪out is a Object

🡪println is a Method .

**7-feb-2023**

**How to take input from USER in java ?**

For input in java we use **Scanner class** object. Scanner is a class available in java.util package.

**Package :** Package is a collection of classes , interfaces and sub-packages.

**Code :-**

**import java.util.Scanner;**

**|**

**Object**

// ClassName objectName=new ClassName();

Scanner scanner=new Scanner(System.in);

|

Input

For **int** input :-

Scanner class me nextInt() naam ka method hai . Jiska use input lene ke liye krte hai .

Int a;

a=scanner**.nextInt();**

For **float** input :-

Float b;

b=s.**nextFloat();**

For **Double** input :-

double c;

c=s.**nextDouble();**

For **String** input :-

String name;

name=s.**nextLine();**

Build-in class ka first letter capital hota hai.

**Decision Controls in java :-** Decision Controls are used for decision making .

In java programming language there are following types of decision controls :-

1. If Statement
2. If-else Statement
3. Ladder if-else Statement
4. Switch Statement

**If-Statement :-**

**If** is a keyword , which works as decision control .

We attach a condition with if statement, if given statement is true , then if block will executed otherwise no code will executed .

**If(Condition)**

**{**

**// code**

**}**

**If-else Statement :-**

If-else is the variation of if statement, We attach a condition with if statement, if given condition is true then if block code will executed and if given condition is false then else block code will executed.

**if(Condition)**

**{**

**//code**

**}**

**else**

**{**

**//code**

**}**

**Ternary Operator ( ? ) :-**

Ternary Operator is a alternate of if-else .

**(expression1) ? ( expression2) : ( expression3)**

* **If expression1 is true then expression2 will executed and if expression1 is false then expression3 will executed .**

**Question :**

1. **WAP to a find area and perimeter of rectangle .**
2. **WAP to check given year is leap year or not .**
3. **WAP to find roots of Quadratic equation .**

**08/feb/2023**

**Ladder if-else :-**

If you have many conditions and you want to execute code based on those conditions, then you can use ladder if-else .

**If(condition1)**

**{**

**// code1**

**}**

**else if(condition)**

**{**

**//code2**

**}**

**else**

**{**

**//code3**

**}**

**Switch Statement:**

Switch is a keyword, which work as case control. It is used to create a menu based program.

// In switch we cannot passed double or float value.

Switch, case, break, default they are a keyword.

**Switch(expression) //int or char or String**

**{**

**case 1:**

**//code1;**

**break;**

**case2:**

**//code2;**

**break;**

**default:**

**//code**

**}**

**Loop Control:**

If you have a block of code which you want to execute repeatedly up to given condition is true, then you can use a loop control.

In java programming language there are Four types –

1. While loop
2. For loop
3. Do-while loop
4. For each loop

**While loop:**

While is a keyword which works as loop control.

**Initialization of loop counter,**

**While(condition)**

**{**

**//code**

**Updation of loop counter**

**}**

**Question:**

1. **Input coordinates of point and check quadrants.**
2. **Temperature Convertor.**

**For loop:**

For is a keyword, which works as loop control.

Working of for loop is same as while but syntax is different.

**For (Initialization; Condition; Updation)**

**{**

**// code**

**}**

**Do-while loop:**

**Initialization of loop counter;**

**do**

**{**

**// Code;**

**Updation of code;**

**} while(Condition);**

**10/feb/2023**

**Question:**

1. **WAP to find compound interest.**
2. **WAP to find area of perimeter of circle.**
3. **WAP to check given number is Armstrong or not.**

**For each loop:**

For each loop in java is used to traverse elements of a collection like array.

**ARRAY in java:**

Array is a collection of similar data types, that means an array can store multiple values of similar data types.

**How to create Array in java?**

**datatype [] arrayname=new datatype[size];**

**For example:**

**int [] list=new int[10];**

**How to initialization of Array:**

**int [] x = {10,20,30,40,50} ;**

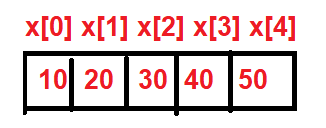
**x[0] = 10**

**x[1] = 20**

**x[2] = 30**

**x[3] = 40**

**x[4] = 50**

****

**How to take input from user for an array?**

**Code Segment:-**

int [] x=new int[5];

int i;

Scanner s=new Scanner(System.in);

System.out.println(“Enter five number”);

for(i=0; i<5; i++)

{

x[i]=s.nextInt();

}

**STRING in java:**

String is a class in java. The object of String class is used to store **sequence of characters**.

**String str=“Softpro India”;**

**toUpperCase() :- toUpperCase() method of String class is used to convert String into Upper Case (Capital letters).**

**toLowerCase() :- toLowerCase() method of String class is used to convert String into Lower Case (Small letters).**

**Length() :- length() method of String Class is used to find length of Sting.**

**equals() :- equals() method of String Class is used to compare two Strings for equality. This method return Boolean value.**

**equalsIgnoreCase() :- This method of String class compare two**

**split() :- split() method of String class split String into substrings.**

**This method return array of String .**

**Eg :**

**String str= “He is a good boy.”**

**String [] words = str.split(“ ”);**

**words[0] = “He”;**

**words[1] = “is”;**

**words[2] = “a”;**

**words[3] = “good”;**

**words[4] = “boy”;**

**replace() :-- replace() method of String class replace one string with another string in given string.**

**replace(findWhat,replaceWith);**

**Method in java :-**

Method is a named block of code, which perform specific task.

If you have a block of code which required different locations of program, then you can create a method of that code and call it from desired locations.

By using method you can avoid to write same code over and over.

Method is used to achieve modularity.

**How to create a method in java?**

**<modifier> <return\_type> method\_name(parameters)**

**{**

**//code**

**}**

**Ex:-**

Public int add(int x, int y)

{

return(x+y);

}

**Types of Methods in java:-**

In java programming language there are two types of methods –

1. Static methods
2. Non-static methods

**Static method –**

Static methods are created by using static, modifier.

These methods are also called **class methods.**

There is no need of object to call static methods.

// Static method ko with object or without object call kiya ja skta hai.

**Non-Static Method –**

Non-static methods are created without using static modifier.

There is a need Object to call non-static methods.

**Note:- Method are created within class and outside of main() method.**

**OOPS**

OOPS stands for **Object Oriented programming System**. It is a mechanism of software development.

**OOPS has four pillars –**

1. **Abstraction**
2. **Encapsulation**
3. **Inheritance**
4. **Polymorphism**

**Abstraction –**

Abstraction is a mechanism to **hide functionalities** of an object.

**Encapsulation –**

Encapsulation is a mechanism to wrap properties and functionalities in a single unit.

That single unit is called object.

**Inheritance –**

Inheritance is a mechanism to **create new product** by using **existing product.**

**Polymorphism –**

Term Polymorphismmeans **one thing many forms.**

**Note :-- Any programming language which follows these four concepts is knows as Object Oriented Language.**

**16/Feb/2023**

**CLASS:-**

Class is a blueprint of object. Class is a container of variables and methods. Class is created by using **“Class”** keyword followed by classname.

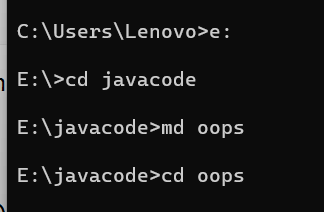
The body of class is enclosed within braces.

**Class Employee**

**{**

**// Variables and Methods**

**}**

****

**Constructor:-**

**Constructor is a special method, which is used to initialize variables.**

**The constructor has following properties –**

1. **Constructor name is same as Class Name.**
2. **Constructor has no return type.**
3. **Constructor Call automatically as soon as object is created.**

**Inheritance In Java:-**

**In Inheritance you can create a new class by using existing class. Existing class is called base class and new created class is called derived class.**

**\_\_\_\_\_\_**

**| A |**

**\_\_\_\_\_\_**

**|**

**\|/**

**\_\_\_\_\_\_**

**| b |**

**\_\_\_\_\_\_**

* **The concept of Inheritance is also called “Reusability” .**

**Class A {**

**//Variable and methods**

**}**

**Class B extends A {**

**//Variable and methods**

**}**

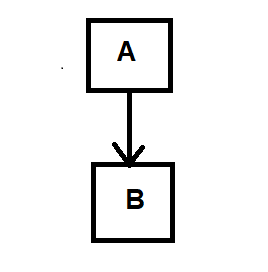
**Types of Inheritance in java:-**

**In java programming language there are three types of inheritance are available:-**

1. **Single Inheritance**
2. **Hierarchical Inheritance**
3. **Multi-level Inheritance**

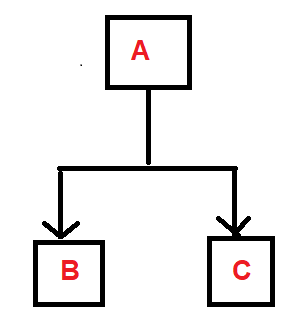
**Single Inheritance:-**

**In Single Inheritance there is a single class and single derived class.**

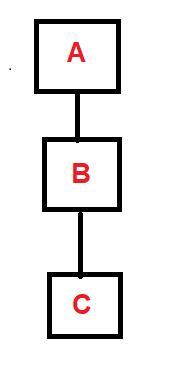
****

**Heirarchical Inheritance:-**

**In Heirarchical Inheritance there is a single base class and multiple derived classes.**

****

**Multi-level Inheritance:-**

****

**Class A**

**{**

**// code**

**}**

**Class B extends A**

**{**

**// code 2**

**}**

**Class C extends B**

**{**

**//code 3**

**}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**18/feb/2023**

**Polymorphism:-**

The term ‘Polymorphism’ means one thing many forms.

**In java there are two types of Polymorphism—**

1. *Compile time polymorphism* (**Method Overloading**)
2. *Run time polymorphism* (**Method Overriding**)

* **Method Overloading—**

In java you can create multiple methods with same name in same class,

but their parameters should be different.

Based on Method parameter it is decided at compilation time that which

method call from where, This is called Method Overloading.

**Method parameters can be different in two types—**

1. Number of parameter can be different
2. Types of parameter can be different

* **Method Overriding:-**

Re-writing of based class method into derived class is called method Overriding.

**For Ex:-**

class Connection

{

void connect()

{

// Connect java code with oracle database

}

}

class NewConnection extends Connection

{

void connect()

{

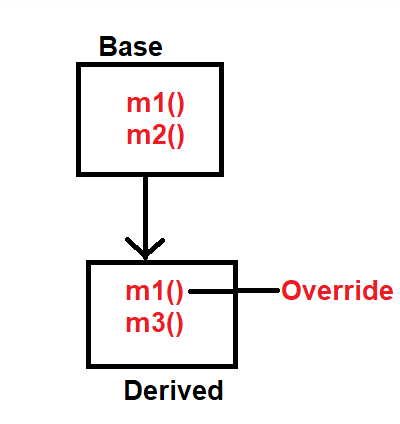
// Connect java code with oracle database

}

}

**There are following rules to perform method overriding—**

1. **Class must be inherited.**
2. **Base class** method **name** and **derived class** method name **must be same**.
3. **Base class** method **parameters** and **derived class** method parameters must be same.
4. Base class method **return type** and derived class method return type can be **same**.



**Exception Handling In Java:-**

**Exception —**

The dictionary meaning of Exception is abnormal terminated ab normally and rest of code is not executed.

**In java programming language there are Three types of Exceptions—**

1. ***Checked Exceptions***
2. ***Unchecked Exceptions***
3. ***Errors***

* **Checked Exceptions —**

Checked Exceptions are those exceptions which are identified

by compiler at **Compilation time**.

**E.g. –**

**ClassNotFoundExceptions, IOExceptions, SQLExceptions,**

**FileNotFoundException, InterruptedException…etc.**

* **Unchecked Exceptions –**

Unchecked Exceptions are those exceptions which are

identified at **Run time**.

**E.g. –**

**NullPointerExceptions, ArithmeticExceptions, InputMismatchedExceptions, ArrayIndexOutOfBoundsException…etc.**

* **Error –**

Error are occurred due to **lack of system resources**.

**E.g. –**

**AwtError, FileNotFoundError, IOError,…etc.**

**21/Feb/2023**

**Concept of Interface, Abstract Class & Class –**

**Interface –**

Interface in java is a container of abstract methods.

It is used to achieve full abstraction.

Abstract method ve methods hote hai jinme sirf methods ka declearation hota hai, use nhi hota hai.

**E.g. –**

**Interface MyInterface**

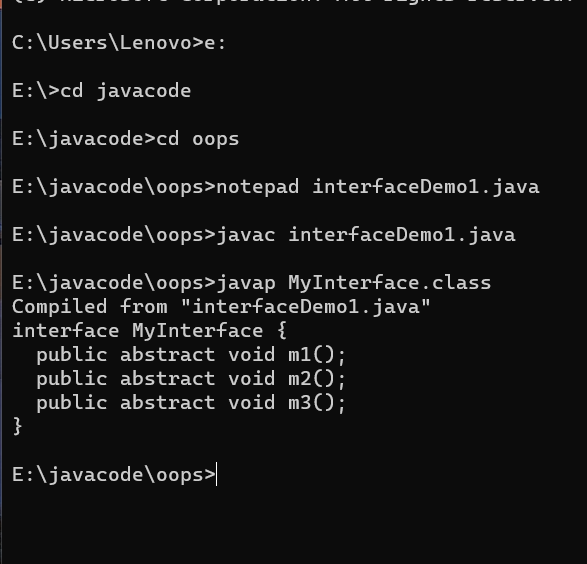
**{**

**void m1(); // public abstract void m1();**

**void m2(); // public abstract void m2();**

**void m3(); // public abstract void m3();**

**}**

****

****

**Object –**

**Class ka object bnega alwage**

**Object sirf usi ka bnega jaha sari files implemented hogi. [ like that🡪 class]**

**Interface –**

If you have requirements but you don’t know about its implementations, then you can use interface contain abstracts method only.

* **An interface can extends (Inherit) another Interface.**
* **You can’t Create Object of Interface.**

**Abstract Class –**

**Abstract class is a class which contain abstract methods and implemented method both.**

* **If you have requirements, you know about implementations but not complete implementations, then you can use abstract class.**
* **An abstract class can implement interface.**
* **An abstract class can extends another abstract class.**
* **You can’t create object of abstract class.**

**Class –**

**Class is a container of implemented methods.**

**If you have requirements and you know about complete implementation then you can use class.**

* **A class can implements an Interface.**
* **A class can extends abstract class.**
* **A class can extends another class.**
* **You can create object of class.**

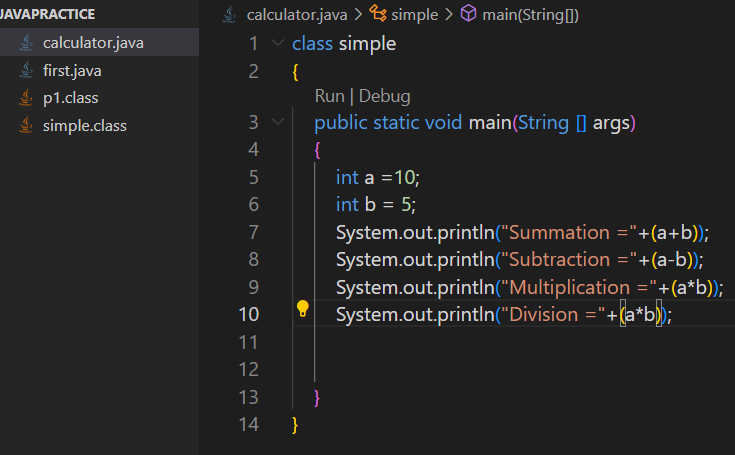
**Examples – (javacode 🡪 oops 🡪 interfaceDemo2.java )**

**JAVAPRACTICE –**

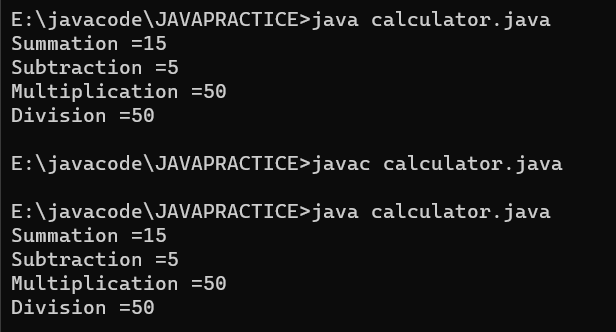
Programs –

1. **Simple Calculator –**

Without take input from user.

****

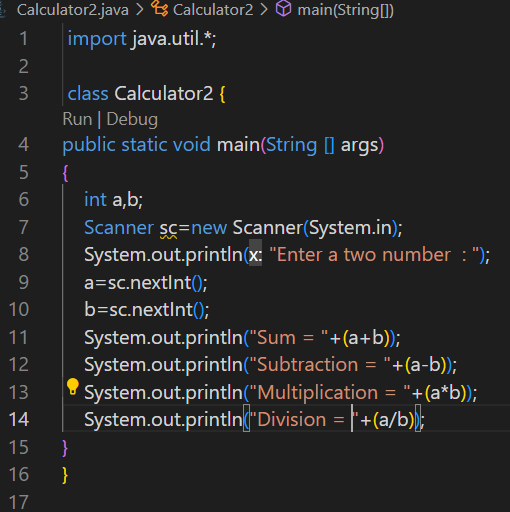
**Sol –**



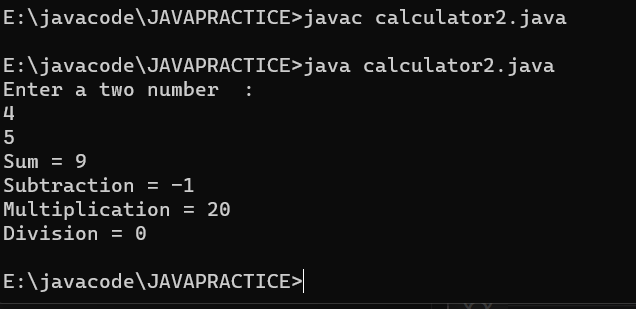
**INPUT FROM USER** –

1. **WAP a program to make a Simple Calculator.**

**SOL :-**

****

**After RUN –**

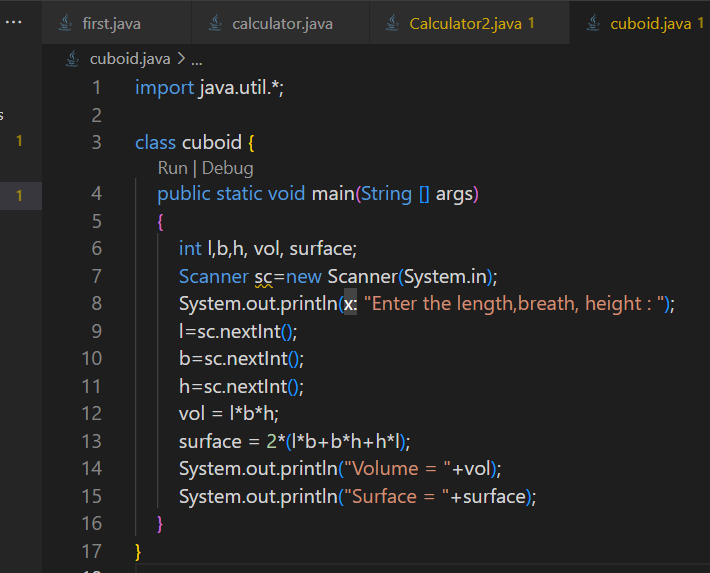
****

1. **Write a program to find the surface area of cuboid .**

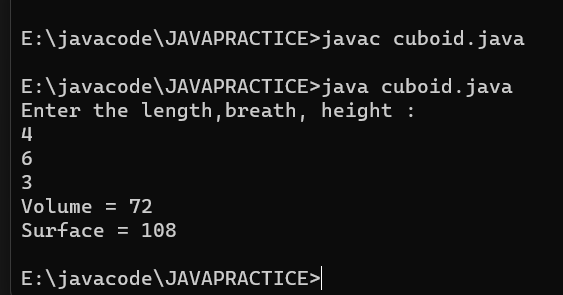
**Volume = l\*b\*h**

**Surface = 2(l\*b+b\*h+h\*l)**

**Sol –**

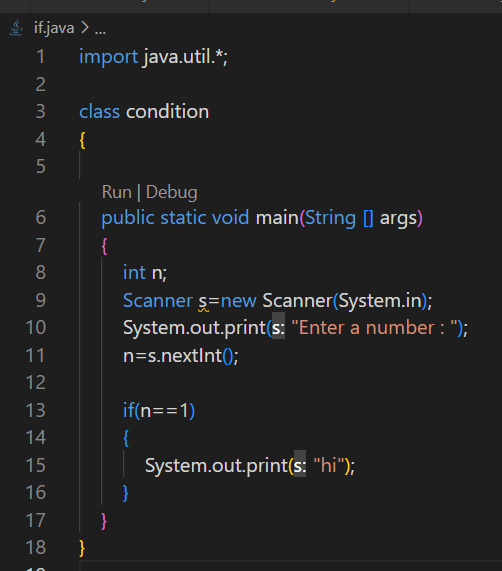
****

**Running –**

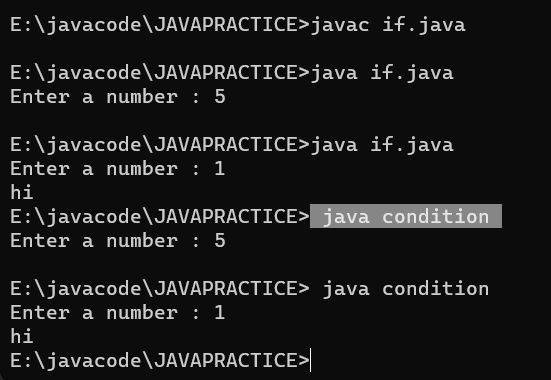
****

1. **WAP to take a number as input, if number is 1 then print “Hi” .**

**Sol –**

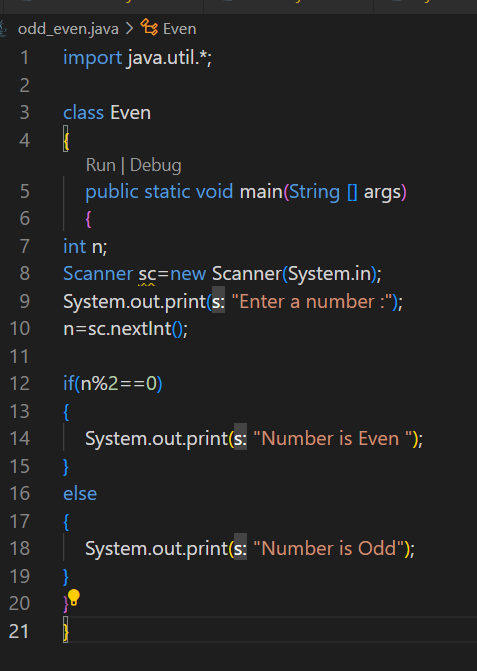
****

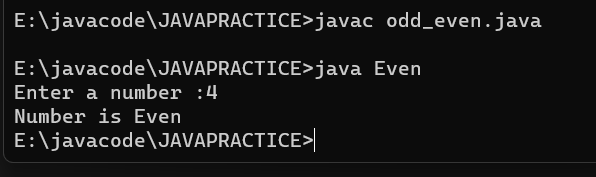
**Running –**

****

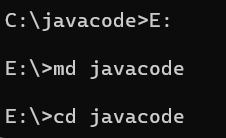
1. **WAP to check given number is Even or Odd .**

**Sol –**

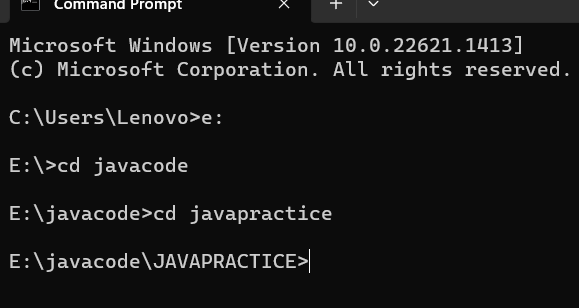
****

****

**FOR MAKE Folder –**



1. **First Program in JAVA.**

****

**Compile –**

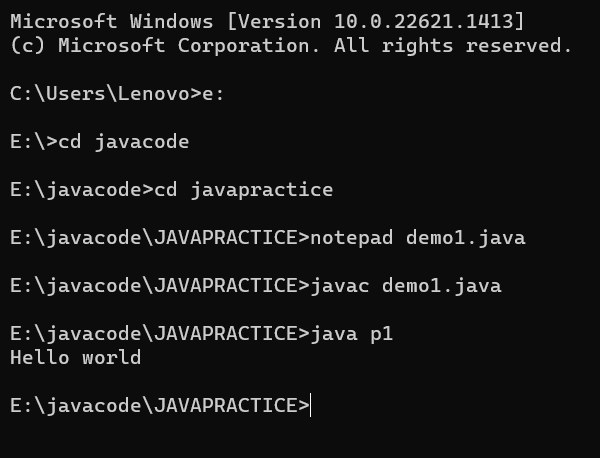
**Javac filename.java**

**Javac demo1.java**

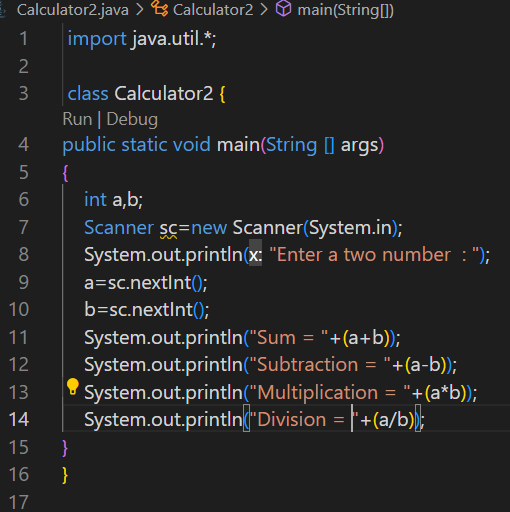
**Running –**

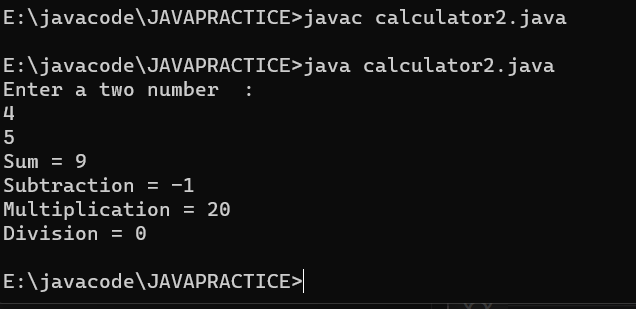
**Java CLASSNAME**

**Java p1**

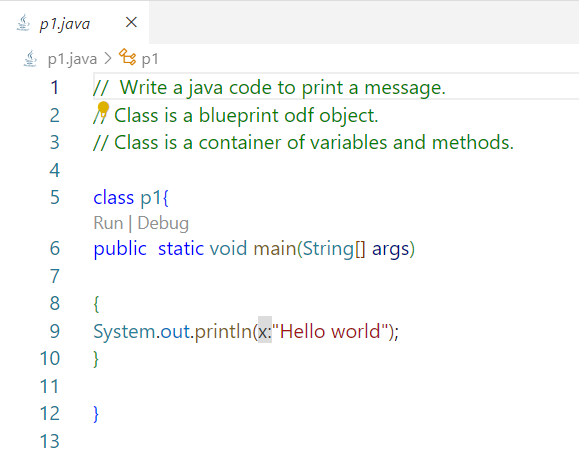
****

1. **Calculator program.**

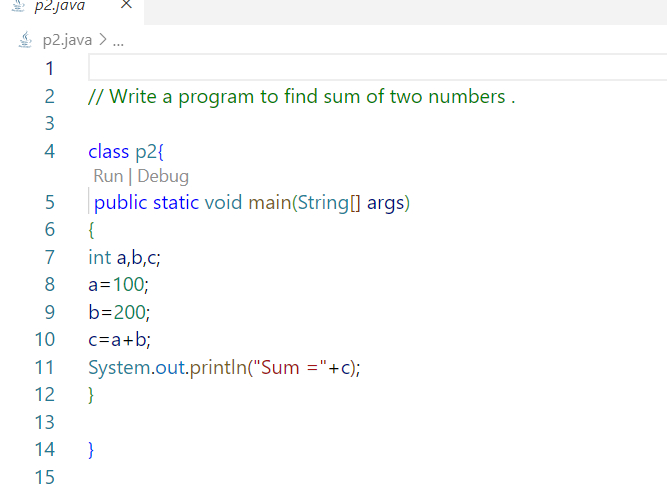




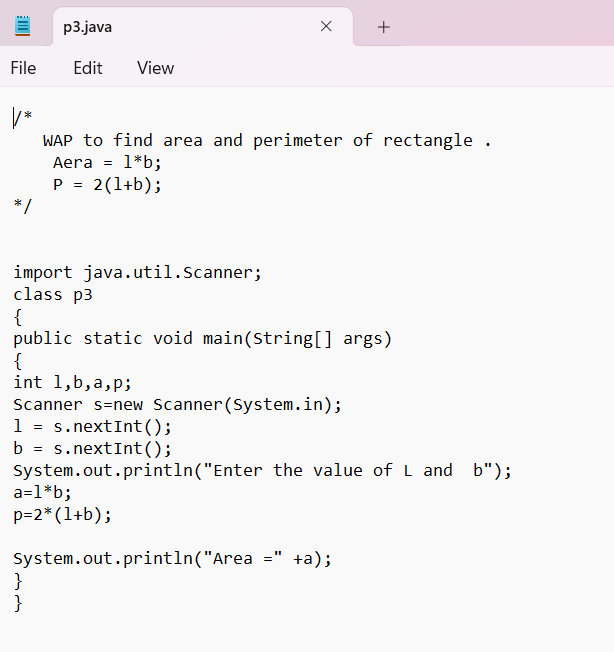
**Programs 🡪 1. )**

****

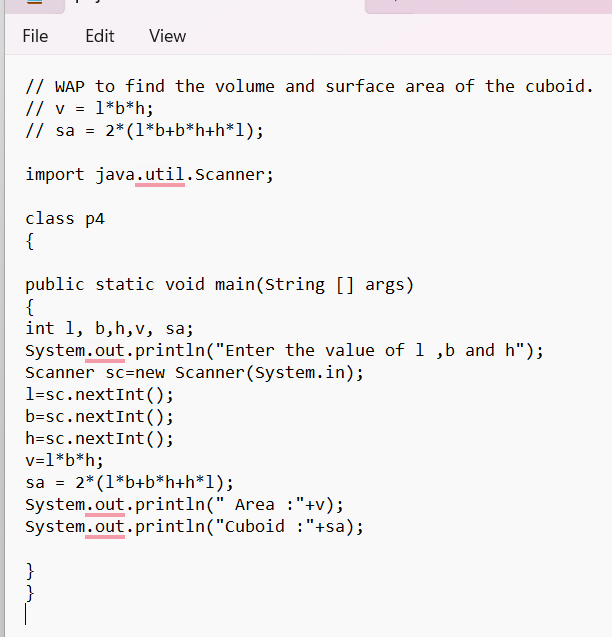
**2. )**

****

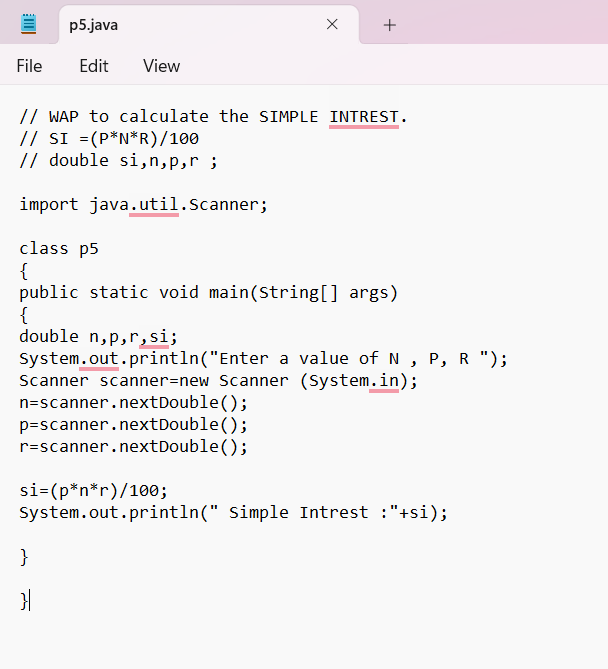
**3.)**

****

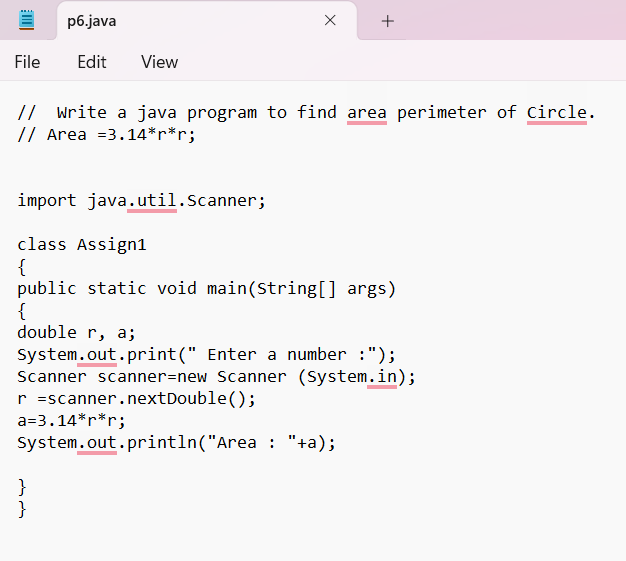
**4.)**

****

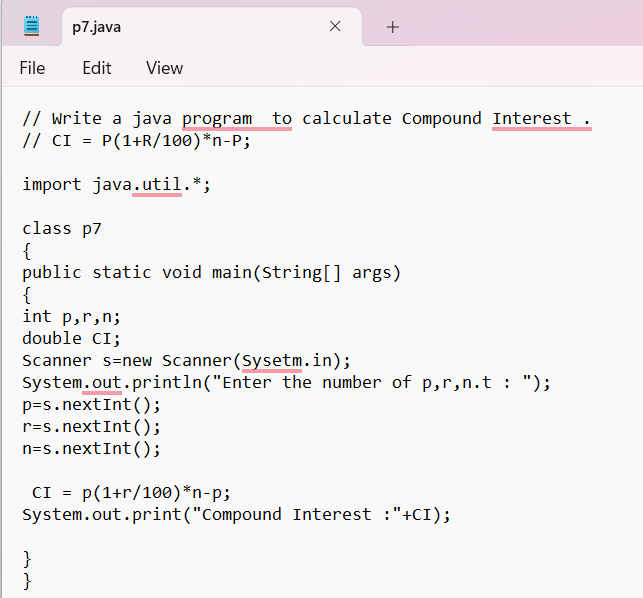
**5.)**

****

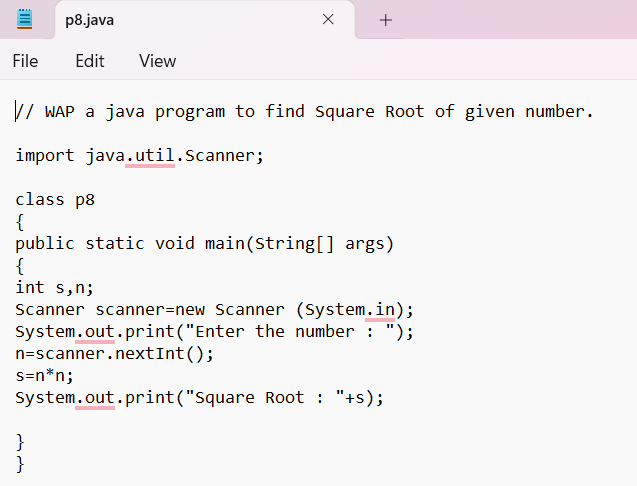
1. **)**

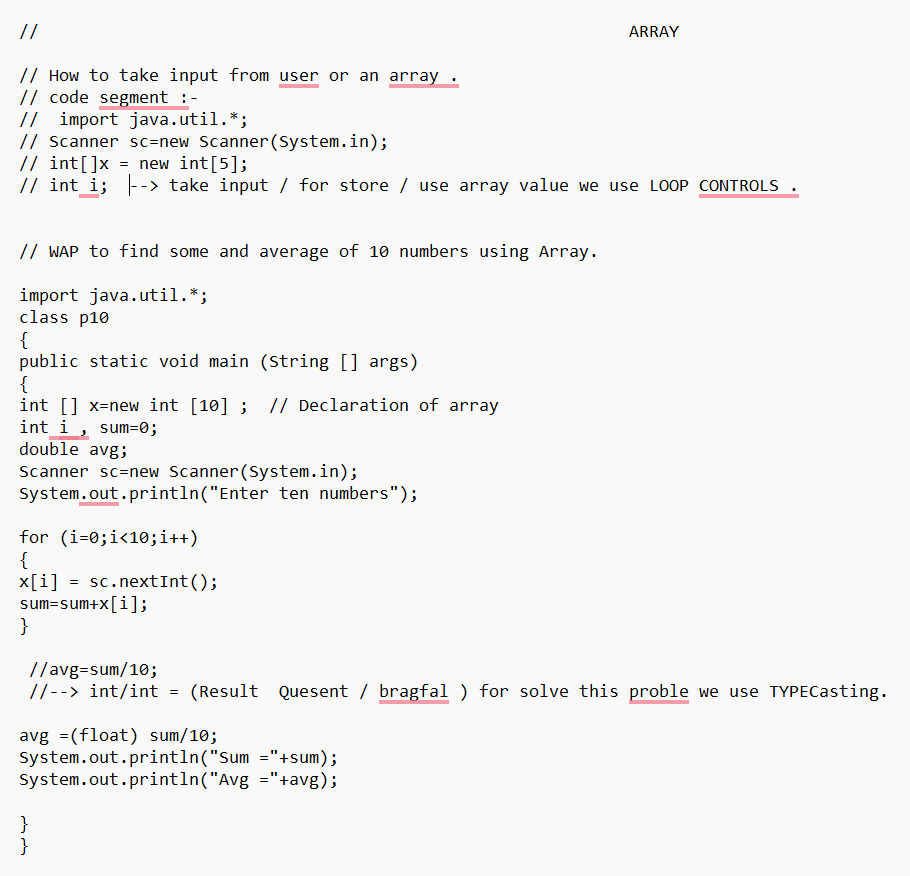
****

**7.)**

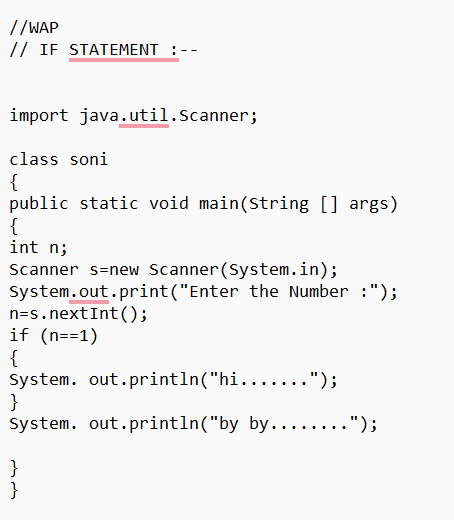
****

**8.)**

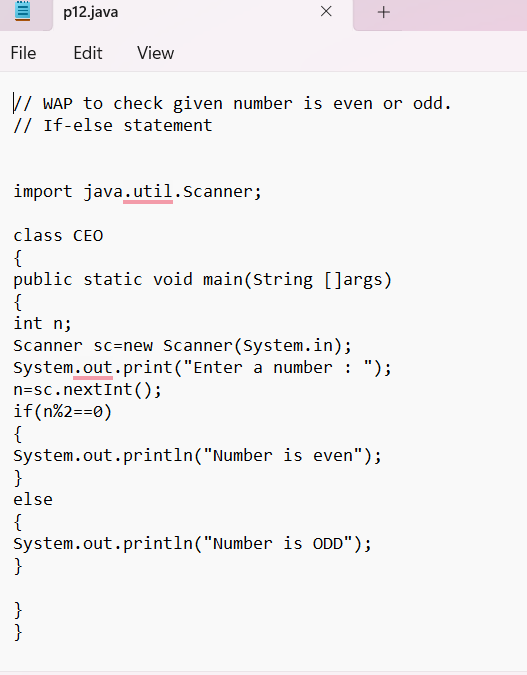
****

**9.) **

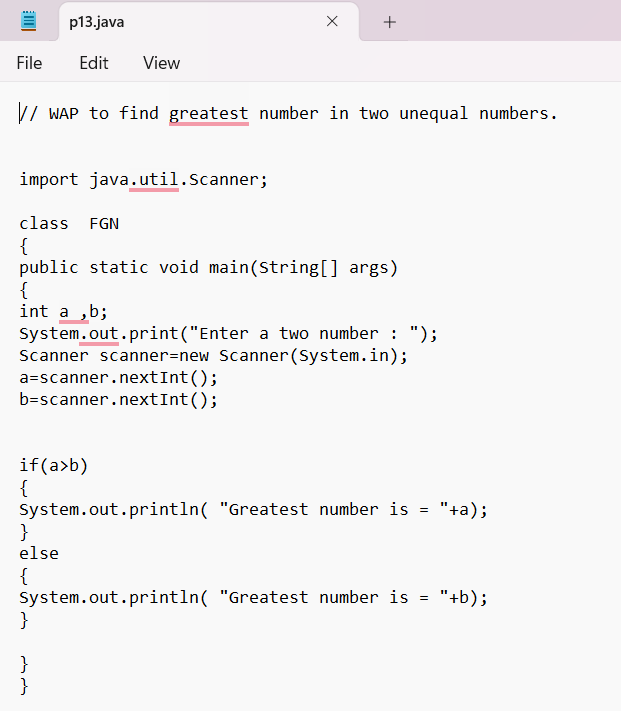
**10.)**

****

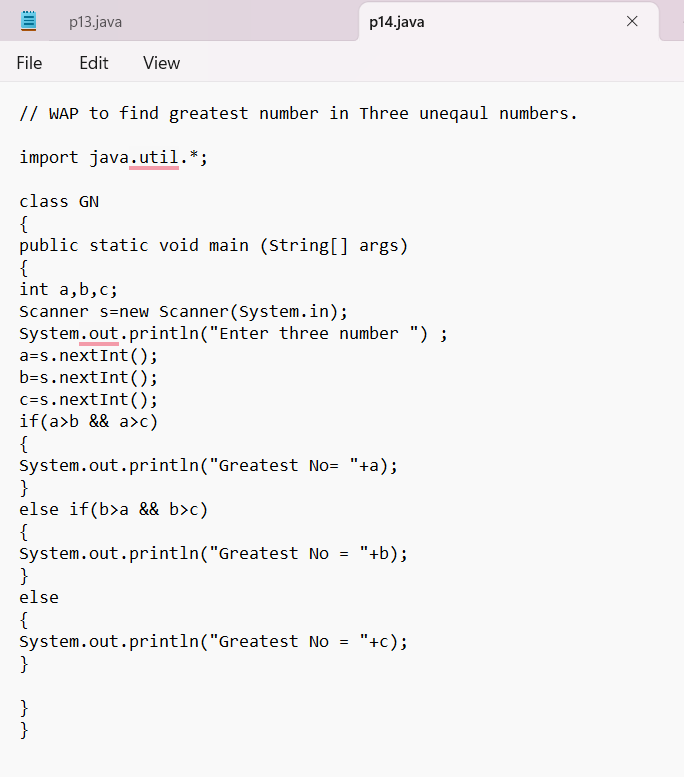
**11.)**

****

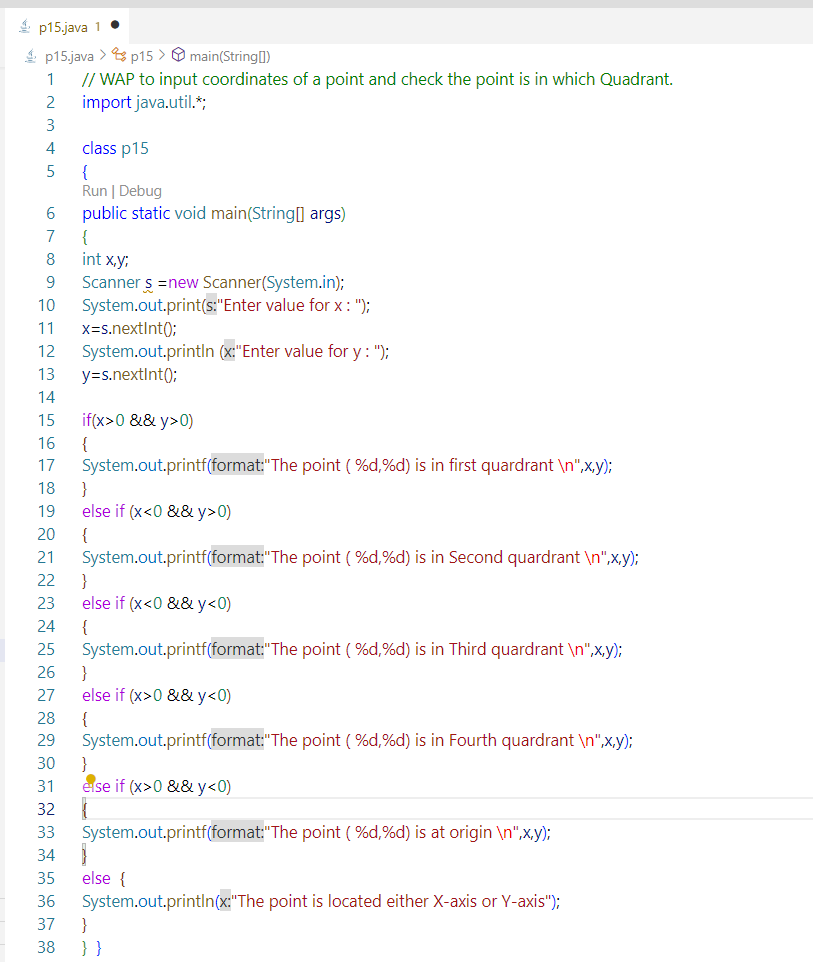
**12.)**

****

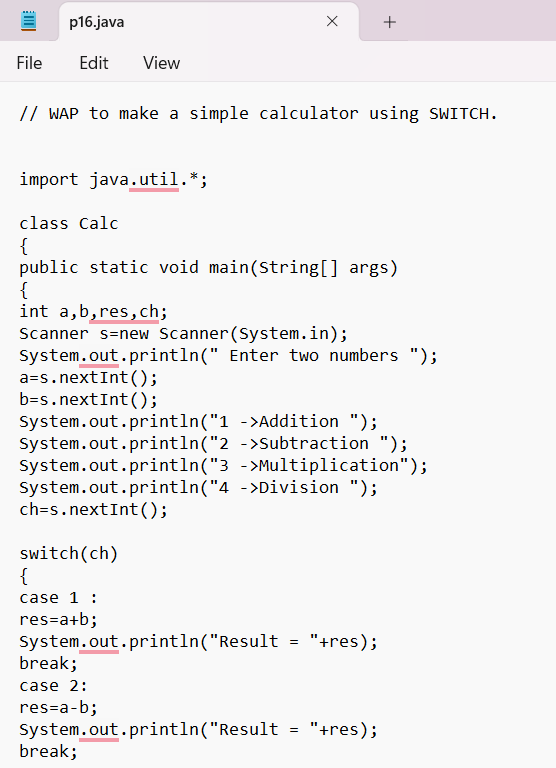
**13.)**

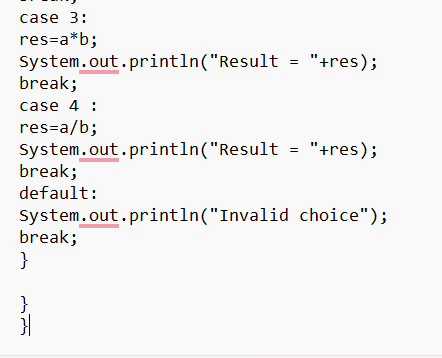
****

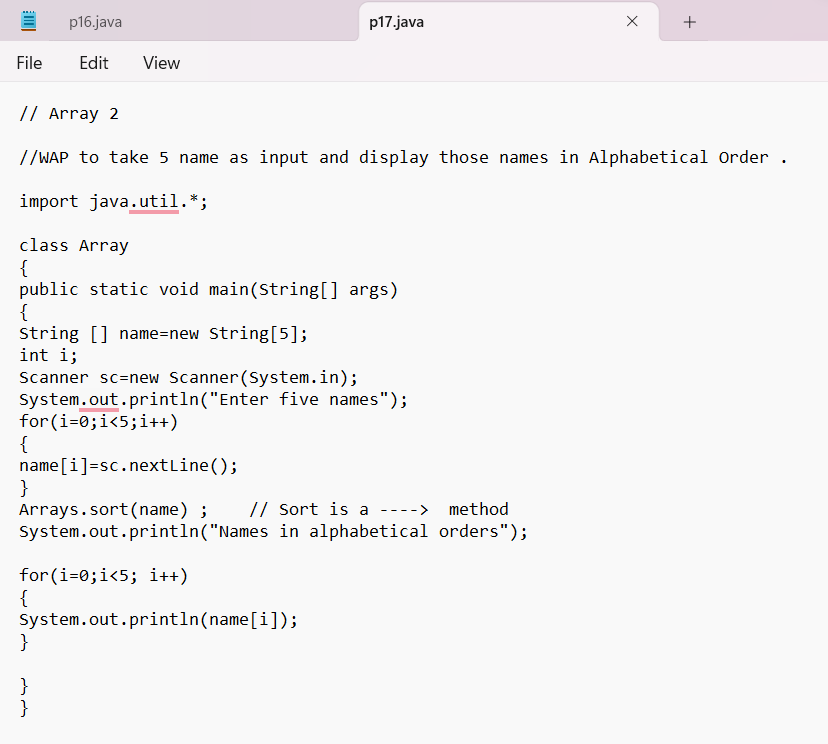
**14.)**

**14.) **

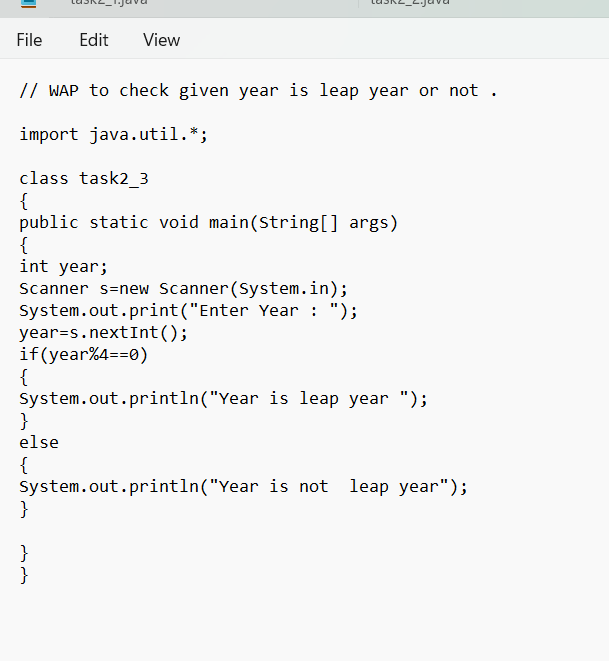
**15.)**

****

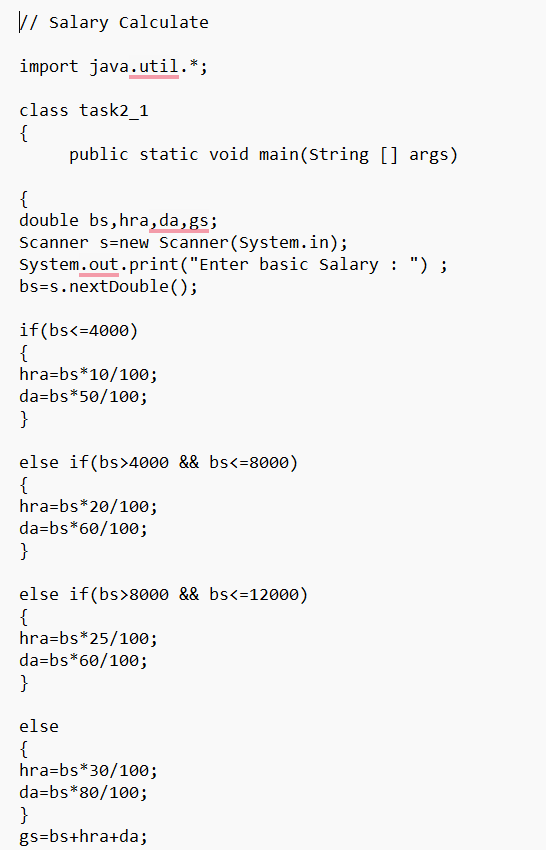
****

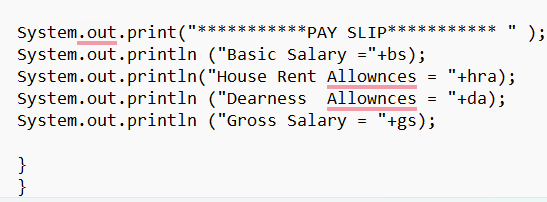
**16.) **

**17.)**

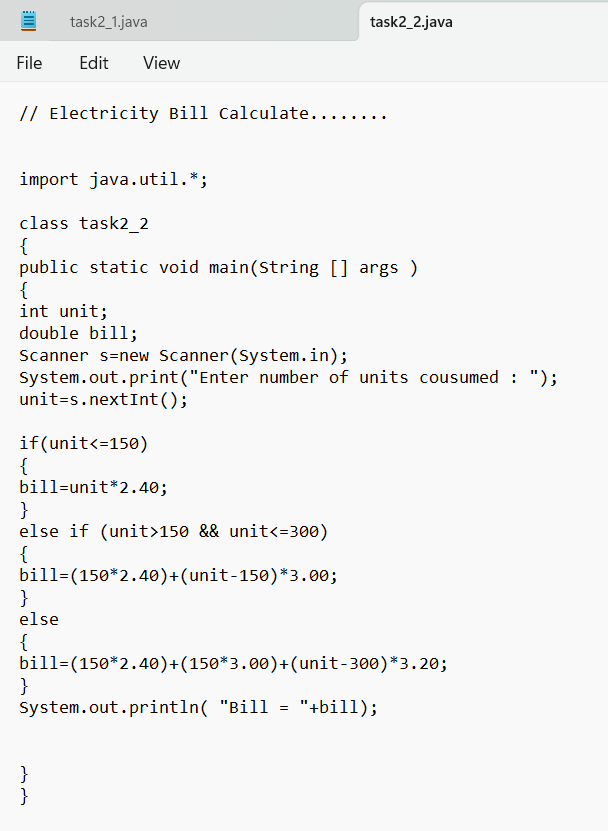
****

**18.)**

****

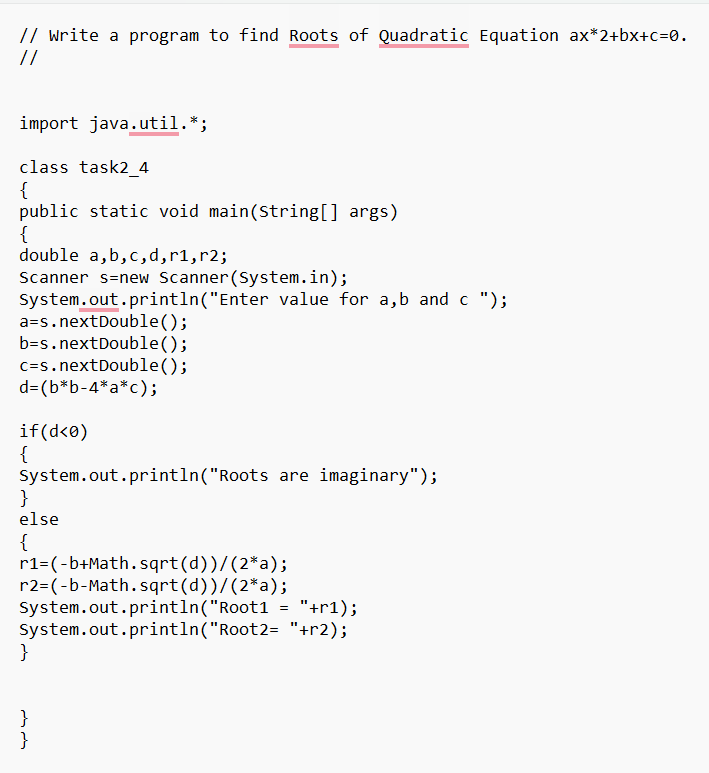
****

**19.)**

****

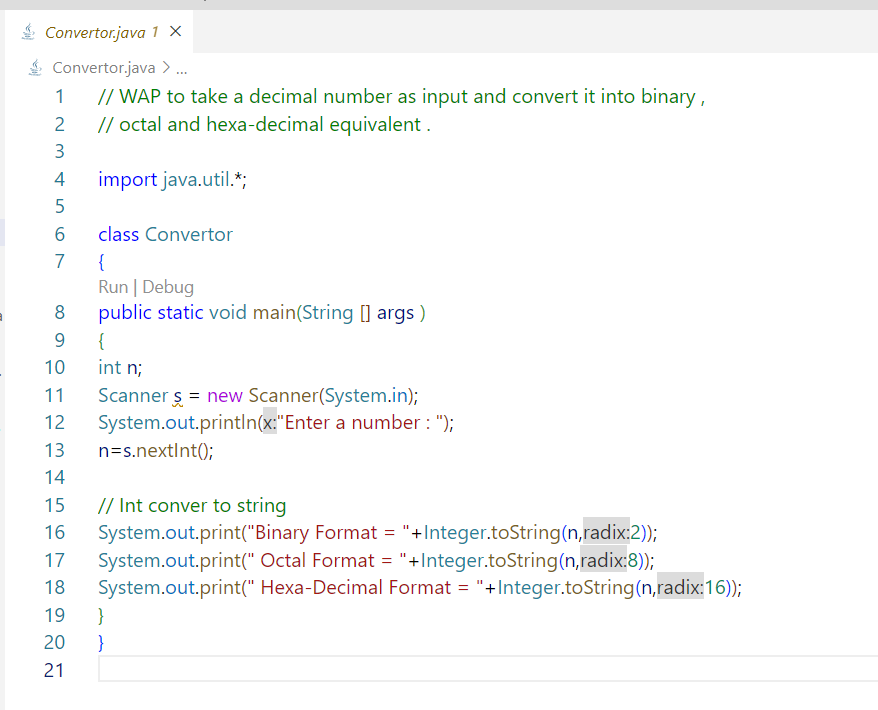
**20.)**

**20.)**

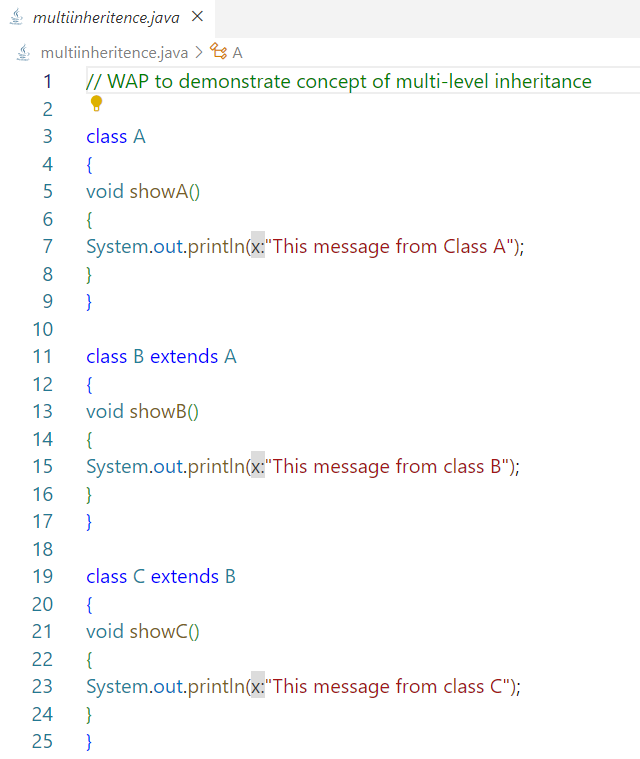
****

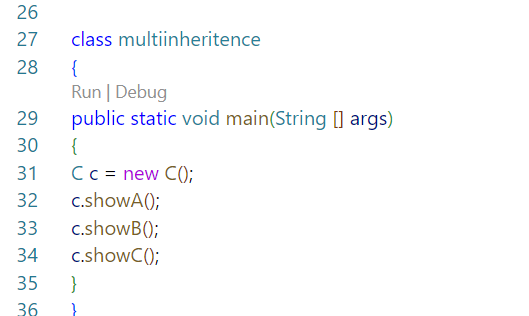
**OOPS**

**1.)**

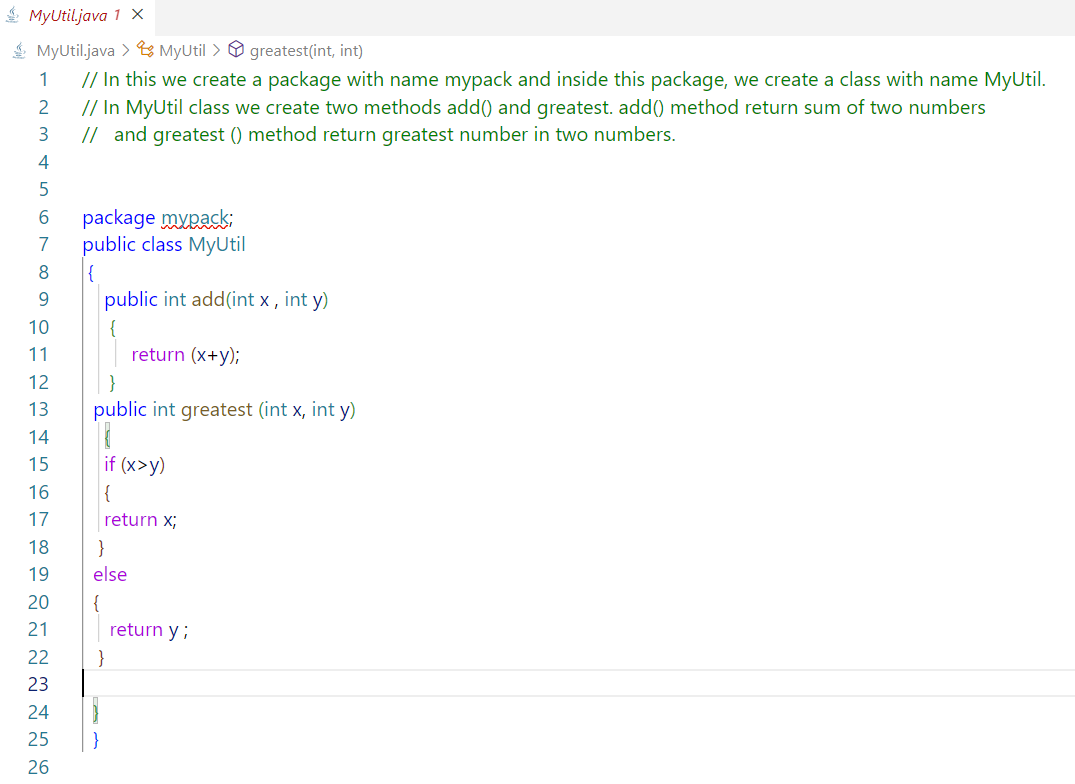
****

**2.)**

****

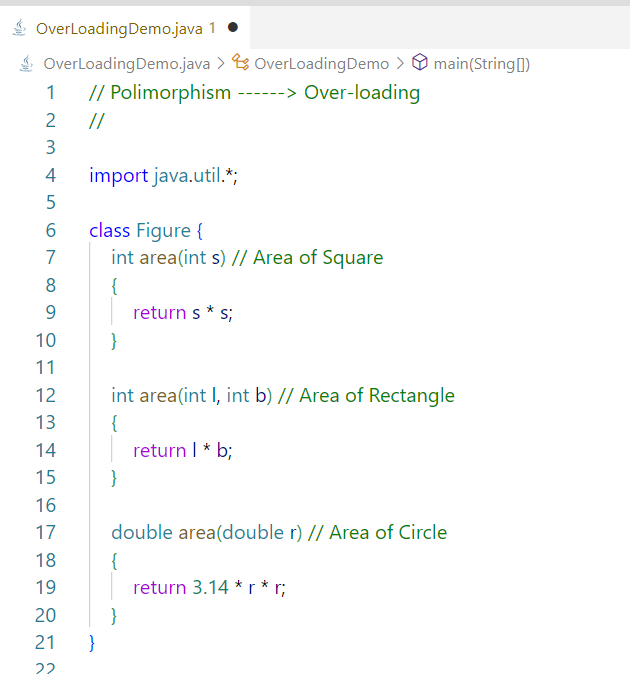
****

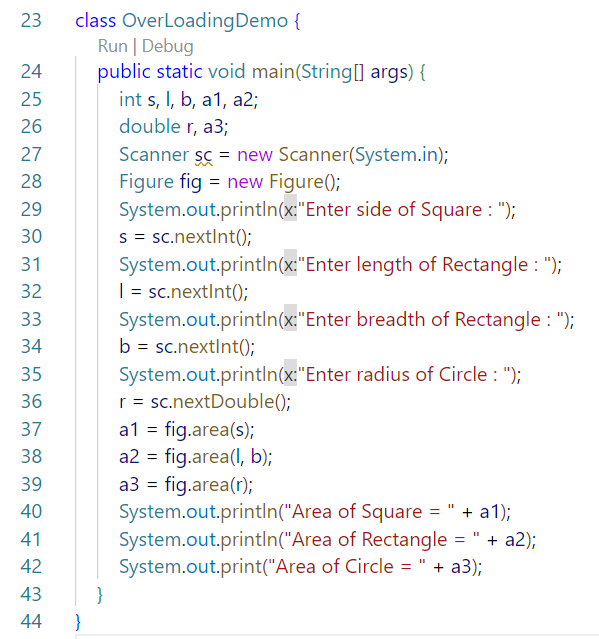
**3.)**

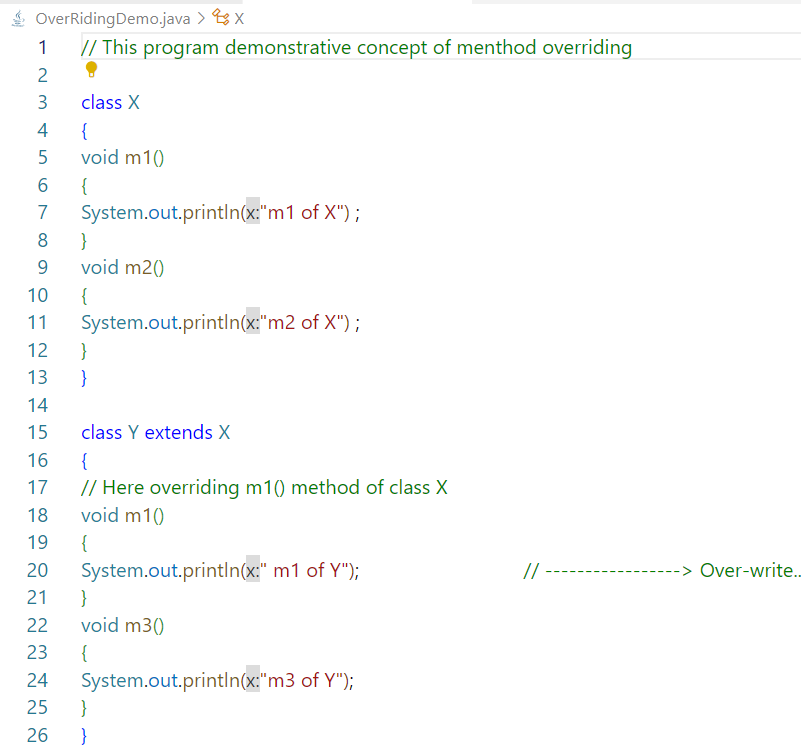
****

**4.)**

**4.)**

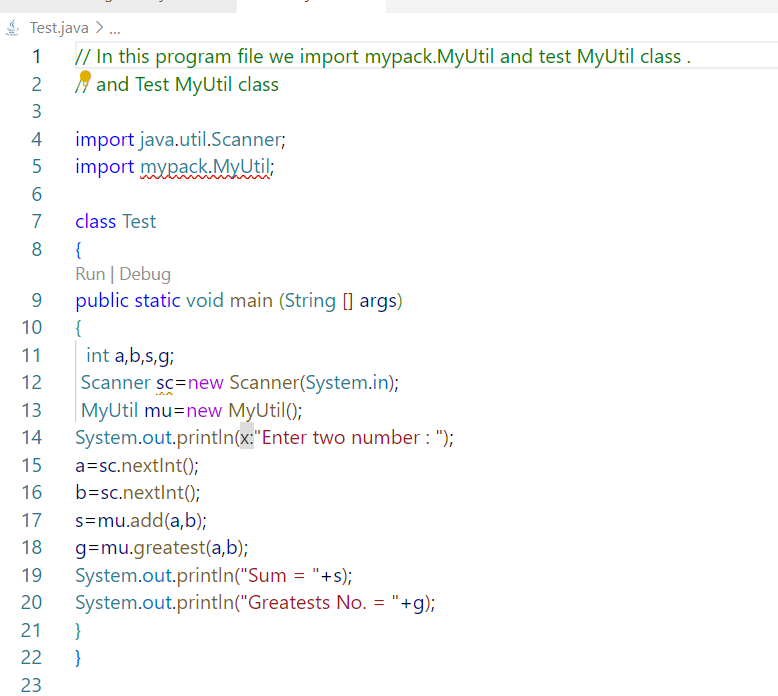
****

****

**5.) **

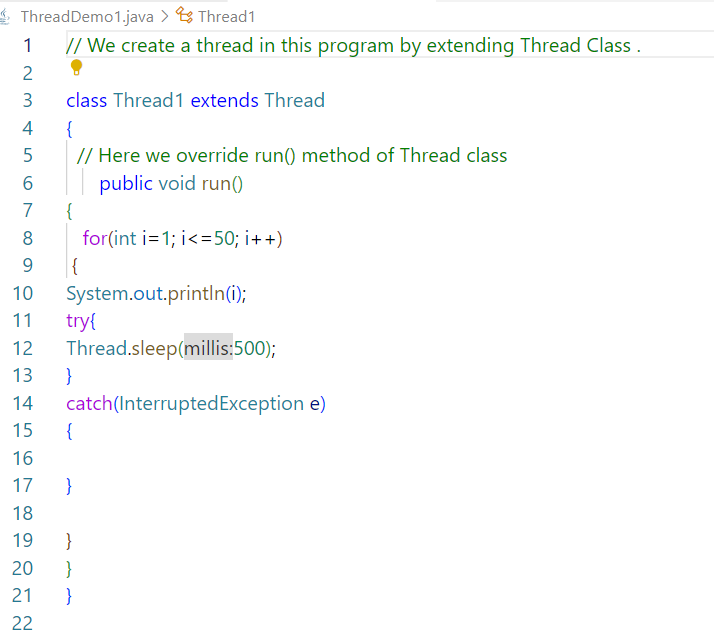
****

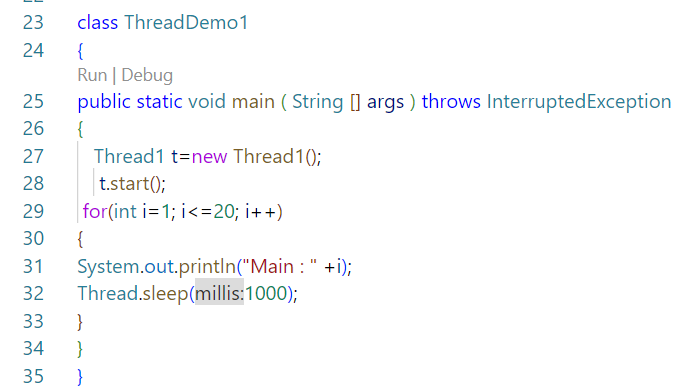
**6.)**

****

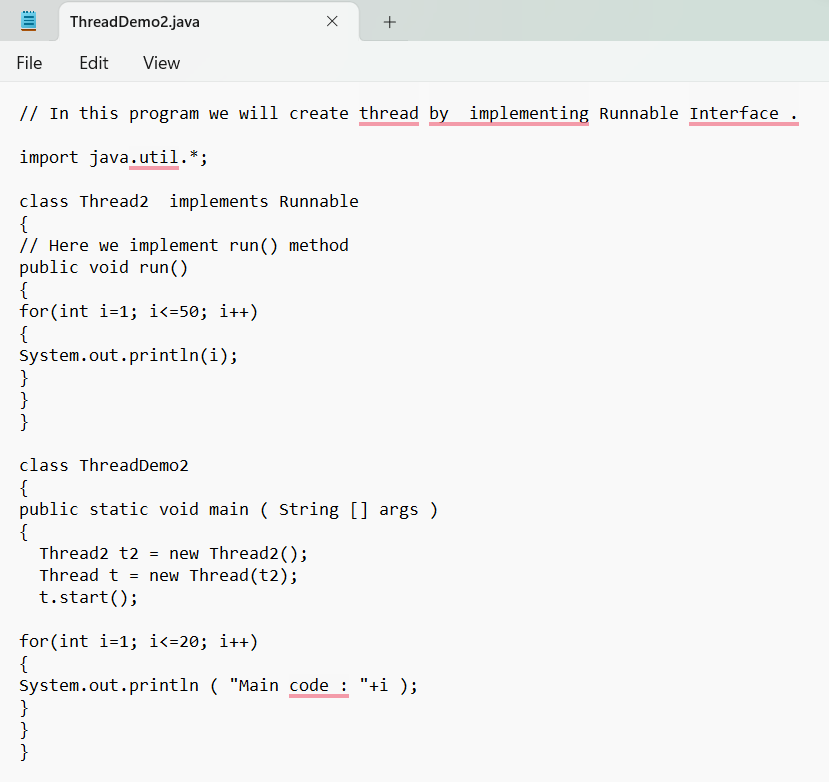
**7.)**

**7.)**

****

****

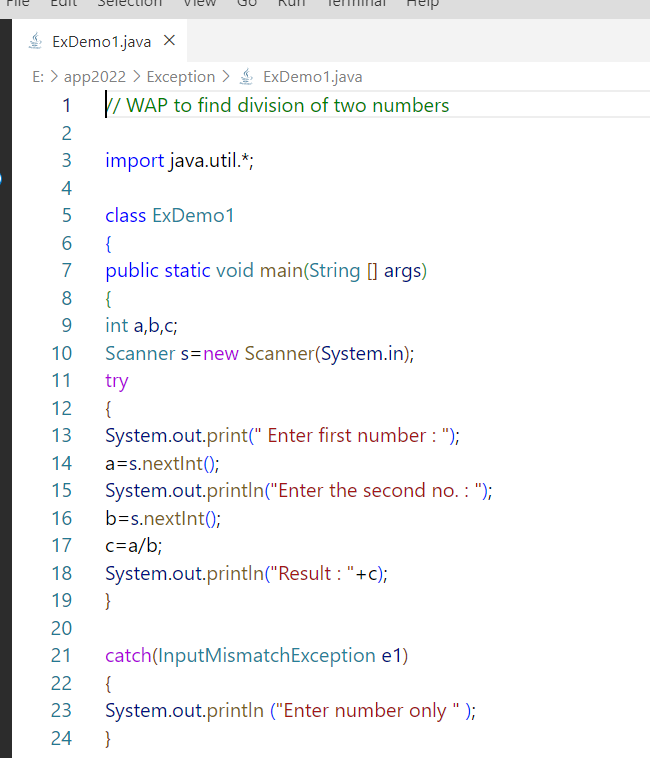
**8.)**

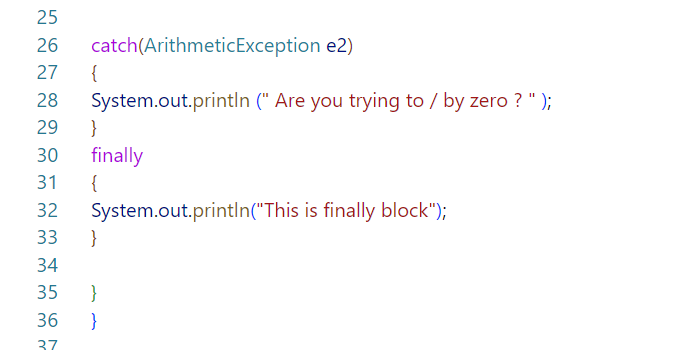
****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

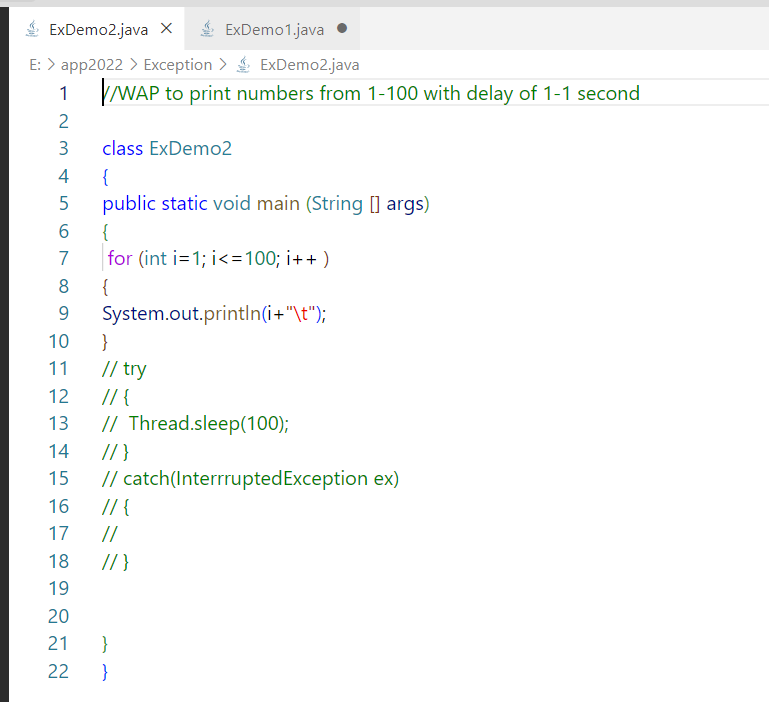
**Exceptions**

**9.)**

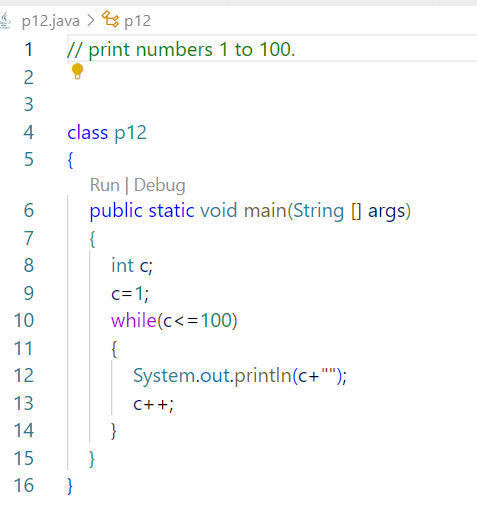
****

****

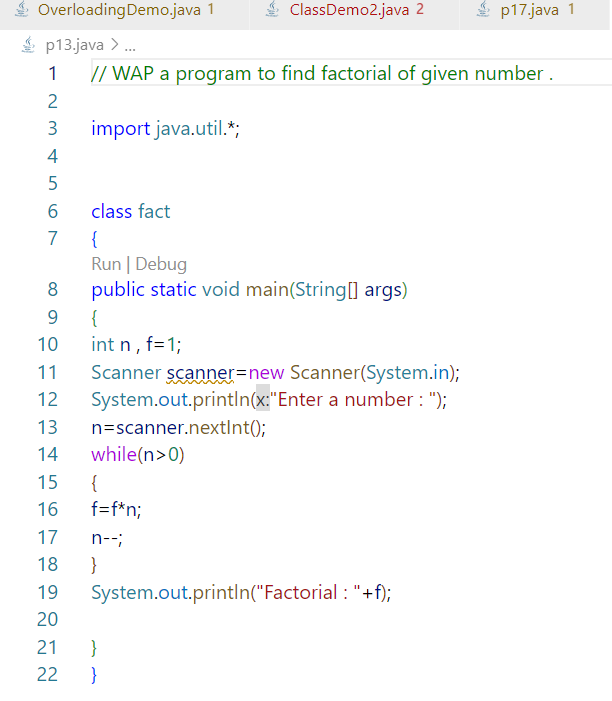
**10.)**

****

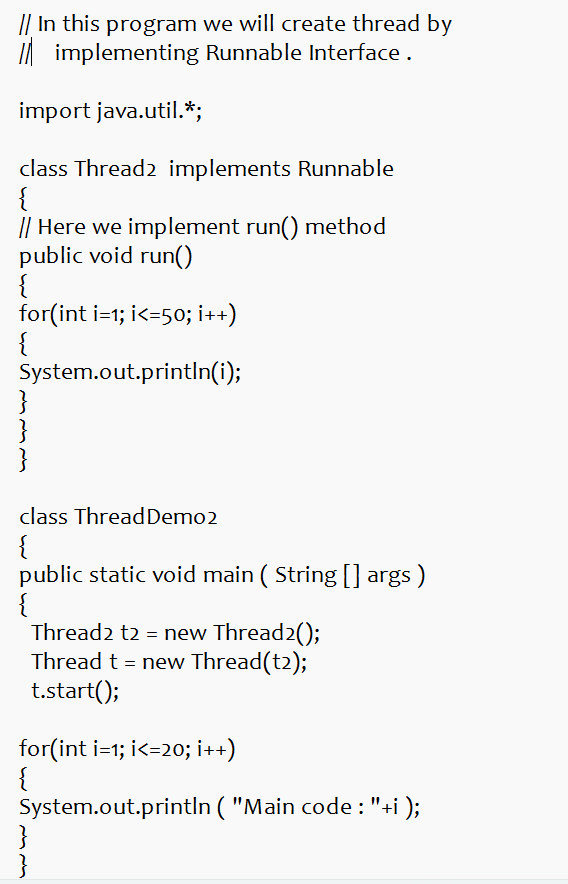
**1.)**

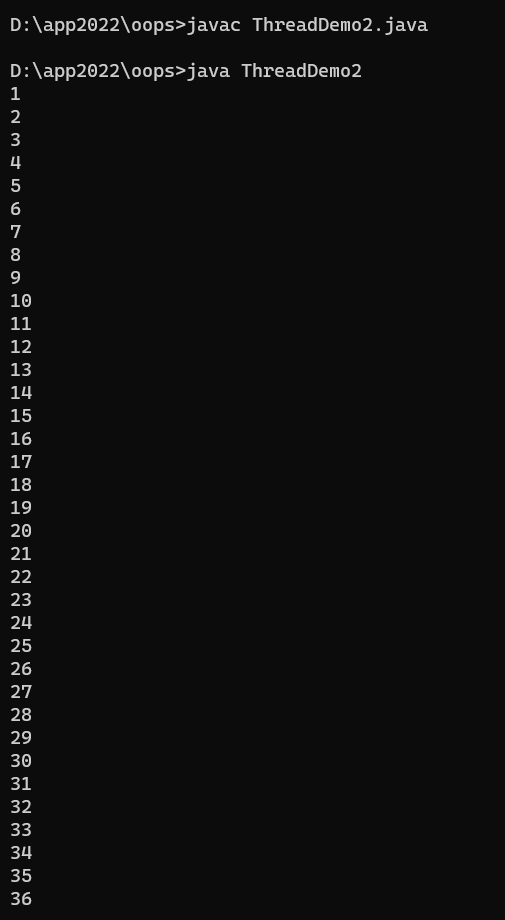
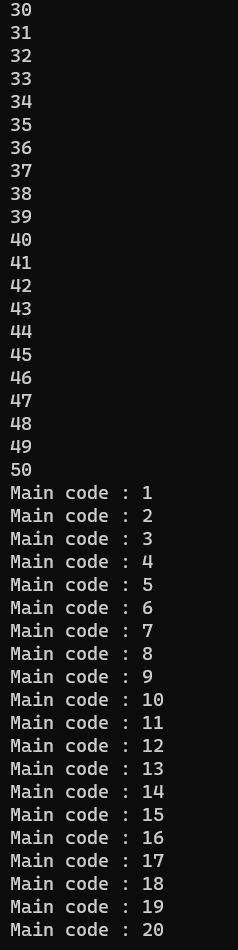
****

**2.)**

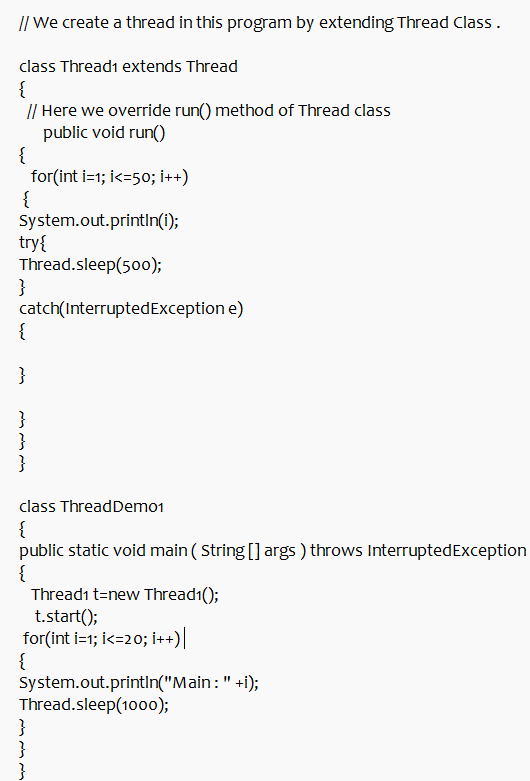
****

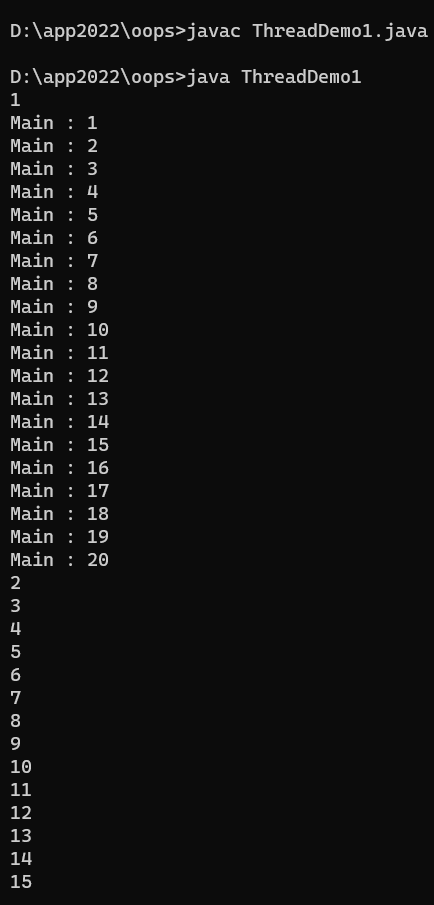
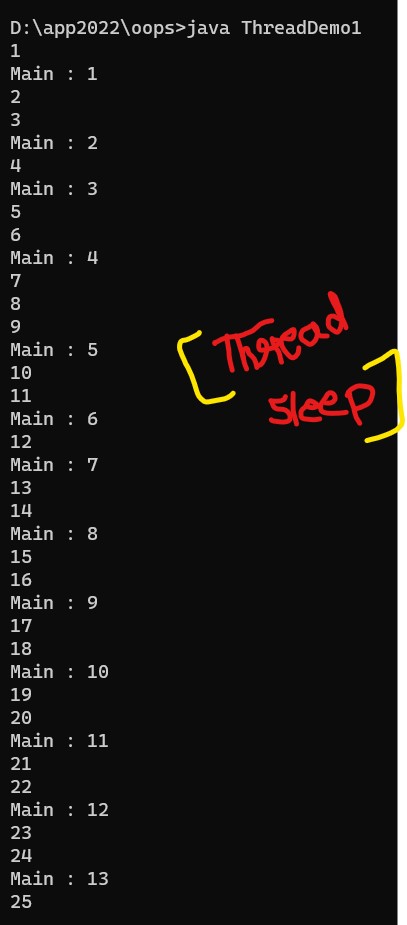
**3.)**



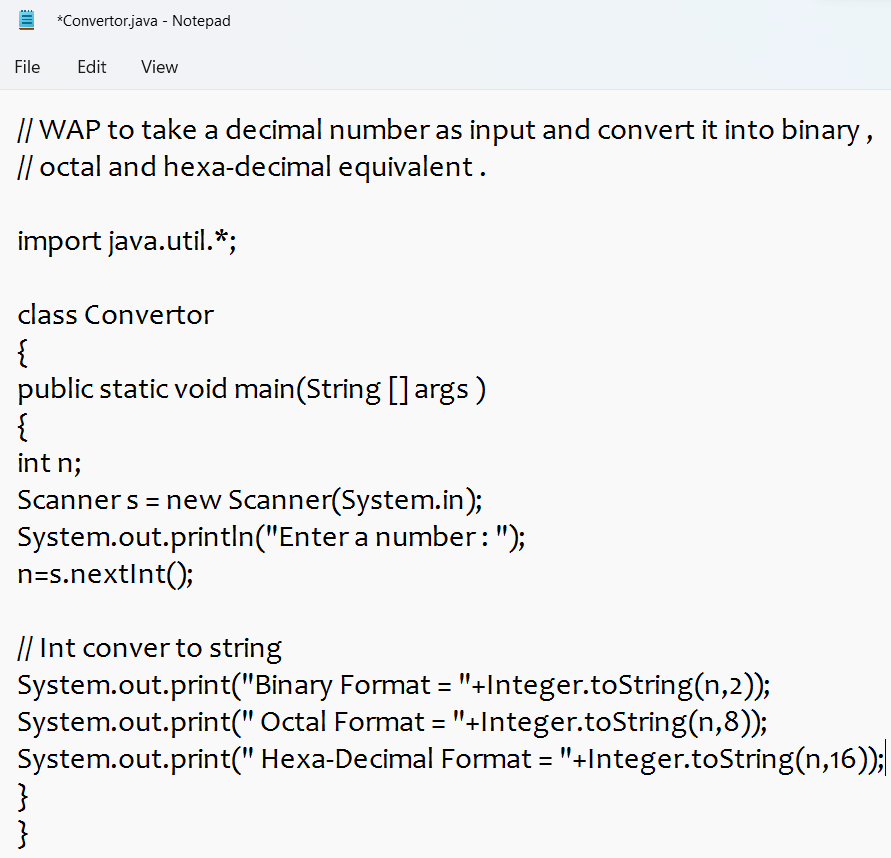
 

**4.)**

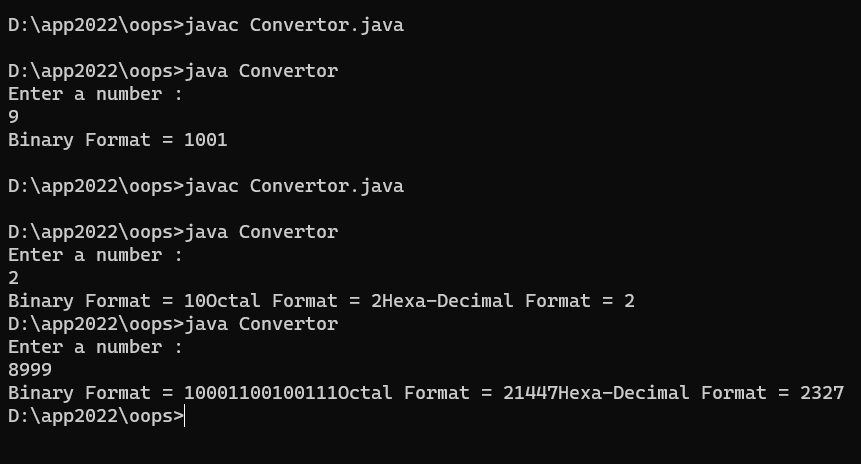


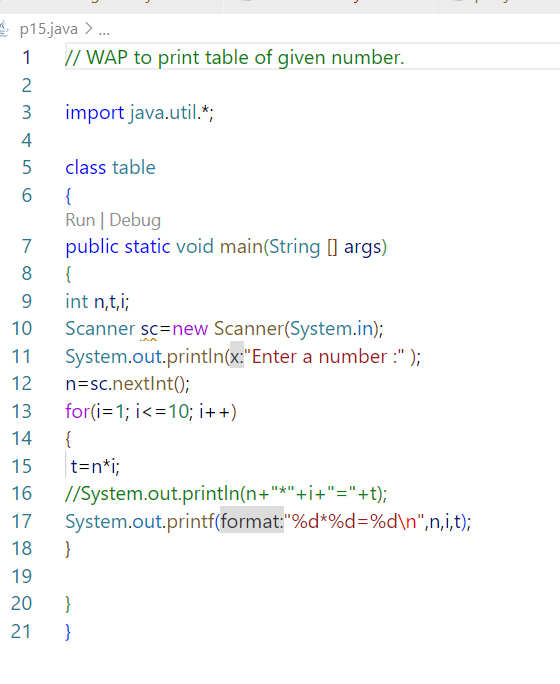
**5.)**



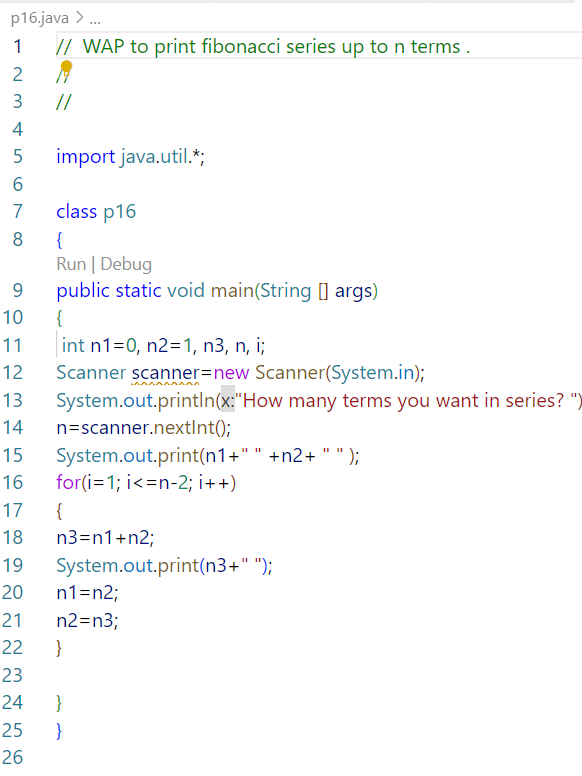




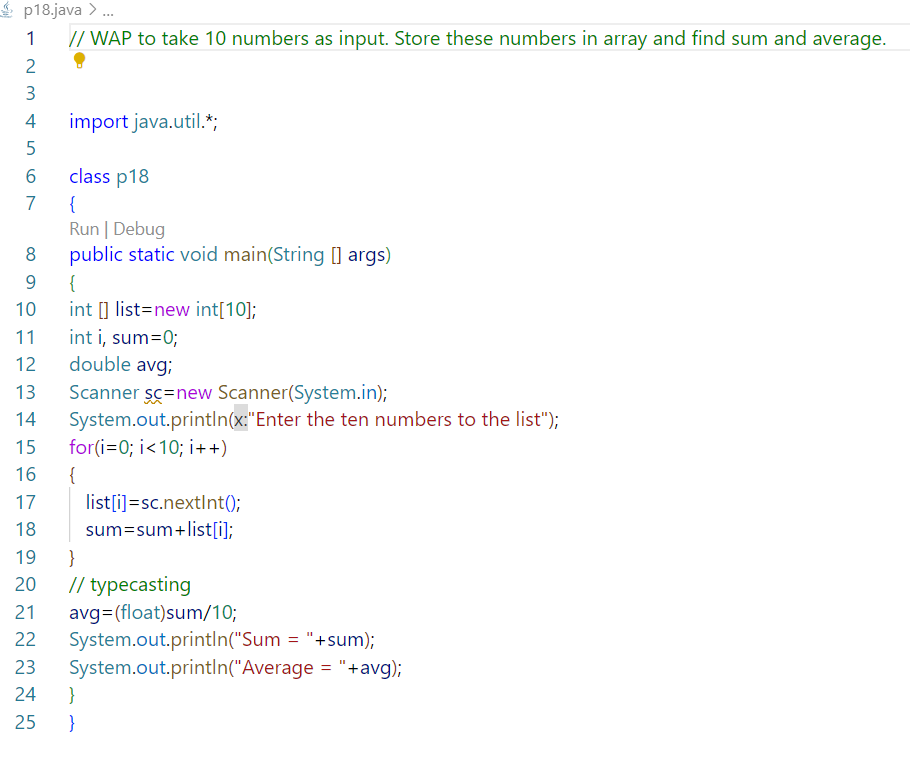
**6.)**

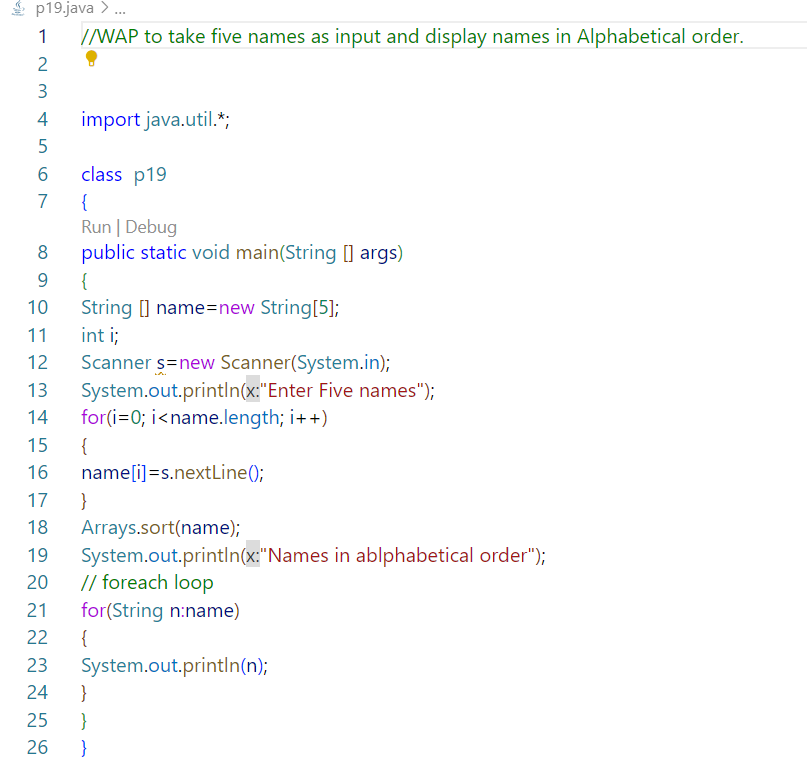
****

**7.)**

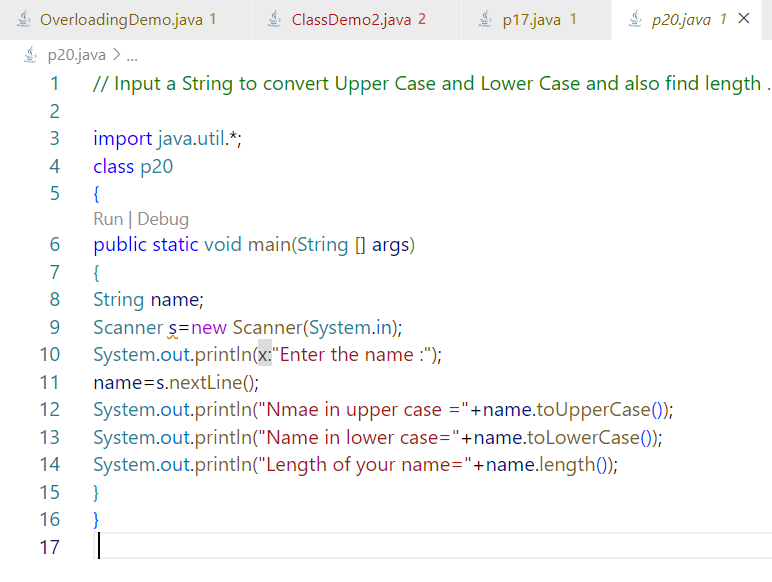
****

**8.)**

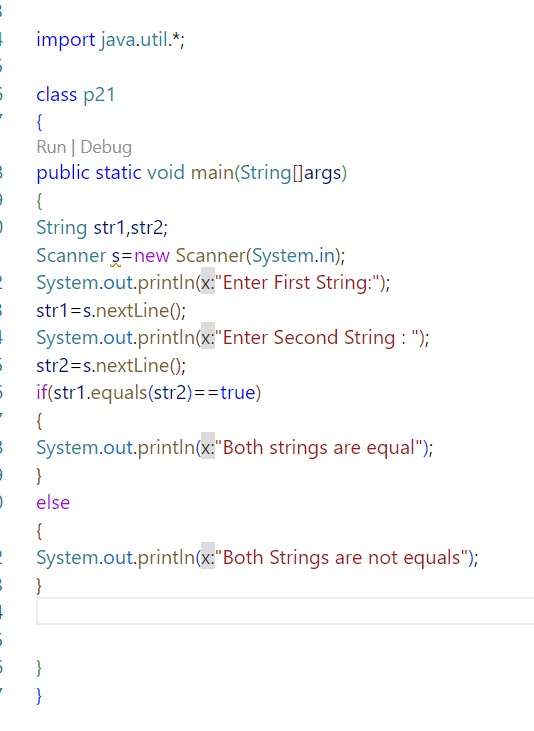
****

**9.) **

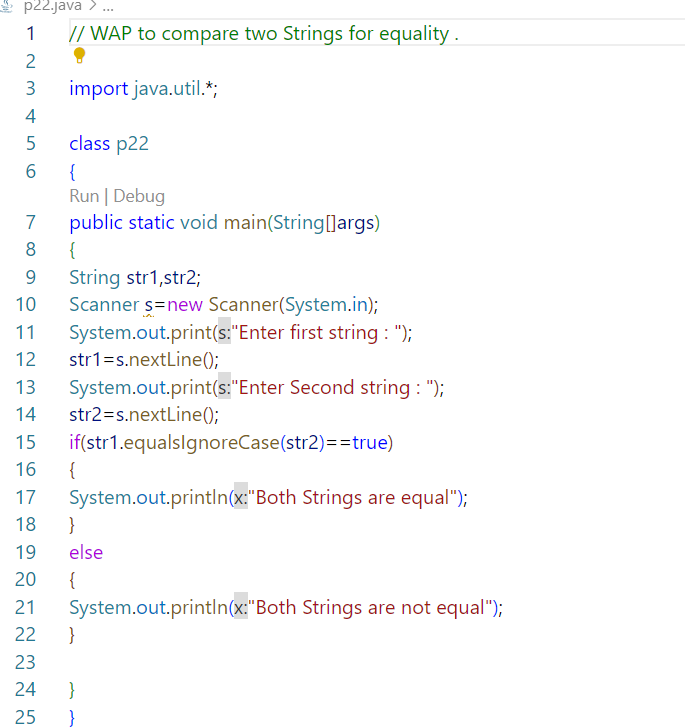
**10.)**

****

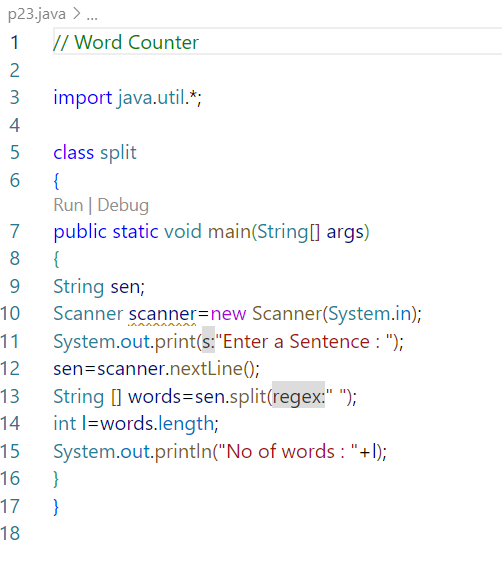
**11.)**

****

**12.)**

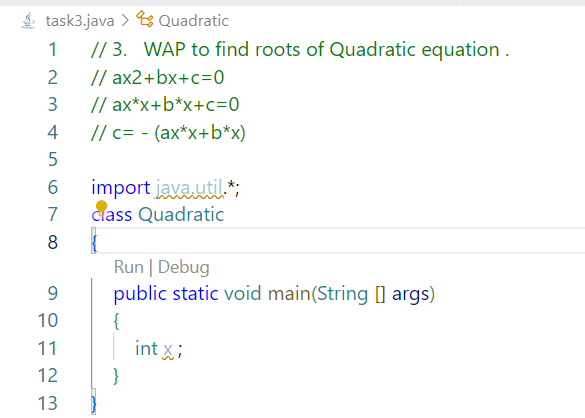
****

**13.)**

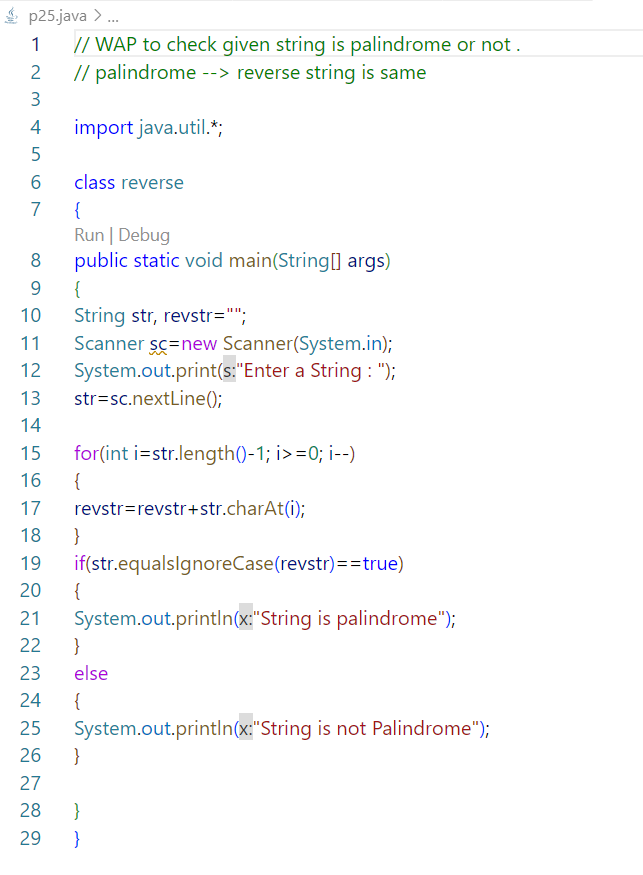
****

**14.) **

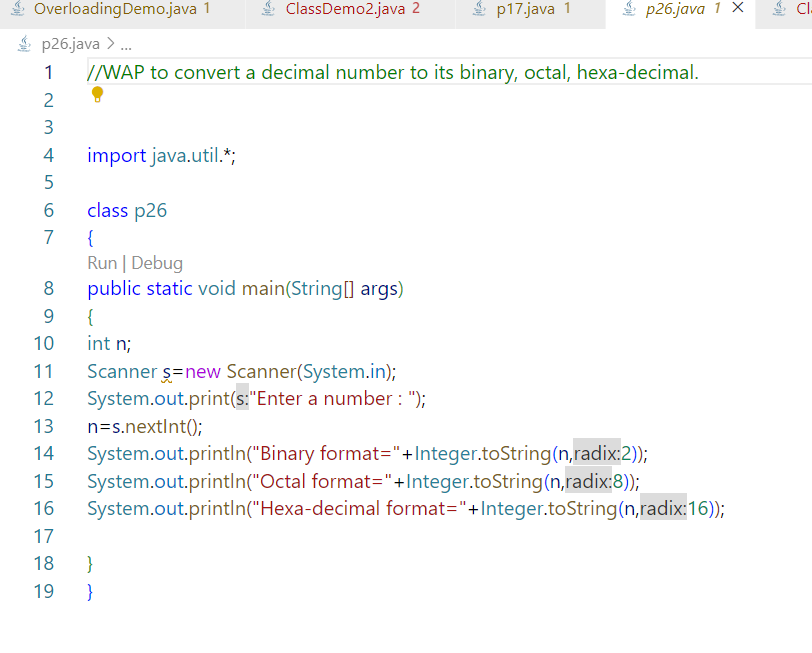
**15.)**

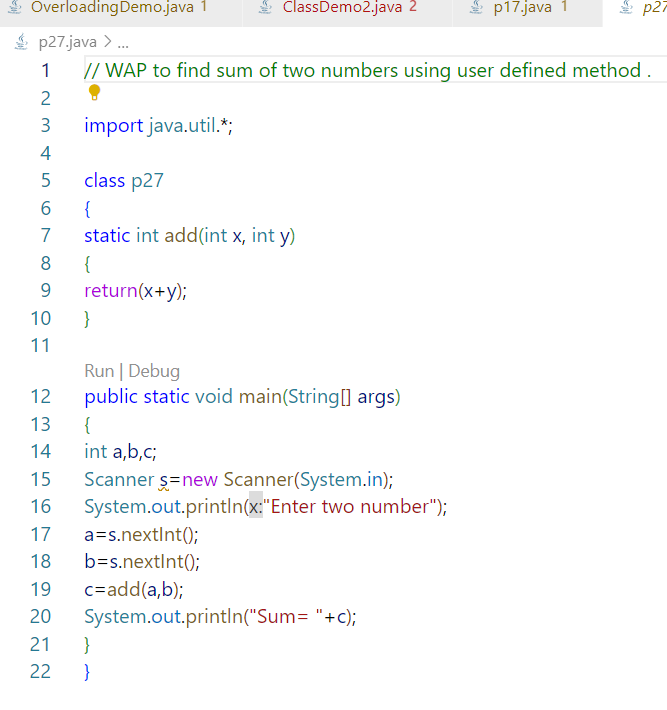
****

**16.)**

****

**17.)**

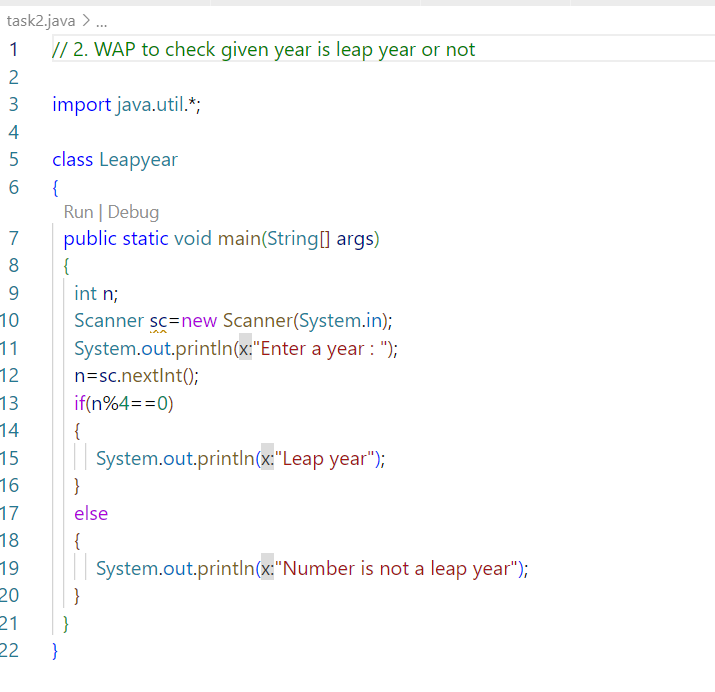
****

**18.) **

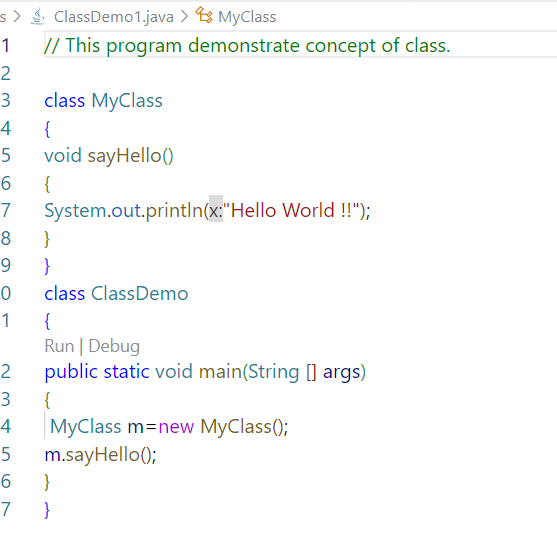
**19.)**

****

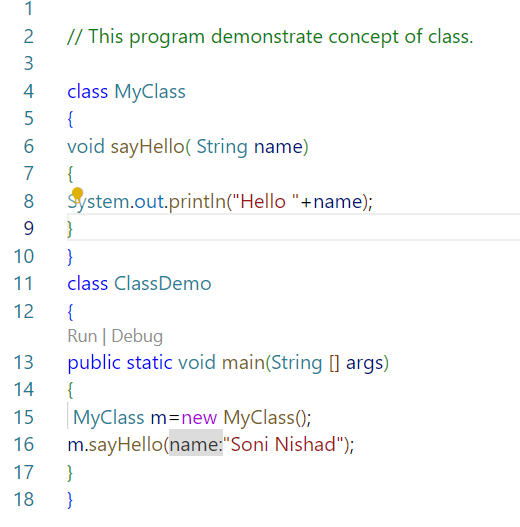
**20.)**

****

**21.)**

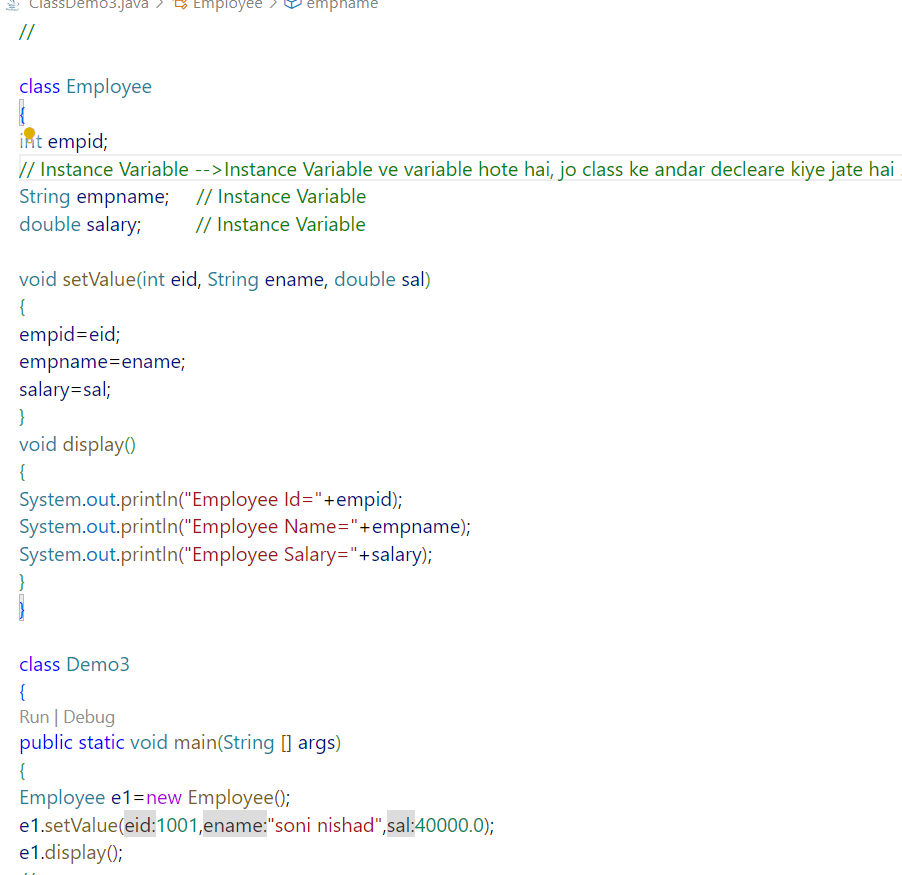
****

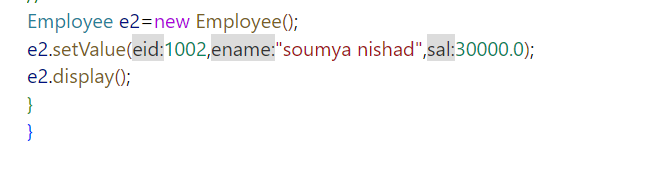
**22.)**

****

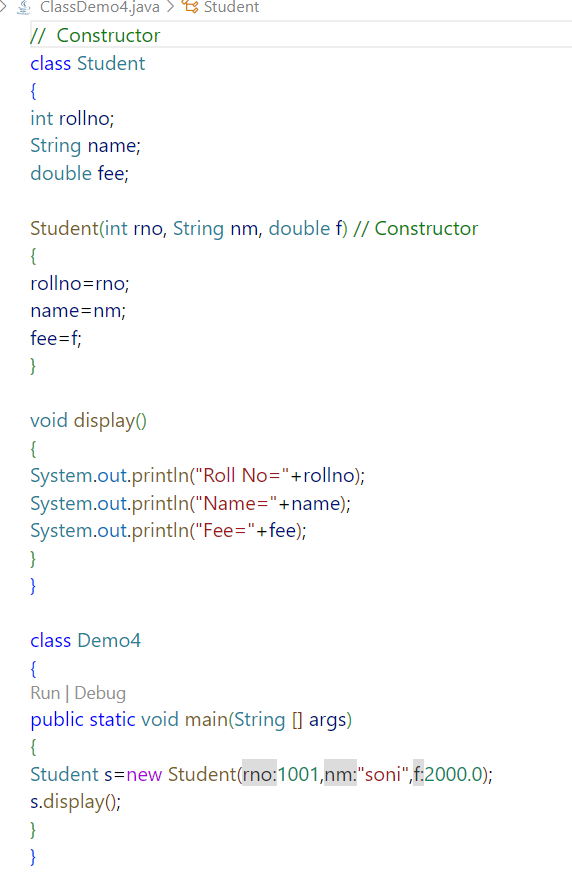
**23.)**

**24.)**

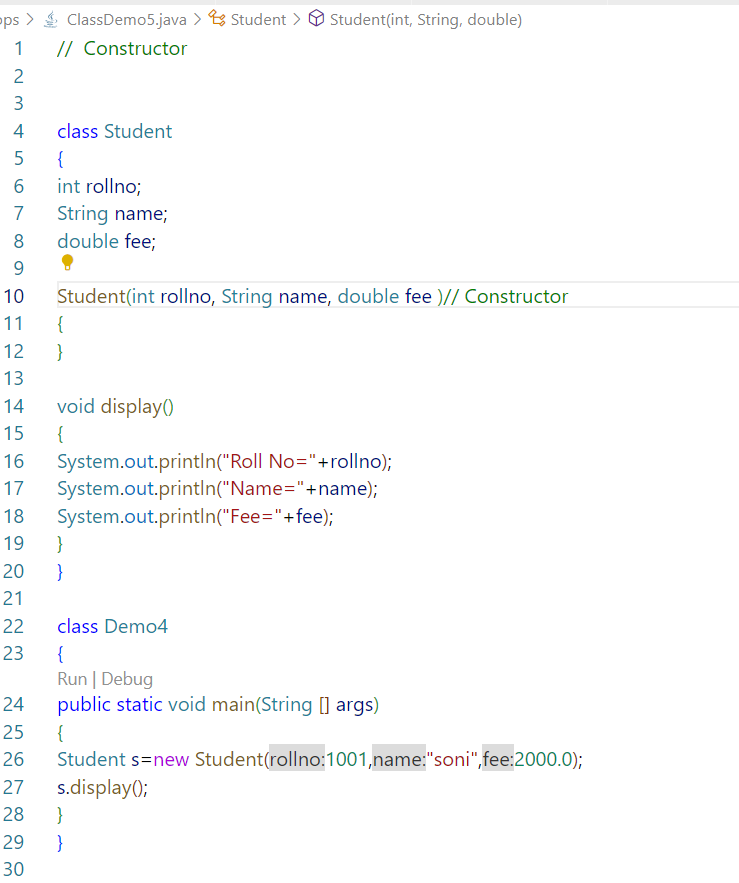
****

****

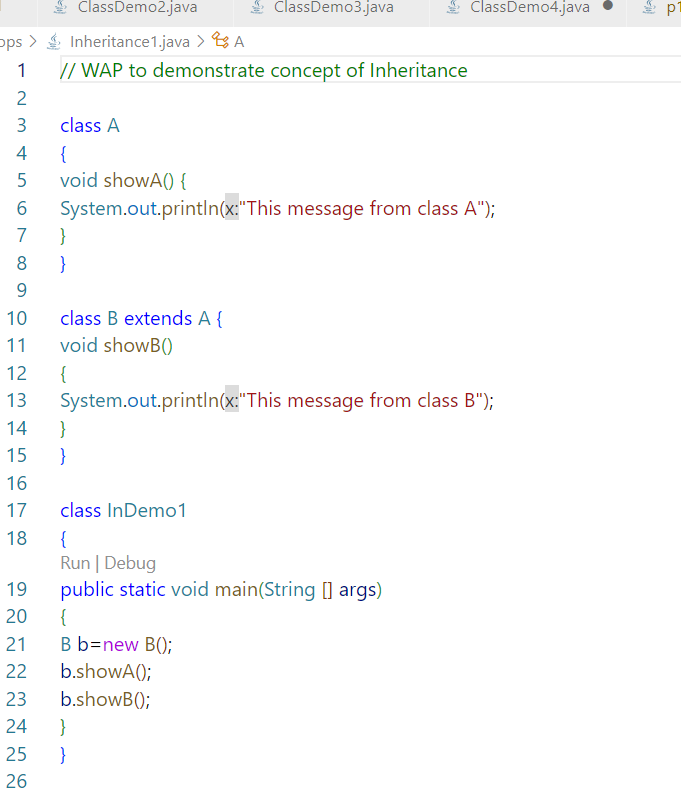
**24.)**

****

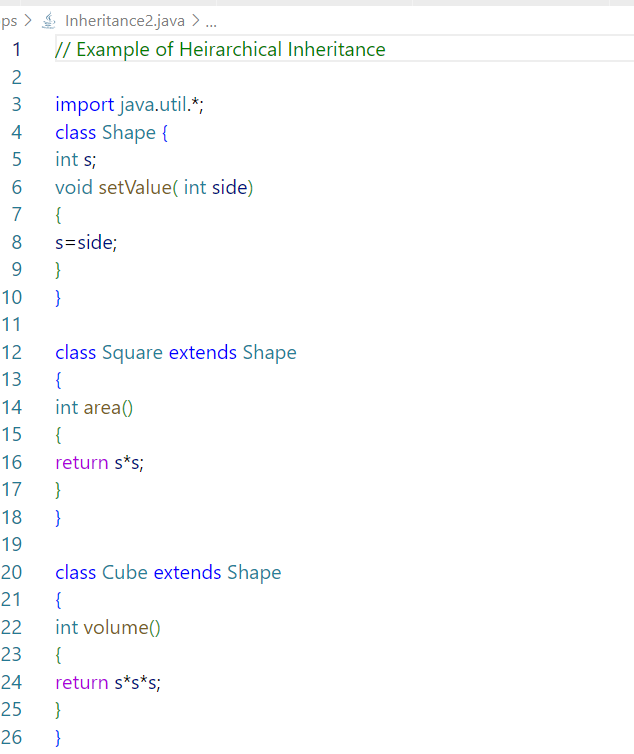
**25.)**

****

**26.)**

****

**27.)**

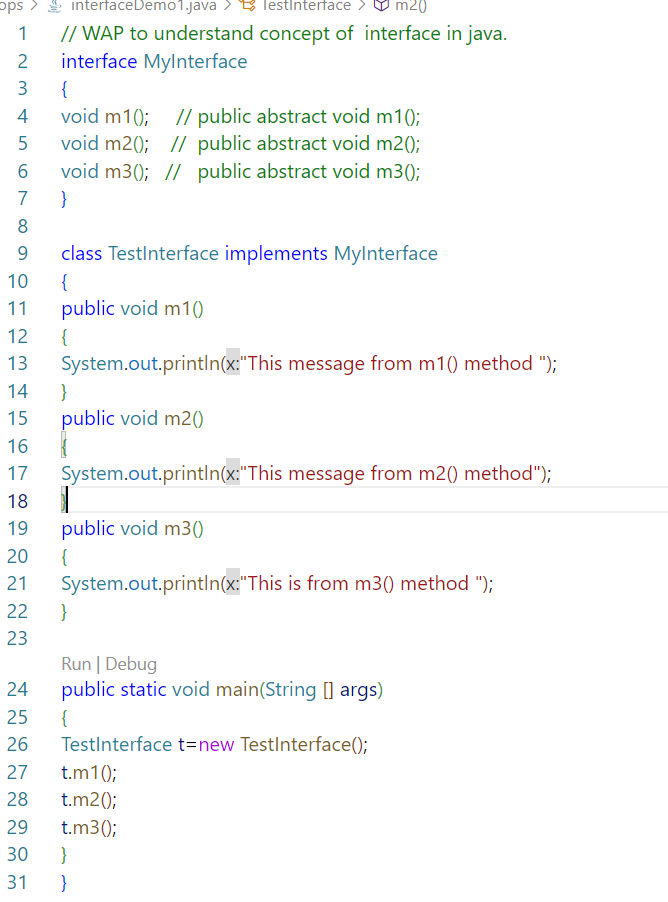
****

****

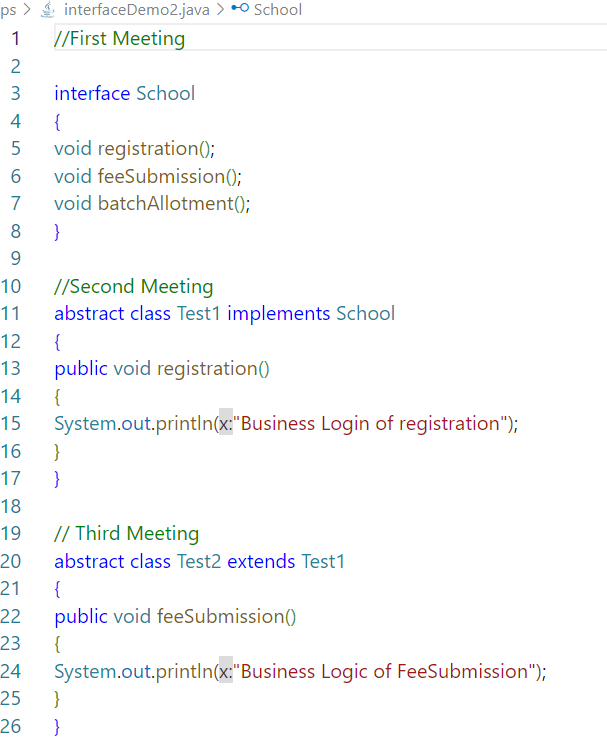
**28.)**

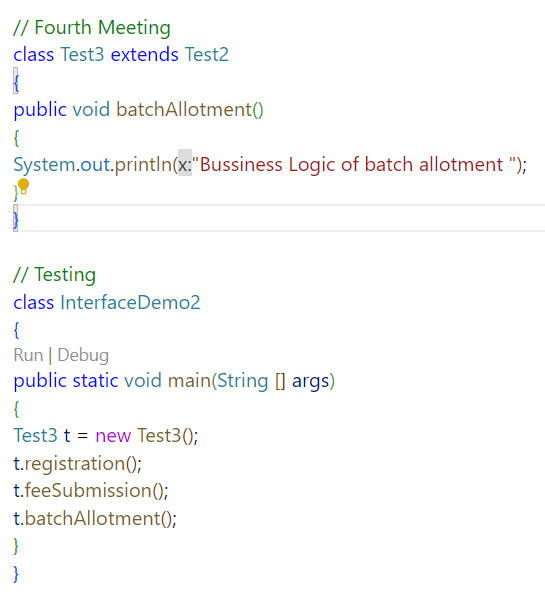
****

**29.)**

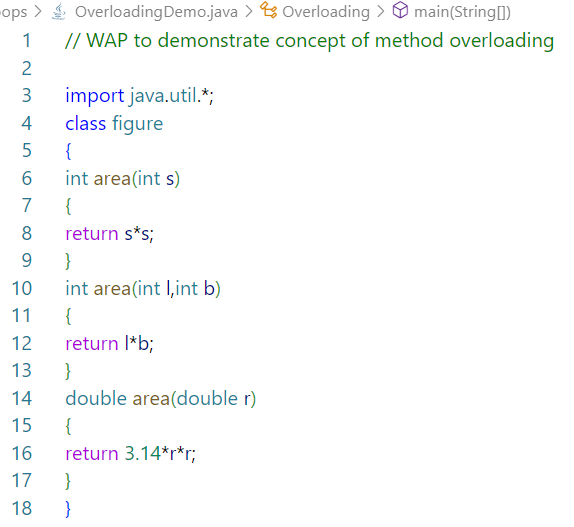
****

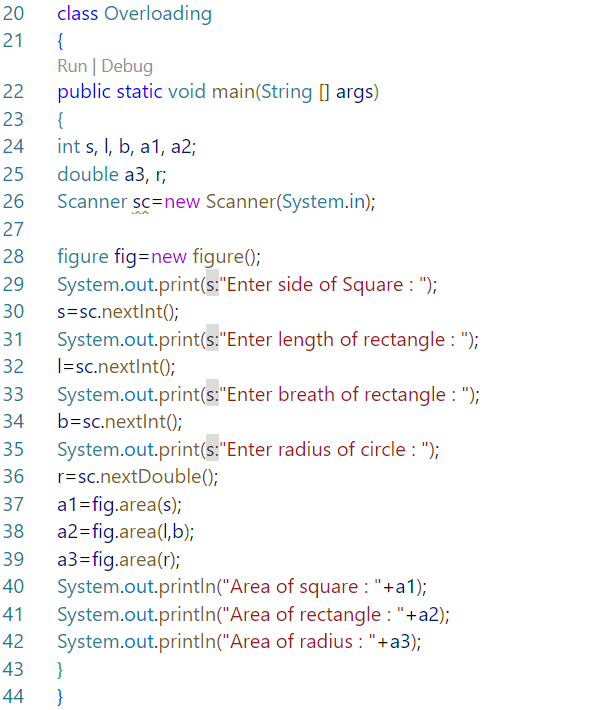
**30.)**

****

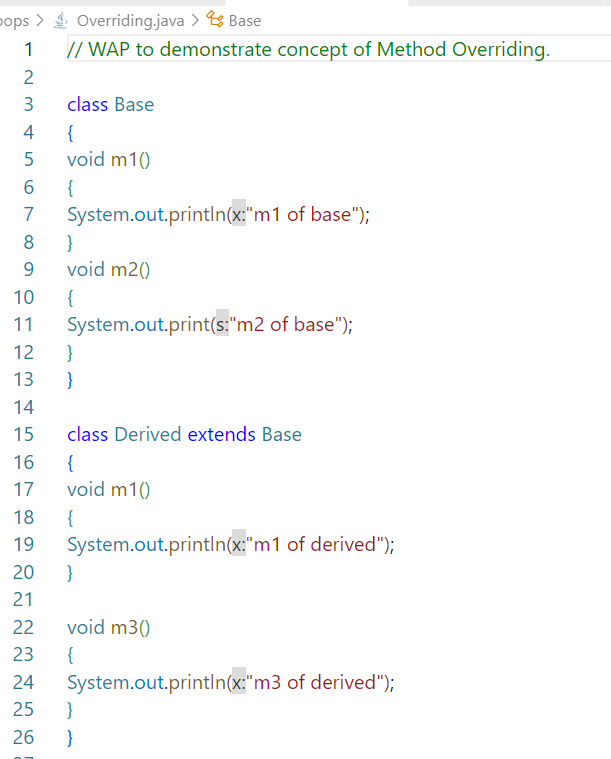
****

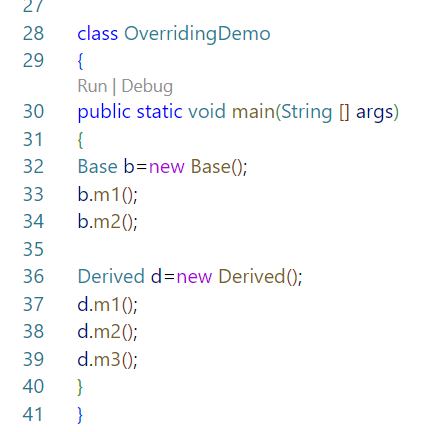
**31.)**

****

****

**32.)**

****

****