***Java Tutorial***

Topics covered:

Java - Home

Java - Overview

Java - Environment Setup

Java - Basic Syntax

Java - Object & Classes

Java – Data types

Java - variables

Java - Operators

Java – Control Statements

Java – User inputs in java

Java - Methods

Java - Constructors

Java – static keyword

Java – this keyword

Java - Inheritance

Java - Polymorphism

Java - Abstraction

Java - Packages

Java – Modifiers

Java - Encapsulation

Java – Strings

Java – Arrays

Java – Exception Handling

Java - Regular Expressions

Java – Date and time

Java – Inner classes

Java – Data Structures

Java – Multithreading

Java - Networking

Java – Sending Email

Java – Applets

Java – Documentation

Java – Collection and Generics

Java Useful Resources

***Home***

* **What is Java?**

Java is a high level ,object oriented Programming Language.

Java Was developed in year **1995** by  **Sun Microsystems**. **James Gosling** is known as the father of java.

* **Features of Java:**

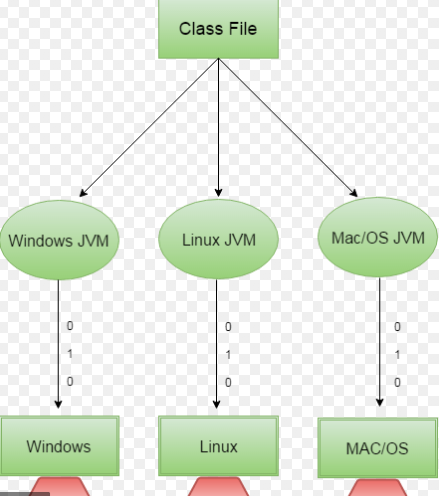
1.**Simple** – Java is a Simple Programming language. The code written in java can be understood easily.

2.**Object Oriented** –Java is Object Oriented Programming Language. In Java Everything is an object.

2.**Complied and Interpreted** – Java code is Compiled and and Interpreted. Java code is first compiled and compiler convert this code into byte code. This byte code is given to the machine using Interpreter.

4.**Platform Independent and** potable - Java is platform independent. Once java code is written it can be run on any platform. JVM(Java Virtual Machine) is different for different Machine.

After converting code into Byte Code the interpreter gives this code to JVM for Execution.



5.**Multithreaded** - Multiple threads can be run simultaneously in java.This concept is known as Multithreading. Due to this java is faster Programming language.

6.**Secure** – Java does not use Runtime Environment of OS(Operating System).It Uses its own JRE(Java Runtime Environment).Therefore it is Secure.

7.**Robust** – Robust means Strong. Since,Java is not using Explicit pointers therefore it is more Secure.

Also Java has automatic Garbage collection.

It has Strong Memory Management.

8.**Architecture Neutral** - The features of java does not depends upon the features of Machine.

For Example,In C language the the size of an integer is 2 bytes for 32 bit architechture but for 64 bits integer size is 4 bytes. In Java Integer size does not depends upon architechture.

* **Applications of Java:**

There are mainly four Applications of Java.

1. Standalone Application : Standalone applications are windows based application. Examples of

Standalone/Windows Application are Notepad, Calculator. These applications are built using Swing .

2.Mobile Application : Java is used for creating mobile application. For creating mobile application Android Studio is used. Android studio is using a java programming language.

3.Web Application : Java is used for creating Websites . For creation of website Java uses Hibernate, Servlet etc.

4.Enterprises Application : Enterprise Applications can be designed using Java. JavaEE is used for Enterprise Application.

***Overview***

* **History Of Java:**
* Creating a Programming language which is simple ,Robust, Object-Oriented, Distributed, platform independent and portable was the principle of Sun Microsystem.
* The **“Green Team“** of Sun microsystem initiated the project in 1991. Initially Java is called as “Green talk” and Extension of the file was “**.gt”**.
* Java Was first developed for Electronic Devices like set-top boxes.
* Name java is afterwards changed to “Oak” Which was the name of a tree outside James Gosling’s Office.
* In 1995 , “Oak” is changed to “Java” because It Was already a trademark by company “Oak Technologies”.
* Then Extension is changed to **“.java”**
* **Tools required for performing Java Programs:**
* Any Operating System Like Windows, Linux.
* Java JDK(Java Development kit).
* Any Compiler/IDE(Where we Can Run our Program) Like Eclipse or Notepad.

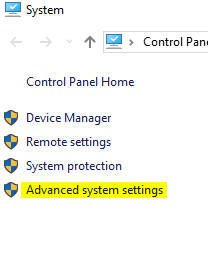
***Environmental Setup***

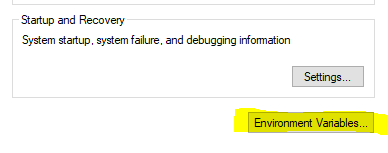
* To run our java program we require JDK(Java Development Kit).
* From The given link you can download Java SE according to your Operating system .

<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>

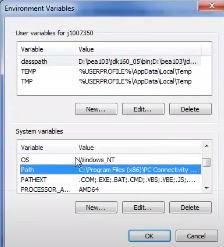
* Once Java is installed on your machine do the following steps to setup an Environment.
* **For Windows:**

1. Firstly, Find your Java file where it is installed. Let’s Say that our Java file is installed at ***“C:\Program files\java\jdk”***
2. **Right** Click on “This PC” .
3. Then go “Advanced System Setting” and then click on “Environment variables”.

****

****

1. Search for a variable name **“Path”** and **double** **click** on it.

****

1. Now just append the path of your java file till bin directory after a semicolon(;) .
2. Now click on “Ok” button.
3. This complete the all steps for java installation.

* **IDE to run JAVA programs**:

**1 .Notepad:** We can run our program simply on Notepad .

**2 . Eclipse:** Eclipse is an IDE which is used to run java programs. for this tutorial we will use Eclipse SE version which is available free.

You can download Eclipse SE from the following link

“ [https://www.eclipse.org](https://www.eclipse.org/)”.

**3.** You can also use any online compiler.

**Eg:** “onlinegdb.com”

***Basic Syntax***

* **Keywords in java :**
* Keywords are some **“reserved”** words in programming language.
* Whenever we give any command related to keyword in java, compiler executes that command and it realizes that this word does a particular function in java.
* There are 48 keywords in java. All keywords are listed below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| abstract | boolean | break | byte | case | catch | char | class |
| continue | default | do | double | else | enum | extends | final |
| finally | float | for | if | implements | import | instanceof | int |
| interface | long | native | new | null | package | private | protected |
| public | return | short | static | strictfp | super | switch | synchronized |
| this | throw | throws | transient | try | void | volatile | while |

* Before Writing the code let us discuss about the name of a file where we are writing our code.
* The name of our “.**java file”** should be same as that of the “**main”** class.
* Basic Syntax program in java

class **Class\_Name**

{

public static void main(String args[])

{

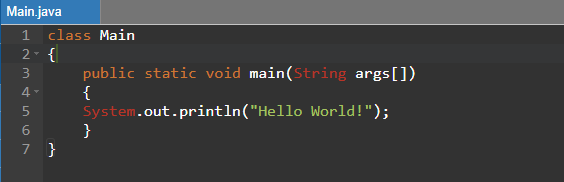
//Code;

}

}

* **Printing “Hello World!” program in java:**

**Code:**

****

**Output:**

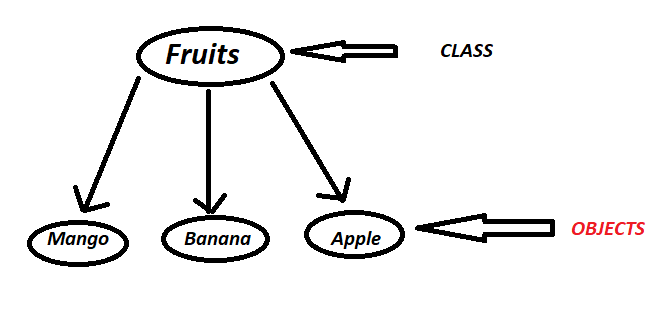
****

* Here, **“Main.java”** is name of the source file.
* “class **Main**” ,in this command. **Main** in name of the class.
* **“System.out.println()” ,** this function in java is used to print any statement.

***Classes and Objects***

* Since, java is an object oriented programming language therefore everything in java program is written under the class.
* **What is class and objects ?**
* Class is a blueprint or we can say it is a prototype for which object is created.
* Class is not a real time entity where as object is a real time entity.
* Using object we can access all the variables and Methods.

Let’s take an example for classes and objects.



Here Class is Fruit Which does not exist in real time but when we talk about Mango ,Banana, Apple It exist in real world .When we talk about fruits we know its properties which is a blueprint of fruits but when we talk about Mango, Banana, Apple ,We can see its colour and taste it therefore it is a real time entity.

Example:

class test

{

//code

}

class Main

{

public static void main(String args[])

test **a**= new test();

}

* In above example, “a” is an object of class “test”.

***Data Types***

* Data type specifies that what kind of data can be stored into the variable.
* For example : **int** age=18 (An integer data is stored in variable.)

**String** Name=”Deepa” (String value is stored into the variable.)

**Boolean** A=True (A Boolean value is stored into the variable.)

* There are mainly two types of data types **1.Primitive 2.Non-Primitive**.
* Following are the primitive data types:

1. byte data type
2. short data type
3. int data type
4. long dataq type
5. long data type
6. Boolean data type
7. float data type
8. double data type
9. char data type

* **byte data type:**
* It is 8 bit primitive data type.
* Its range is between -128 to 127.
* It is 4 times smaller than “**int”** data type

Eg : byte a=14

* Its default value is zero.
* **Short data type:**
* It is 16 bit primitive data type.
* Its range is between -32,768 to 32,767.
* It is 2 times smaller than “**int”** data type.

Eg : short a=20000

* Its default value is zero.
* **Int data type:**
* It is 32 bit primitive data type. It is used to store integer value in variable.
* Its range is between - 2,147,483,648 to 2,147,483,647.

Eg : int a = 200000

* Its default value is zero.
* **Long data type:**
* It is 64 bit primitive data type. It is used to store integer value in variable.its range is more than int data type.
* Its range is between -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

Eg : long a = 2000000

* Its default value is zero
* **boolean data type:**
* It is primitive data type.
* boolean data type stores only two possible values that is **true/false**.

It specifies one bit information but its size can not be defined.

Eg:Boolean a=true.

* **Float data type:**
* It is 32-bit IEEE 754 floating point primitive data type. It is used to store value having decimal point in variable.
* Its range is unlimited.

Eg : float a=25.50

* Its default value is 0.0.
* **Double data type:**
* Double data type is similar to float data type.
* It is 64-bit IEEE 754 floating point primitive data type. It is used to store value having decimal point in variable. Its range is more than float.

Eg : double a=22.50

* Its default value is 0.0.
* **Char data type:**
* It is 16-bit Unicode character primitive data type. It is used to store character value in variable.
* Its range is “\uuuu” to “\ffff”.

double a0=”f”

***Variables:***

* Variables in java are used to store some data.
* There are some rules to write variables in java. They are listed below

1. The name of a variable can not start with a number.
2. Space is not allowed between the name of variable , we can use underscore(\_) in between the name of variable.
3. All variables should start with a letter (A to Z or a to z), currency character ($) or an underscore (\_).
4. Any keyword of java can not be used as variable.
5. Since, java is a case sensitive language therefore variables are also case sensitive.

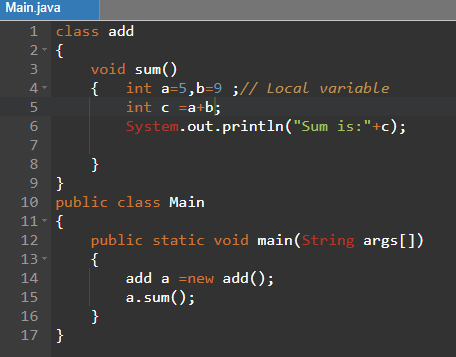
Eg: Abc is different then variable name abc.

* There are three types of variables.

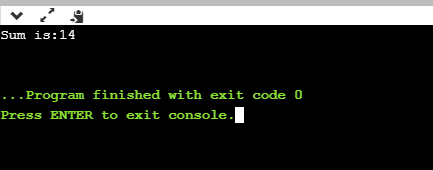
1. **Local variable.**
2. **Instance variable.**
3. **static variable.**

* **Local variable :** These variables are defined inside the method or function and can not be used outside any method. Anywhere inside the method , it can be access.

**Example:**

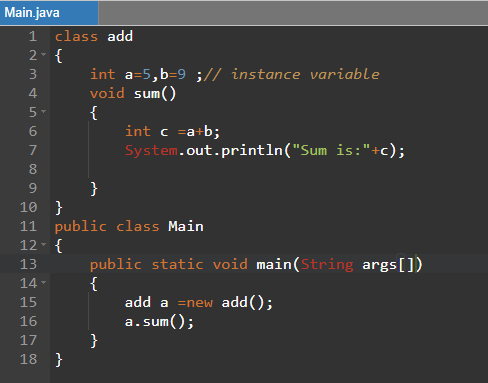


Output:

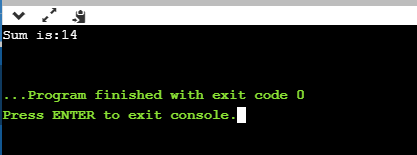


* **Instance variable :** These variables are defined inside the class and above method. Within a class anywhere these variables can be used.

**Example:**

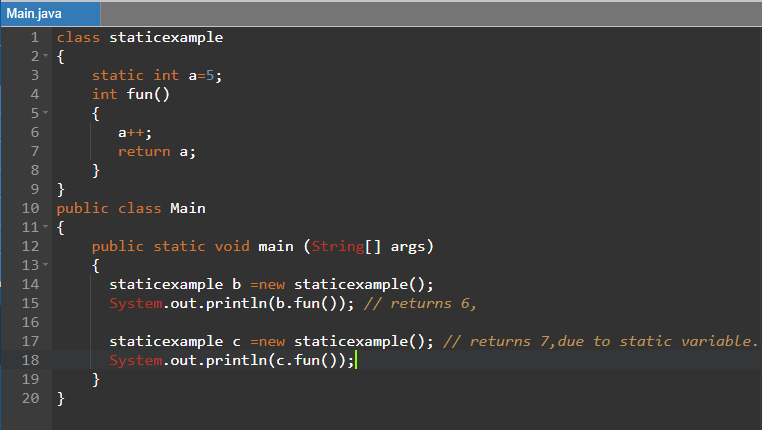
****

**Output:**

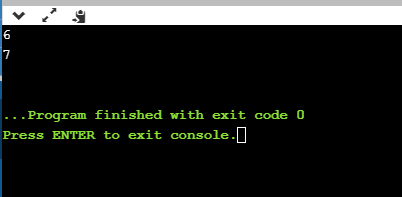
****

* **static variable :** The variable defined with keyword **“static”** is called as static variable. This variable can not be used as local variable.

**Ex ample:**

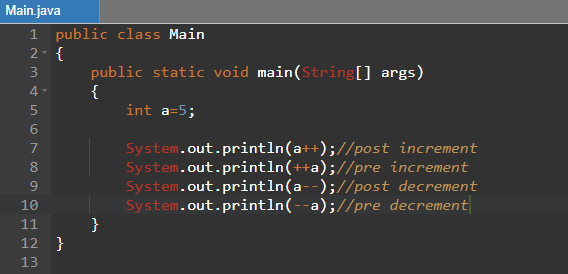
****

**Output:**

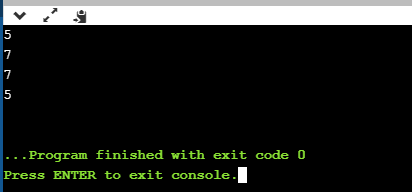
****

***Operators***

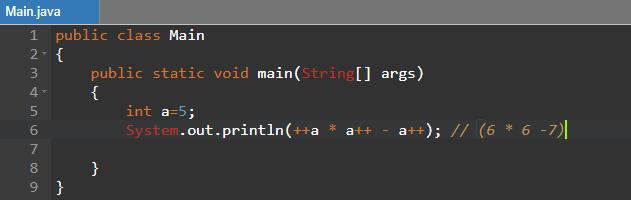
* Operators in java are used to perform some operations on variable. Eg : +,-,%,\* etc.
* Following are the operators in java.
* Unary operator.
* Assignment operator.
* Arithmetic operator.
* Relational operator.
* Shift operator.
* Logical operator.
* Bitwise operator.
* Ternary operator.
* **Unary operator:**
* Unary operator is used only for one variable.
* It is used to increment/decrement, negate the value, Inverting the Boolean expression.
* Example 1:

****

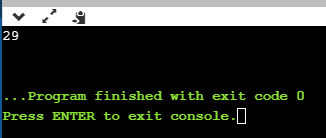
**Output:**

****

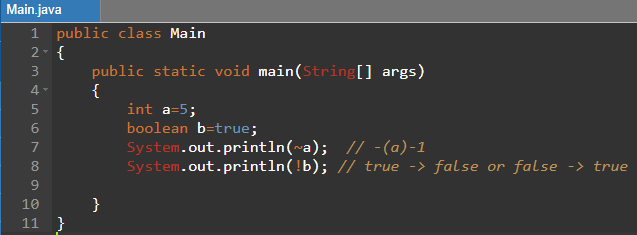
**Example 2:**

****

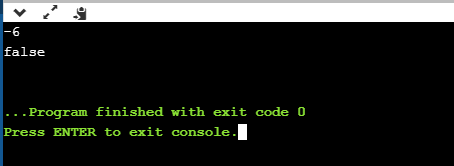
**Output:**

****

**Example 3:**

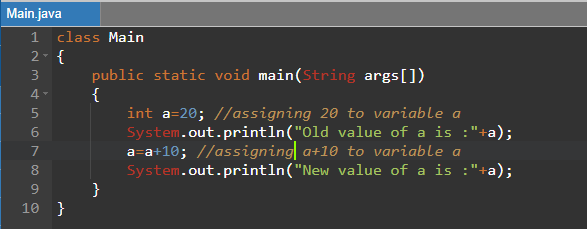
****

**Output:**

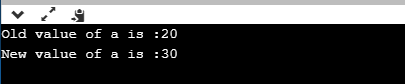
****

* **Assignment operator :**
* Assignment operator is used to assign any value to any variable.
* It is denoted by “=”.
* Eg : int a = 5,float b=7.5.
* Example:

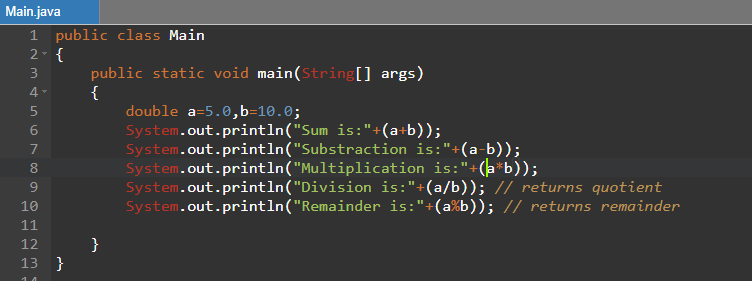
Code:



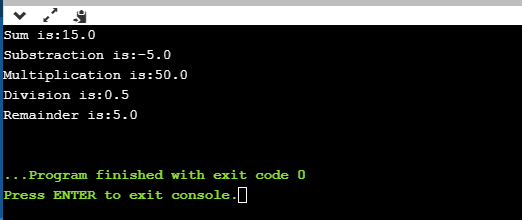
Output:



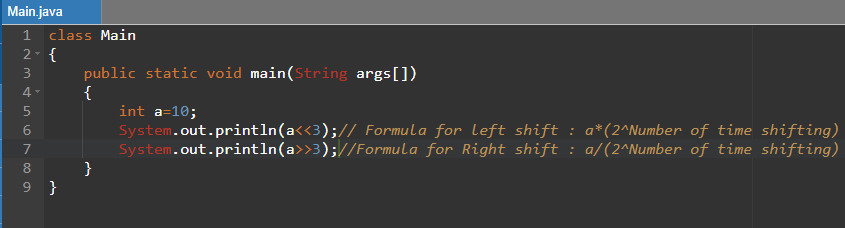
* **Arithmetic operator :**
* Arithmetic operator is used to perform any arithmetic operation on the variables.
* There are 5 arithmetic operators Which are given in following according to precedence.
* \*, / , %
* +, -
* Example:



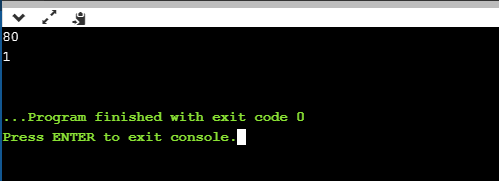
Output:



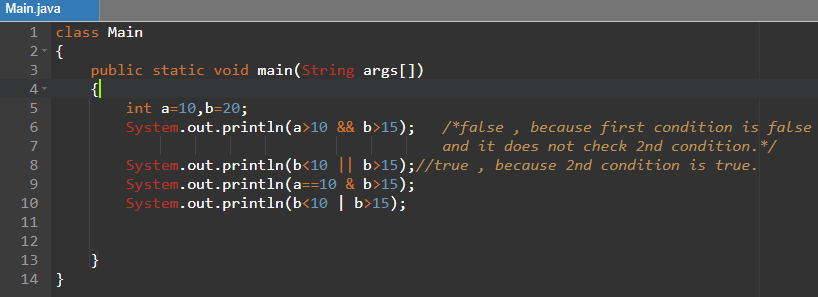
* **Relational operator:**
* Relational operator is used to compare two variables.
* Relational operators are >(Less then) ,<(greater than) ,>=(Greater then or equal to) ,<=(Less then or equal to) ,==(Equal),!=(Not Equal to).
* Examples: Examples of this operator is covered in Example of logical and Bitwise operator.
* **Shift operator**:
* Shift operators in java are of two types. That is <<(left shift operator) and >>(Right shift operator).
* Left shift operator is used to shift all the values to the left side according to specified number.
* Right shift operator is used to shift all the values to the Right side according to specified number.
* Example:



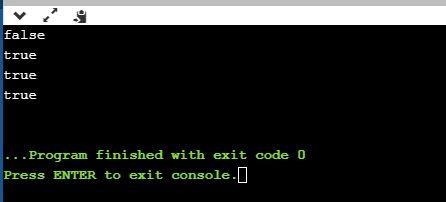
Output:



* **Logical operato**r:
* Logical operator are of two types. That is logical AND(&&) and logical OR(||).
* Logical AND(&&) applied between two conditions. If two conditions are true then result is true. It doesn’t check the second condition.
* Logical OR(||) applied between two conditions. If any one of the conditions is true then result is true. It doesn’t check the second condition.
* **Bitwise operator:**
* Bitwise operator check both the conditions.
* Bitwise operators are of two types . that is Bitwise AND(&) ,Bitwise OR(|) and Bitwise Ex-OR(^).
* Examples:

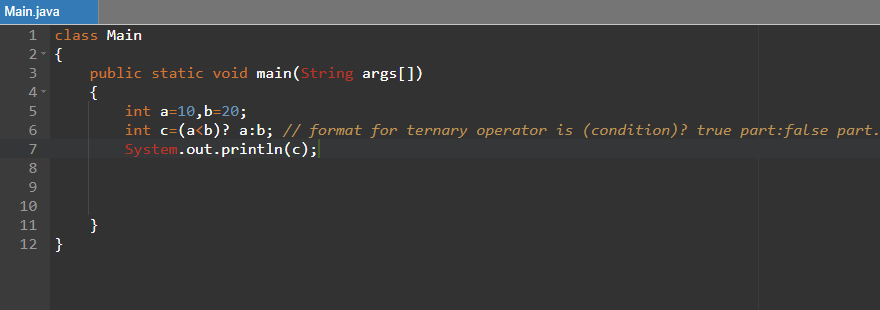
****

**Output:**

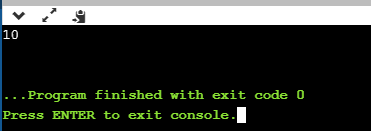
****

* **Ternary operator:**
* Ternary operator is used as a replacement of “**if else”** block.
* It using three operands in an expression.

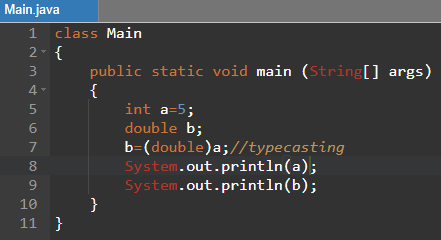
Example:



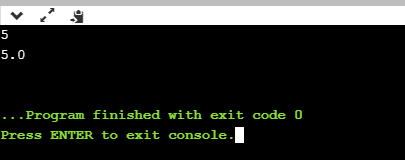
**Output:**



* **Typecasting:** converting variable’s data type is called as type casting.



**Output:**



***Control Statements***

* Following are the control statements in java
* comments
* If else
* break
* Switch
* While
* do while
* for loop
* continue
* **comments:**
* comments in java are mainly used to indicate that which line of code is doing what process .
* There are two types of comment in java

1. Single line comment.
2. Multi line comment.

* Single line comment- whatever in java written after **“//”** is called as single line comment.
* Multi line comment-for comments in multiline we have to use **“/\*”** at start of line and **“\*/”** at the end of line.

Whatever is written between **“/\* \*/ “**is a multiline comment.

* **“if else” control statement.**
* Syntax for if else is as follows.

If(condition)

{

//if condition true then this part is going to be perform.

}

else

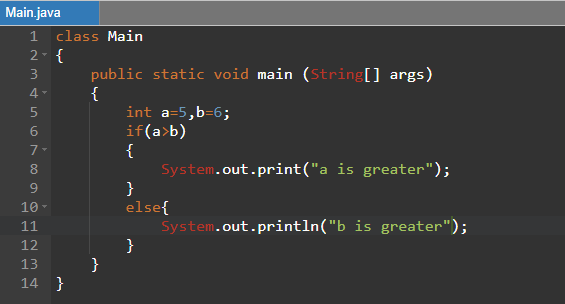
{

//if condition is not true then this part is going to perform.

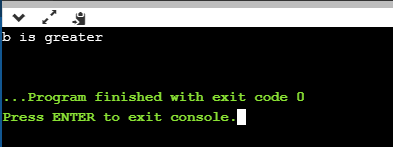
**}**

* **“if else”**  control statements are of three types:

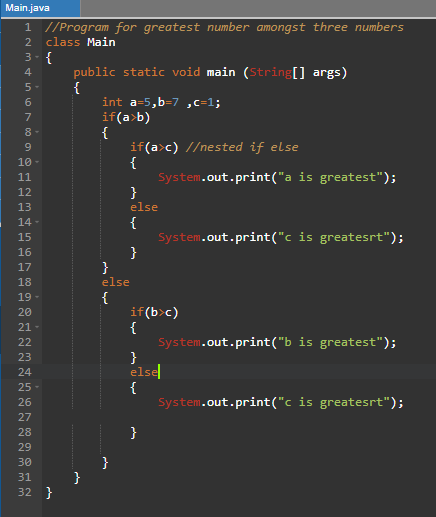
1. If else.
2. Nested if else – “if else” within” if” or “ else” part is called nested if else.
3. If else ladder. -

* Example 1:

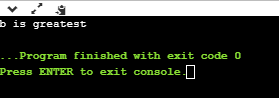
**Output:**

****

* **Example2 (nested if else):**

****

**Output:**

****

* **Example 3: (if else Ladder)convert any number into Word. Eg: input = “1”**

**output =”one”.**

**Code:**

**class Main**

**{**

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

**{**

**int a=5;**

**if(a==0)**

**{**

**System.out.print("Number is Zero");**

**}**

**else**

**{**

**if(a==1)**

**{**

**System.out.print("Number is one");**

**}**

**else**

**{**

**if(a==2)**

**{**

**System.out.print("Number is Two");**

**}**

**else**

**{**

**if(a==3)**

**{**

**System.out.print("Number is Three");**

**}**

**else**

**{**

**if(a==4)**

**{**

**System.out.print("Number is four");**

**}**

**else**

**{**

**if(a==5)**

**{**

**System.out.print("Number is Five");**

**}**

**else**

**{**

**if(a==6)**

**{**

**System.out.print("Number is Six");**

**}**

**else**

**{**

**if(a==7)**

**{**

**System.out.print("Number is Seven");**

**}**

**else**

**{**

**if(a==8)**

**{**

**System.out.print("Number is Eight");**

**}**

**else**

**{**

**if(a==9)**

**{**

**System.out.print("Number is Nine");**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

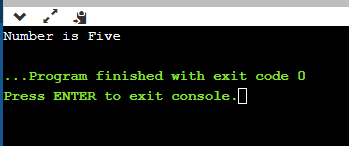
**}**

**}**

**}**

**}**

**Output:**

****

* **break:**
* break control statement is used to terminate the program.
* Example of break control statement is given in code of switch case.
* **Switch Case:**
* we can use switch case rather than using “if else” ladder.
* Using switch case is easy than if else ladder.
* Code of switch case is small.
* Variable passed in switch case can be integer, character and String but we can not pass float number.
* Syntax:

switch( variable)

{

case 1 : //code

break; //if condition is right than break executes and program terminates.

case 2: //code

break;

case n : //code

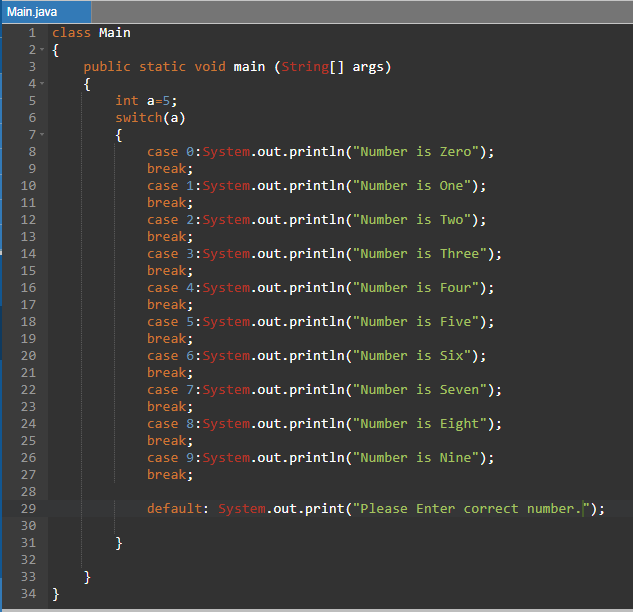
break;

default: //code;

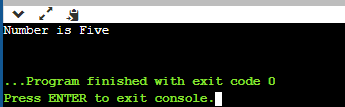
}

* **Example:**

Let’ s take same example which we have taken in if else ladder



Output:



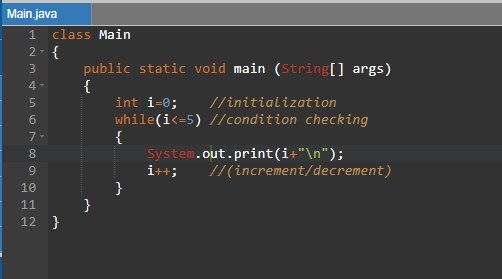
* **while loop:**
* while loop is use to perform some operation repetition. In simple words it is used for iteration.
* Syntax:

while(condition)

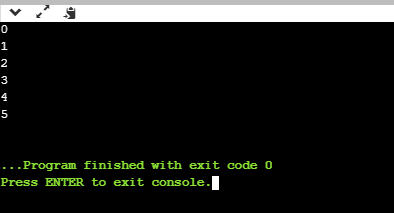
{ //code;

}

* Example:



Output:



* **do-while loop:**
* do while loop is also used for iteration.
* The main difference between “while loop” and “do while loop” is that do while loop is performed at least once.
* Syntax:

do

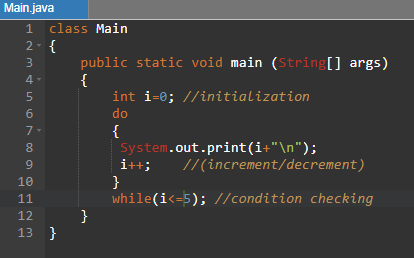
{

//code

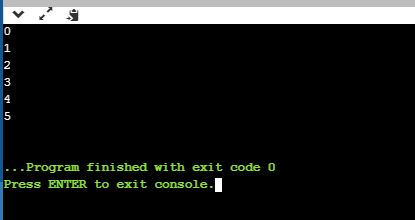
}

while(condition);

Example:



Output:



* **for loop:**
* for loop is also used for iteration.
* Syntax:

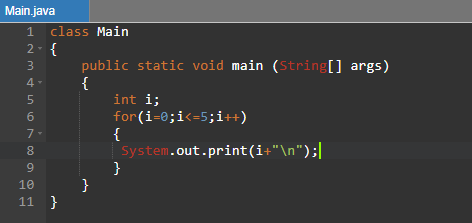
for(initialization ; condition ;increment/decrement)

{

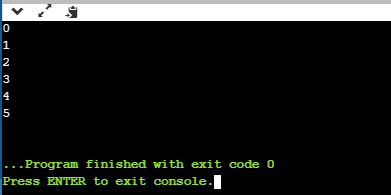
//code

}

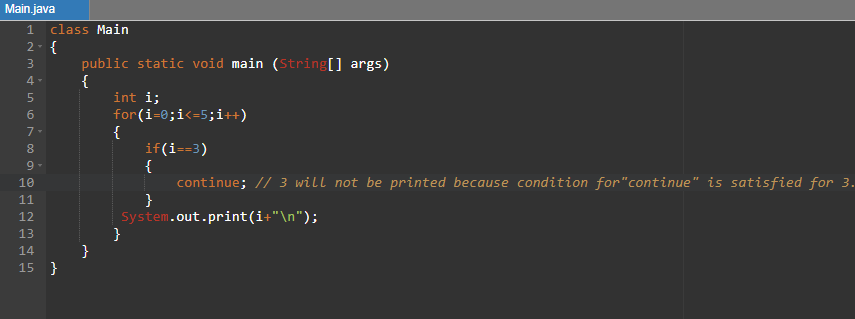
Example:

****

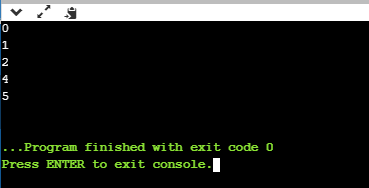
**Output:**

****

* **continue:**
* **continue statement is used between iterations.**
* **when we write “continue;” inside for, while or do while loop then it does not perform that statement or code where condition satisfies for continue.**
* **Example:**



**Output:**

****

***User Input in java***

* Using **“Scanner”** class which is available in **java.util** package we can take input from user.
* Step1 : Import **java.util.Scanner;**
* Step2 : create object of Scanner class.

**Scanner s= new Scanner(System.in);**

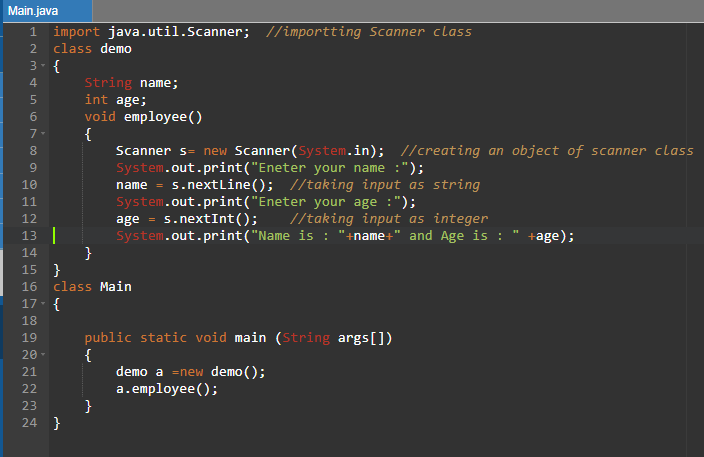
Where s=object.

(System.in)=argument.

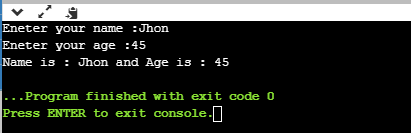
* Step3: assign this object to a variable.

For different data types there are different methods which are given as follows.

* Int nextInt();
* Deouble nextDouble();
* float nexFloat();
* **String nextLine();**
* **byte nextByte();**
* long nextLong();
* boolean nextBoolean();
* short nextShort();
* bigdecimal nextBigdecimal();
* biginteger nextBiginteger();
* Examples:



Output:



***Methods:***

* A method is a block of code which is used to perform some operations in java.
* It provides reusability in code.
* Same Method can be used man times in java.
* A method is always declared inside the class.
* Following is the way for declaring method in java.

public int test(int a,int b)

{

//code;

}

Where,

Public-Access modifier (it can be **public, private, protected or default**.), int – return type, test-class name, **int a** and **int b** – Parameters.

* Naming conventions for method:

1. The name of method must be start with lowercase letter.
2. Example: sum(),test().
3. If there is a method name which consist of two words then we can write it as following example:

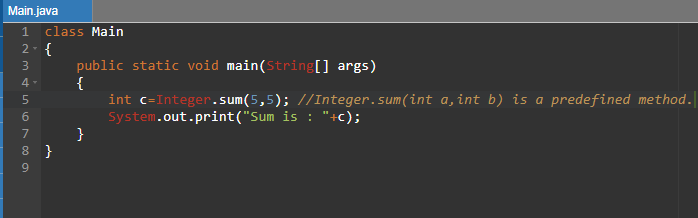
testOne() , testTwo()

* There are mainly two types of methods.

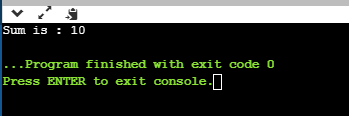
1. Pre-defined method.
2. User-defined method.

* Pre-defined method- The functions are already present in java. You just need to call this function into the program.

Example: Addition of two numbers using pre defined function.

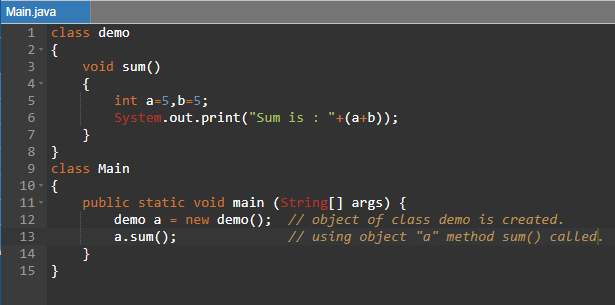


Output:

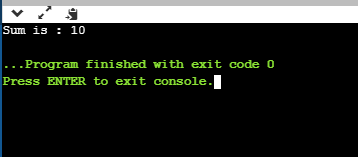


* User-defined Method : These packages are defined by users.

Examples:



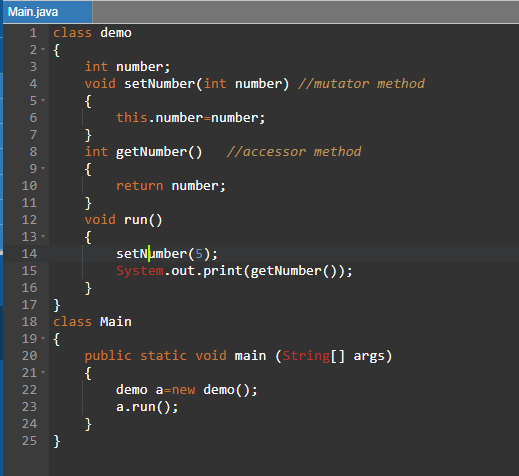
Output:



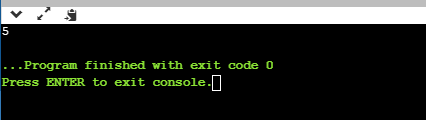
* There are mainly two types of user defined methods:

1. Accessor Method : this method is used to read data from a variable.
2. Mutator method : this method is used to put data in a variable.

* Example:



Output:



***Constructor***

* Constructor in java is similar to method.
* Difference between method and constructor is that the name of constructor is same as that of the name of a class in which constructor is defined.
* Syntax:

class demo

{

Public demo()

{

//code

}

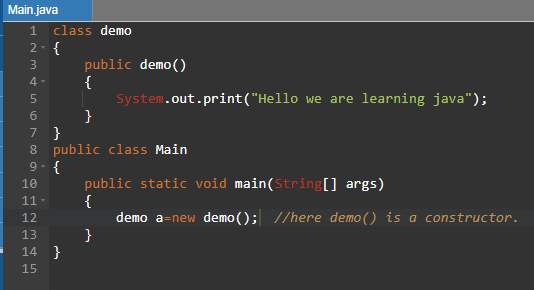
}

* There is no return type associated with constructor.
* It is called when instance or object of a class is created.
* A java constructor can not be static, abstract, final and synchronized.
* There are two types of constructors.

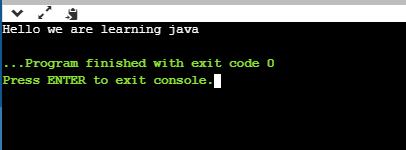
1. Default constructor.(don’t have parameters)
2. Parameterized constructor.

* Default constructor :
* Default constructor is a constructor which does not have any parameters.
* It is used to set value of instance variable as 0 or null.

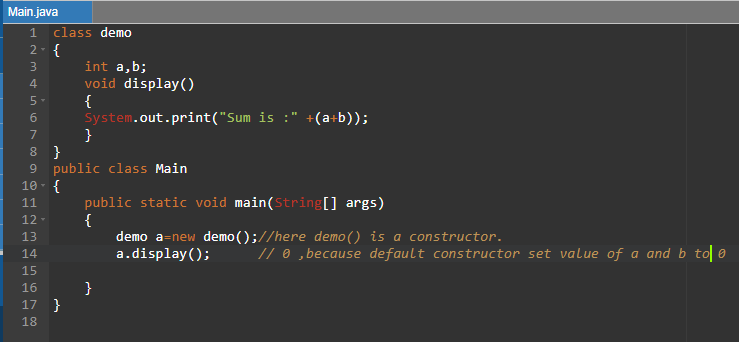
Example 1:



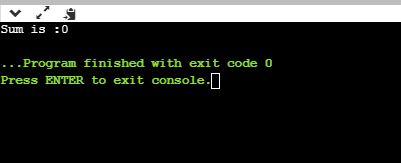
Output:



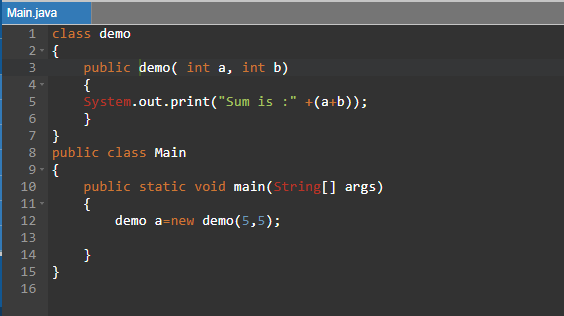
Example 2:



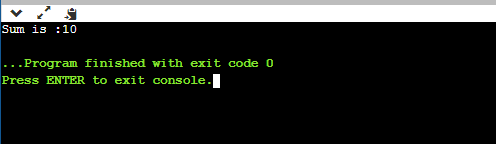
Output:



* Parameterized Constructor:
* Parameterized constructor are constructors which are having parameters .
* Examples:



Output:

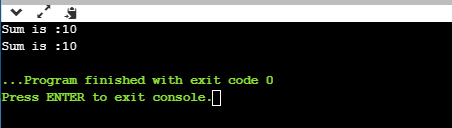


* Copy Constructor :
* Copy constructors are used to copy the value of one object to another.
* There are three ways for this.

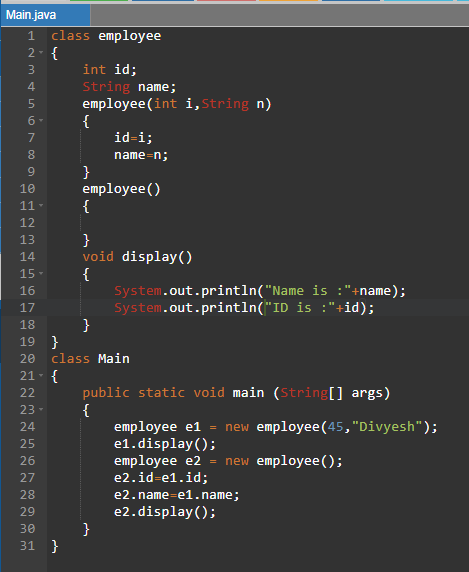
1. Using constructor



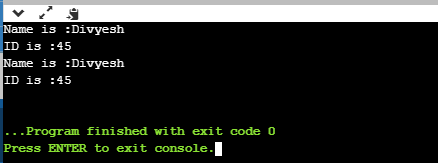
Output:



* Directly copy from object.



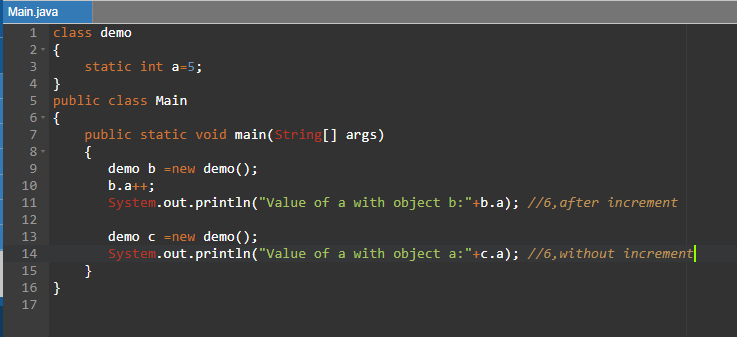
Output:



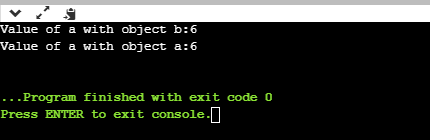
***static Keyword***

* “static” keyword in java is mainly used for memory management.
* static keyword can be used with variable , method ,block and nested class.
* static variable :
* static variable in java is used for memory management.
* Once the value assigned to the static variable , It does not change until you change it.

Example:

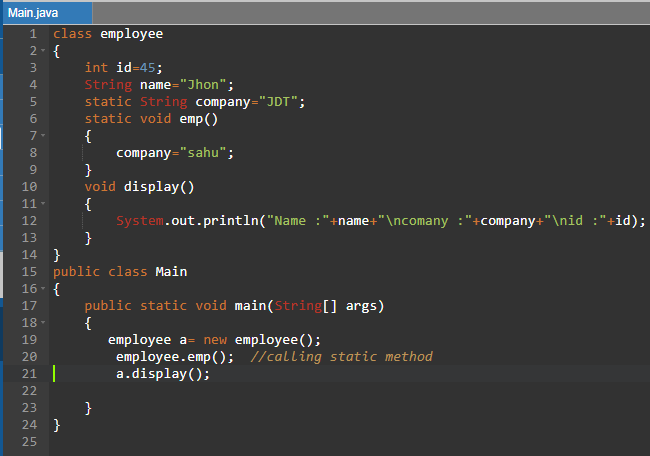


Output:

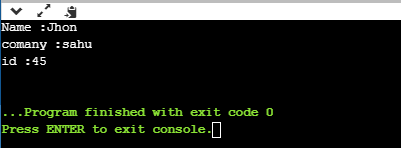


* In above example , a variable “a” is static and it is not associated with an object.
* It creates different memory space.
* Once we increment the value of “a”.
* It is set to 6 till we do not change it.
* static Method:
* The methods with static keyword is called as static method.
* It can be used without creating an object of a class.
* It belongs to a class rather than object.

Example:



Output:



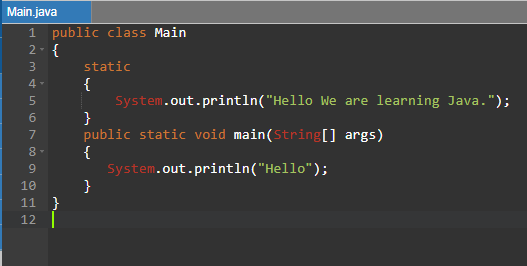
* Restriction for static methods:

1 .non static members can not be used in static method.

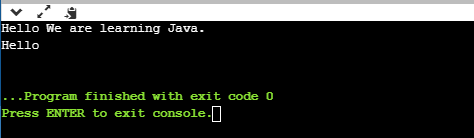
2.super() and this keyword can not be used with static method.

* Static block:
  + static blocks are defined before the main method.
  + It is performed before main method.

Example:



Output:

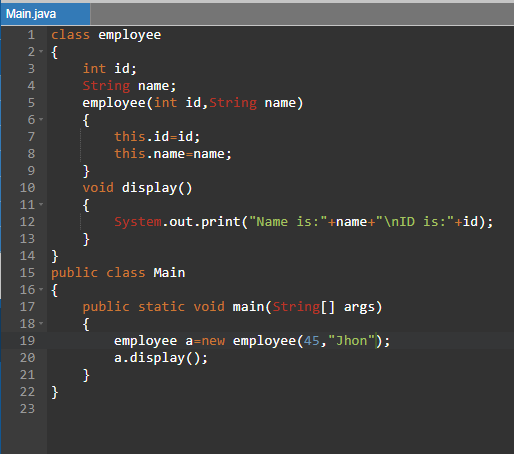


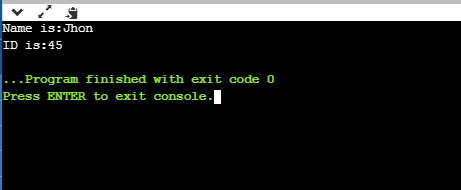
***this Keyword***

* this keyword in java is mainly used for following :

1. In class, if the names of a instance variables is same as that of name of parameters in method then this keyword can be used.

Example:

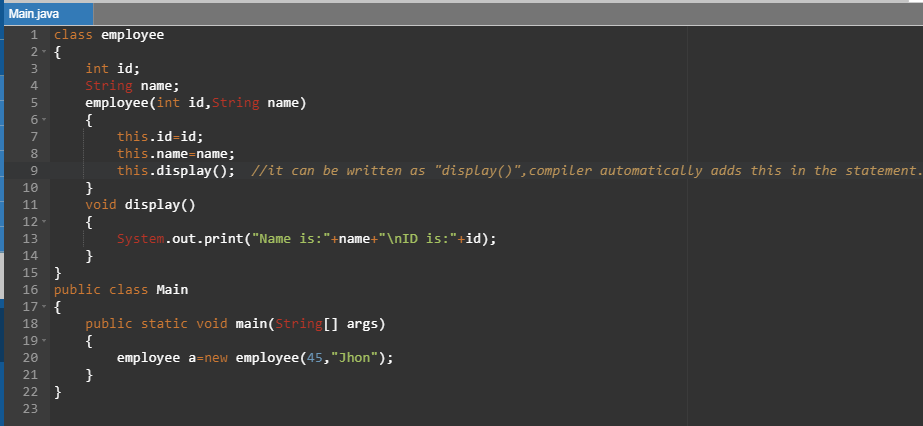


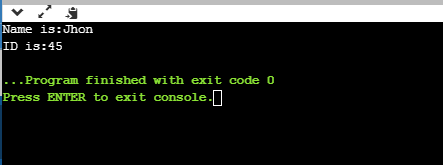
: 

1. To invoke, current class method.

If we do not write this keyword than compiler automatically adds this keyword to the statement.

Example:

 Output:

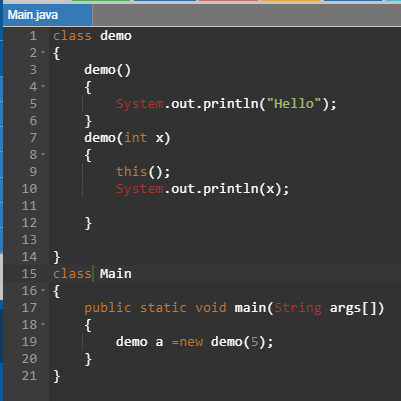


1. To invoke current class constructor.

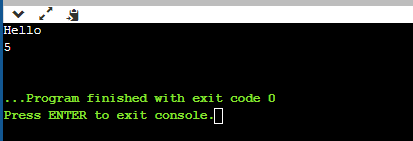
To use, the current class constructor in java we use this() method.

* Calling of this() function must be first statement in a constructor.

Example:



Output:

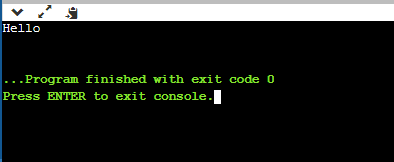


1. this keyword can be passed as an argument in Method.

Example:

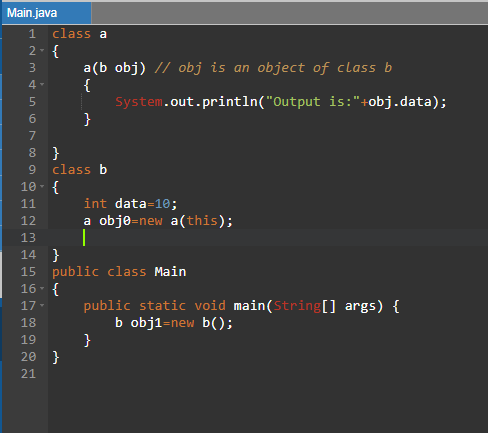


Output:

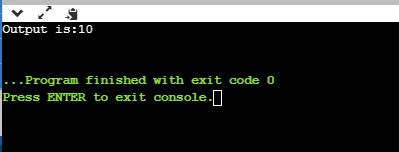


1. “this” can be passed as an argument in a constructor.

Example:



Output:



***Inheritance***

* Inheritance in java is used for inheriting all the properties of previous object to new one.
* Because of inheritance we can use all the properties and methods of parent class in child class.
* Inheritance represents IS-A relationship which is also known as parent-child relationship.
* There are following types of Inheritance.

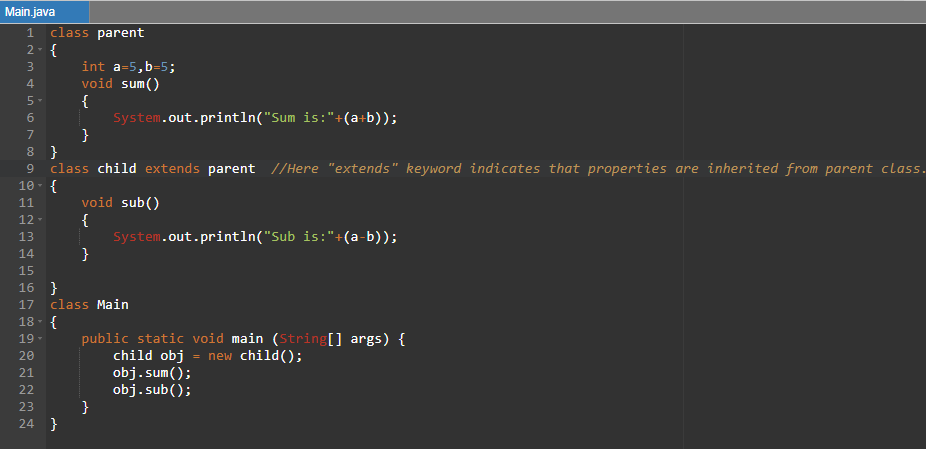
1. Single level Inheritance:

Parent class

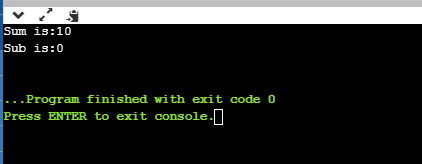
Child class

In single level inheritance all the properties and methods of parent class is inherited in child class.

Example:



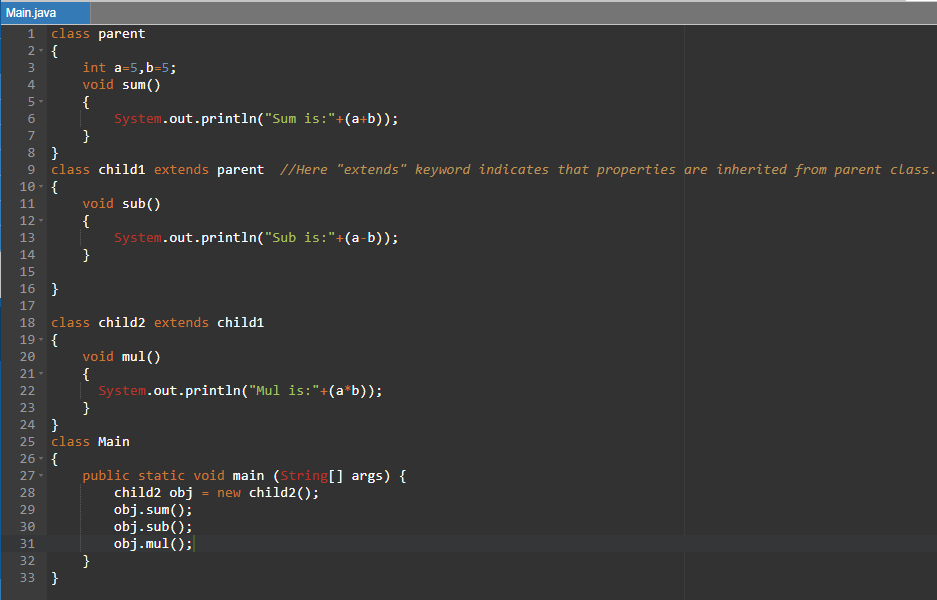
Output:



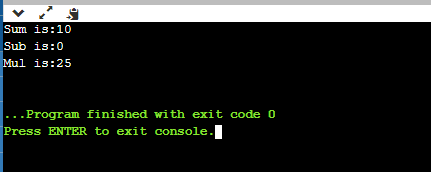
1. Multilevel Inheritance.

Whenever there is a chain of inheritance than it is called as multilevel inheritance.

Example:



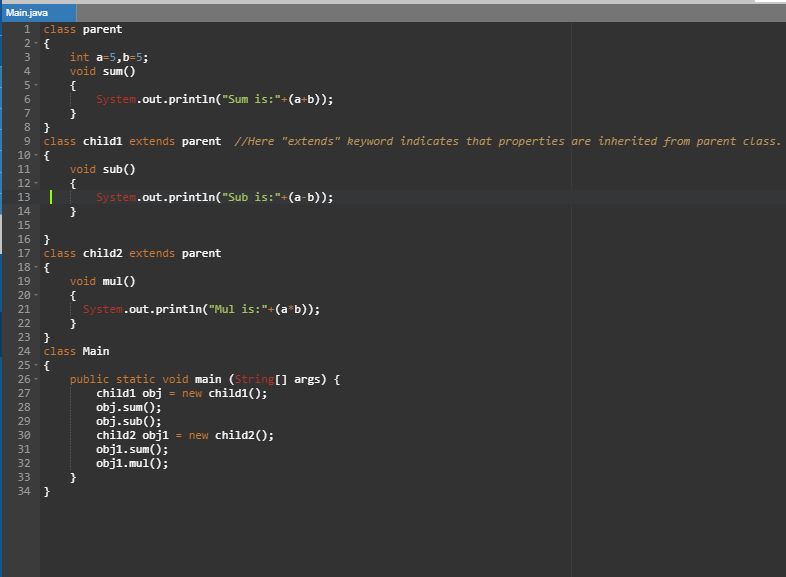
Output:



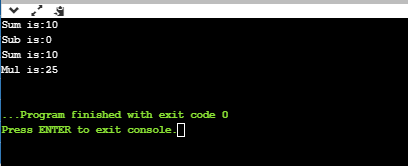
1. Hierarchal Inheritance:

When one base class or parent class is inherited into two different sub class or child class than it is called as Hierarchal Inheritance

Example:



Output:



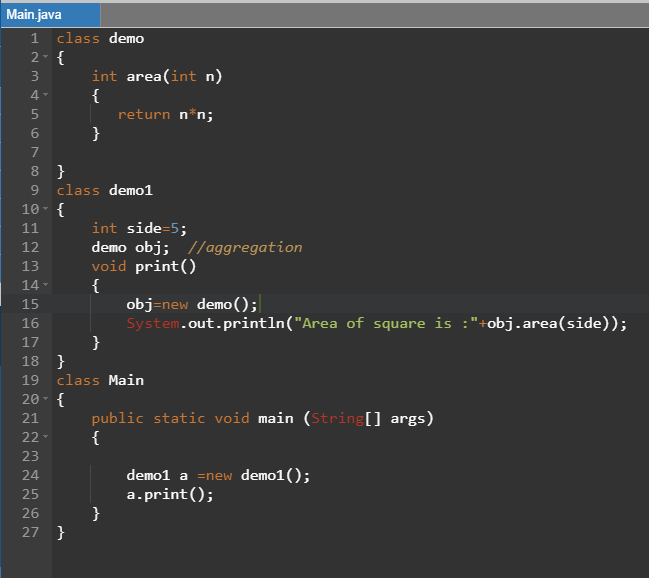
1. Multiple inheritance:

Whenever you import two classes in one class then it is called as Multiple inheritance.

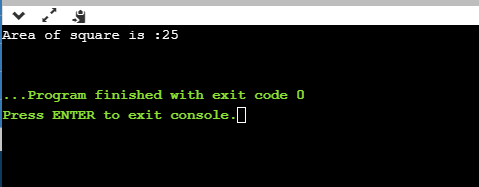
Multiple inheritance is not possible in java. Because, if there is a same name of method in different classes and if you extend it in same class them compiler will get confused that which method is going to run Therefore it gives run time error.

* Also, There is something called as HAS-A relationship in java.
* This “HAS-A” relationship is also called as “aggregation” in java.
* It is used for reusability of a code.

Example:



Output:



***Polymorphism***

* Poly – many and morph- forms.
* Polymorphism means an object can have a different forms or behavior.
* Example : Mother is a daughter ,sister and an employee at a same time.
* There are different ways of polymorphism in java.

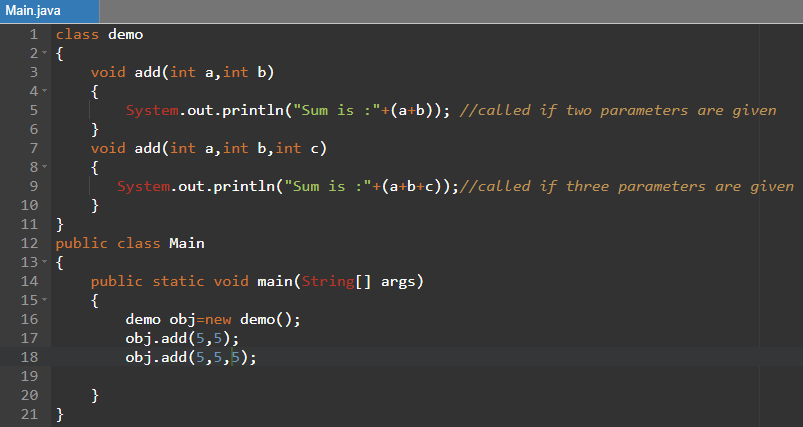
1. **Method Overloading:**

Whenever you have to perform different operations under the same name of a method then it is called as Method overloading.

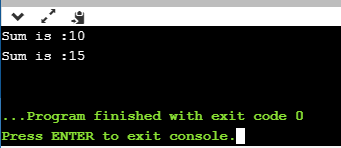
There are two ways of method overloading.

1. By changing the number of arguments in method.

Example:

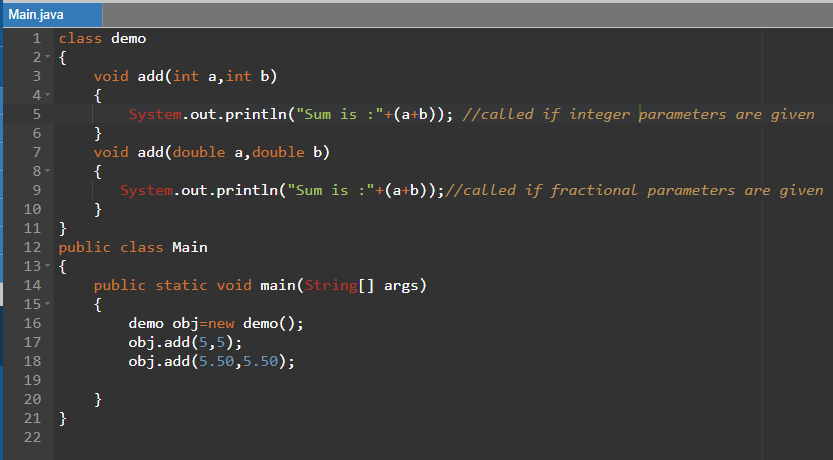


Output:

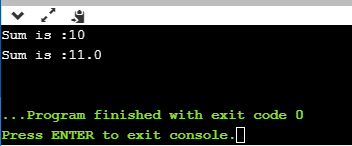


1. By changing the data type of an argument.

Example:

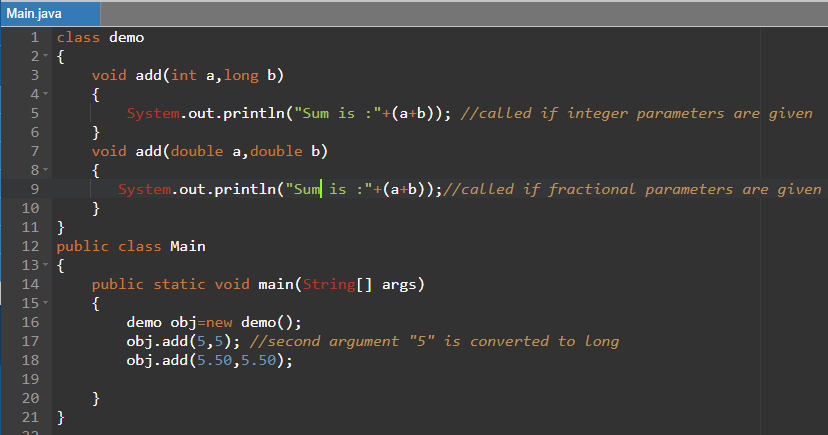


Output:

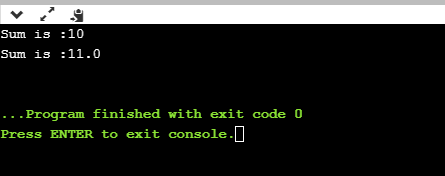


* Method overloading with type promotion : In method overloading , byte can be converted to short, int, double, long, float also short an be converted into int, double, long, float and char can be converted into int, double, long, float.

Example:



Output:



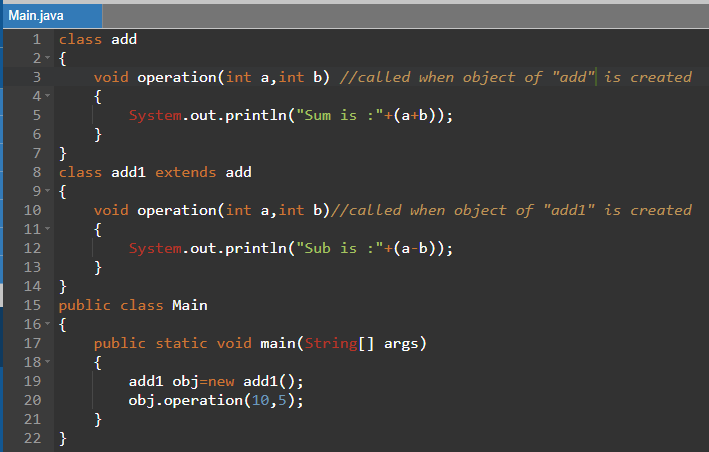
1. **Method Overriding:**

If a subclass is having same method name as of parent class method name than it is called as Method overriding.

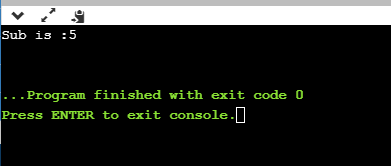
It is also known as java runtime polymorphism.

Use of java overriding is that providing specific implementation to a method which is already present into the super class.

Example:



Output:



Rules for java overriding:

* The name of a method in super class and sub class must be same.
* Number of parameters must be same as that of number of parameters in method of super class.
* There must be IS-A relationship.

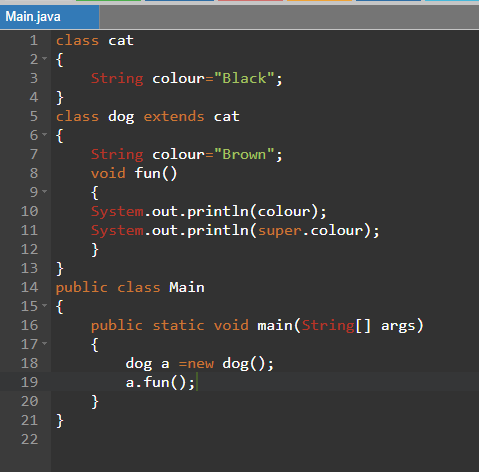
1. **super Keyword:**

If there is same name of variable , Method or constructor in super class than super keyword in java is used to access that thing in sub class.

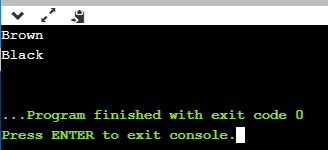
Super keyword is used for following:

* To access same name of variable present in super class in sub class.

Example:



Output:

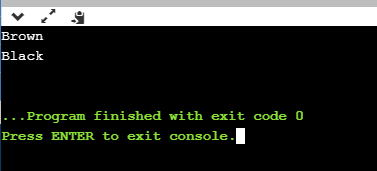


* To access same name of method present in super class in sub class.

Example:

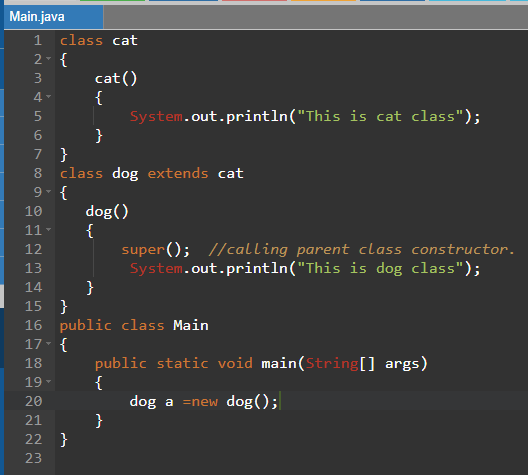


Output:

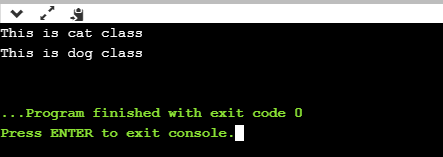


* To access same name of constructor present in super class in sub class.

Example:



Output:



If you do not call “super()” then compiler implicitly adds super() into constructor.

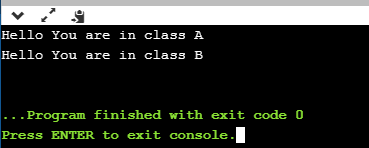
1. **Covariant return type:**

Before java 5, It was not possible to override a method by changing a return type of a method. But now using covariant return type we can change the return type of a class.

Example:



Output:



Here you can see that the return type of method msg() in class A is A, But the return type of method msg() in B is B.

1. **final keyword:**

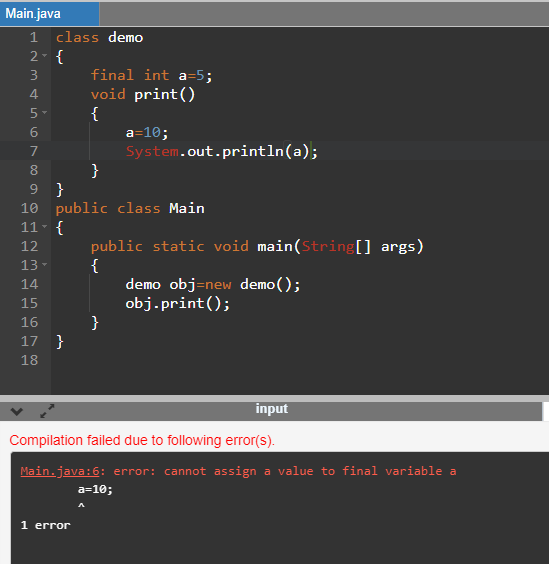
final keyword is used when the value of any variable, method or class is going to be fixed.

final keyword is used with following:

* variable
* method
* clsass
* final keyword with variable:

If you make variable final than you can not change the value of variable.

Example:

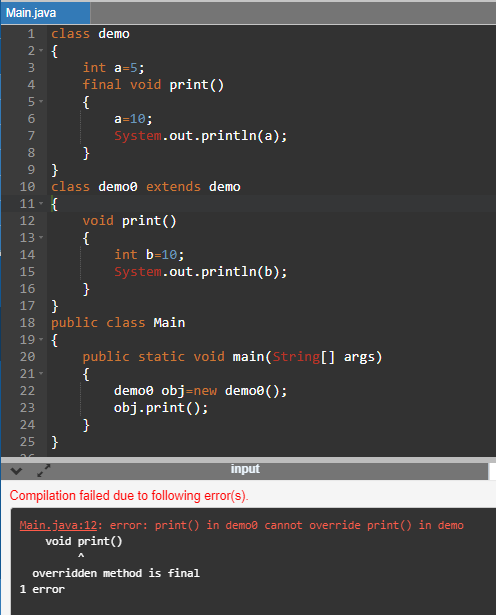


Since, we can not change final variable therefore we got an error in output.

* Final method :

If we make any method final than we can not override that method.

Example:

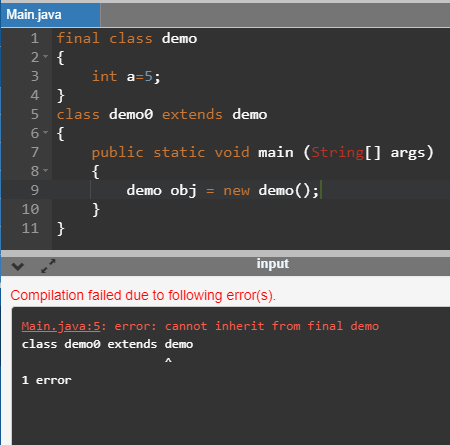


Here in above example, if we override a final method than we got compile time error.

* Final class:

final class can not be inherited in any class.

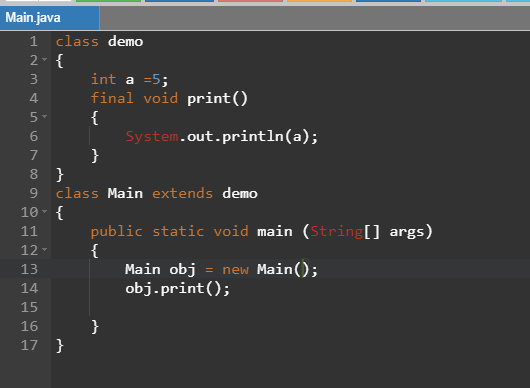
Example:



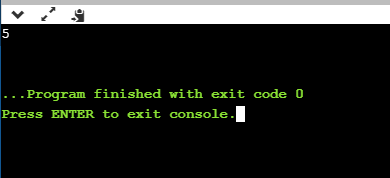
Here, in above example class demo is final therefore it can not be extended.

* Final method can be inherited.

Example:



Output:

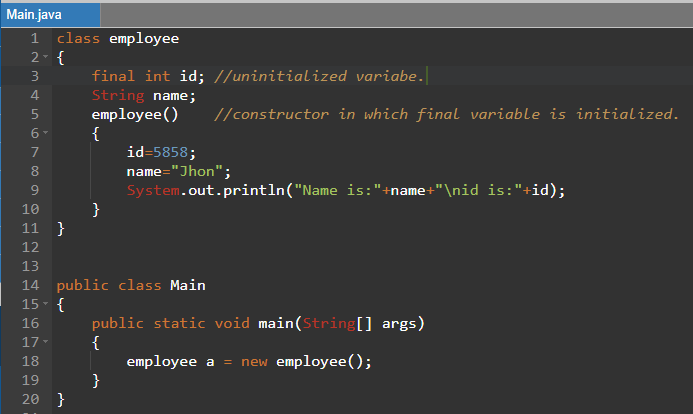


* What is uninitialized final variable ?

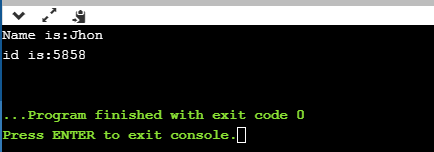
Uninitialized final variable is that variables which is not initialized but it is final.

These variables can be defined inside constructor only.

Example:



Output:

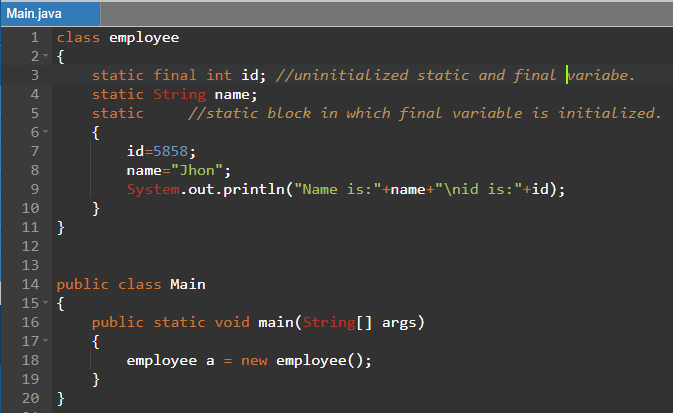


* Static final uninitialized variable:

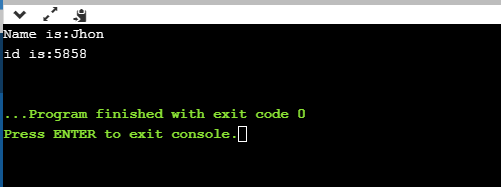
Static final uninitialized variable is variable with static and final keyword.

It can be initialized within static block only.

Example:



Output:



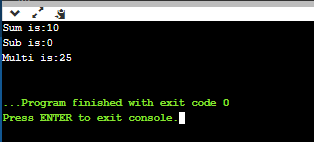
* **Upcasting in java:**

Here , object of base class is created but the reference is given to the sub class.

Example:



Output:



***Abstraction***

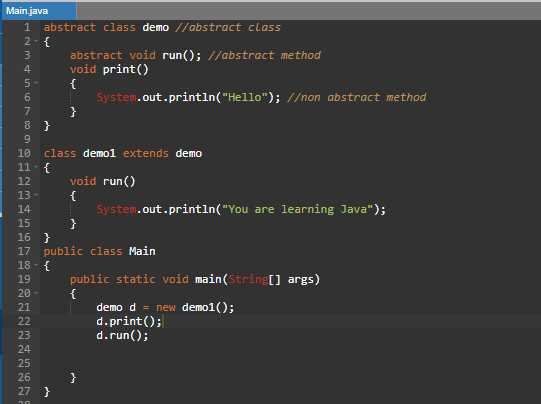
* Abstraction in java is used to hide data or an information from user end.
* Abstraction can be done with the help of “abstract class” and interface.
* **abstract class:**

abstract class is a class which is defined with keyword abstract.

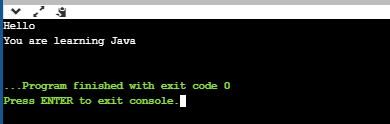
Point s to be remember:

1. abstract class must be written with abstract keyword.
2. In abstract class abstract methods and non abstract methods both can be written.
3. Body of abstract method can not be defined inside the abstract class.
4. It can have final methods.
5. It can have constructors and static methods also.

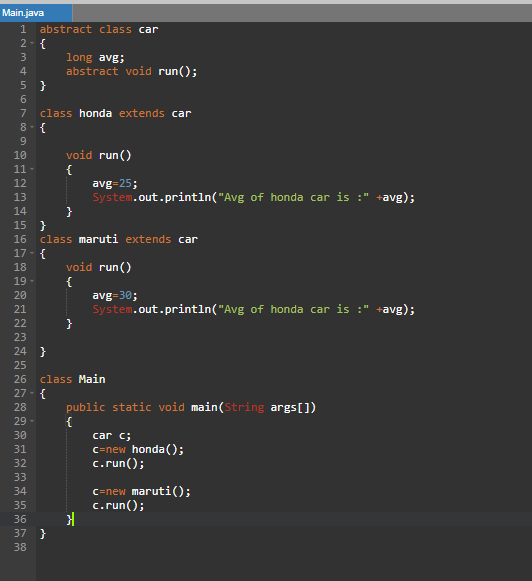
Example:



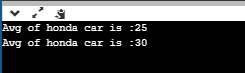
Output:



Example2:



Output:



* **Java interface:**

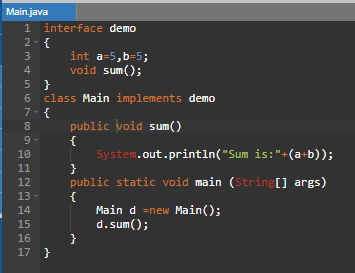
Java interface is used to achieve abstraction in java.

It provides 100% abstraction where as abstract class provides 0 to 100% abstraction.

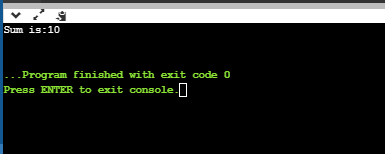
Java interface can have only abstract, default and static methods.

* Java interface is used to achieve abstraction.
* We can implement multiple inheritance using java interface.

Example:

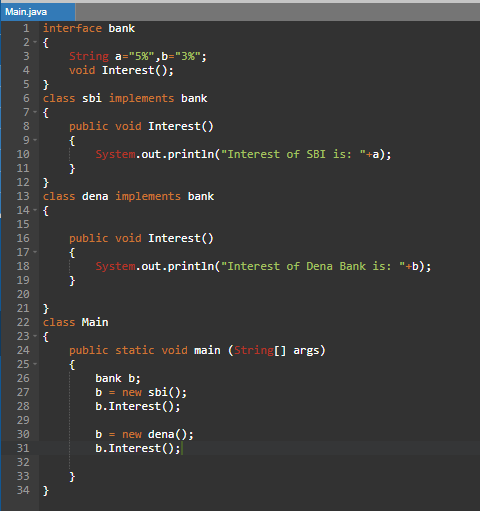


Output:

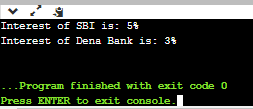


Here, in above example compiler adds public , static , final to a and b and public , abstract to method.

Example2:



Output:

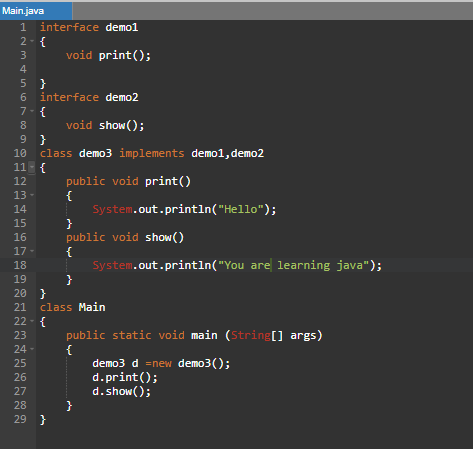


* Multiple Inheritance in java using interface:

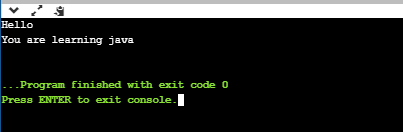
When two or more objects are inherited in other class than it is called as multiple inheritance in java.

Multiple inheritance in java is not possible using class due to ambiguity but it is possible with the help of interface.

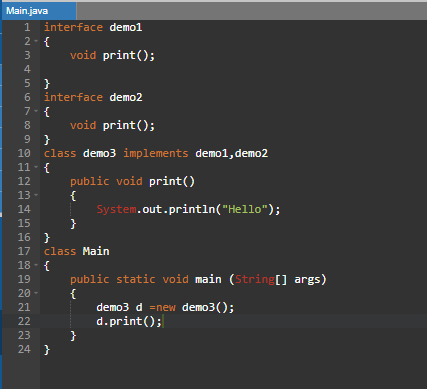
Example:



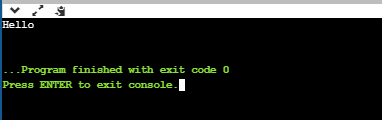
Output:



Example 2:



Output:



Here , in above example both interfaces demo1 and demo2 both having same method print().

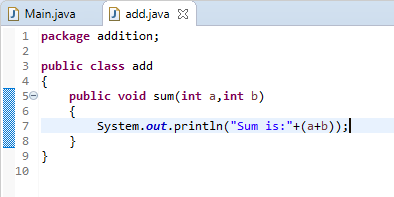
This print() method is declared in demo3 therefore there is no ambiguity with Interface and we can perform multiple inheritance.

***Package***

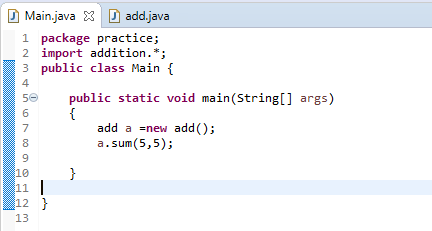
* Java Package is used for classification for class and interface.
* It provides access protection.
* Also it removes naming collisions.
* **“package”** keyword is used to create a package.

**Example:**

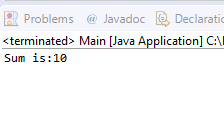
**Package:**

****

**Program in which package is imported:**

****

**Output:**

****

Here in above example, “package addition” is a package name in which class “add” is defined. This package is imported into the Main program as “import addition.\*”

Where \* indicates that all classes and methods are imported and so that we can use it in a program.

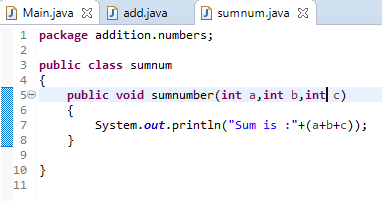
* There are three ways to import any package.

1. Using “import package\_name.\*”-Importing all the classes.
2. Using “import package\_name.class.name”- Importing particular class.
3. Using package at the time of object creation as “package\_name.class\_name a=new package\_name.class\_name()”.

* You can create sub packages also in java. That is package within a package is called as a sub package.
* If you import java main package then you can not use sub package classes.
* Also is you import java sub package only then you can not use java main package.

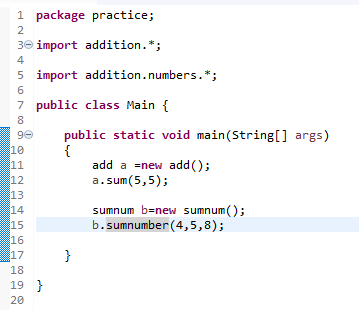
Example:

Lets take above example of “package addition” and create a sub package in it.

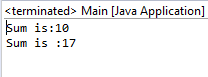


Here in above example as you can see that in statement “package addition.numbers” numbers is a sub package of addition package.

We can import this package in our main program as shown below.



Output:



In similar way we can create n number of packages an sum packages in our program.

* **Access Modifiers:**

Access modifiers in java are used to specify accessibility of a variable, class, constructor or any method.

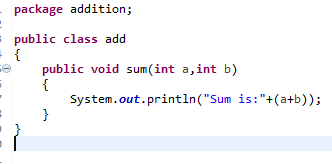
* There are mainly two types of Modifiers in java Access modifiers and Non- Access modifiers.
* There are mainly four types of access modifiers in java

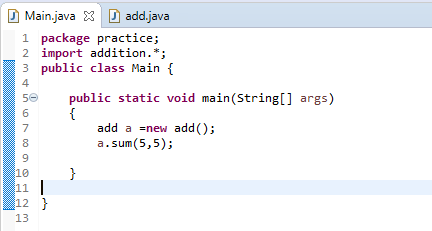
1. public
2. private
3. default
4. protected
5. public access modifier:

“public” access modifier can be used everywhere.

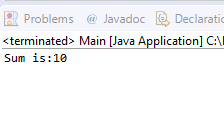
That is it can be used inside class , outside a class , within a package and outside a package.

Example:





Output:

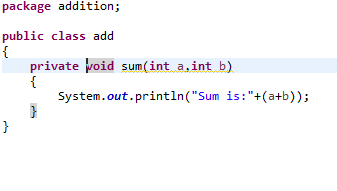


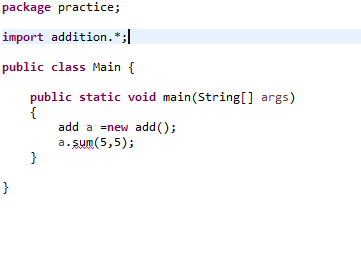
Here in above example , You can see that in our package addition is class add is **“public”**

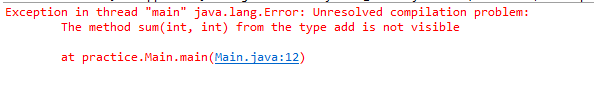
So that it can be used outside the package also.

1. private access modifiers:

private modifier can be used only within a class.



Output:

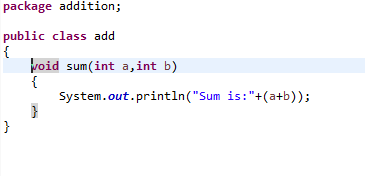


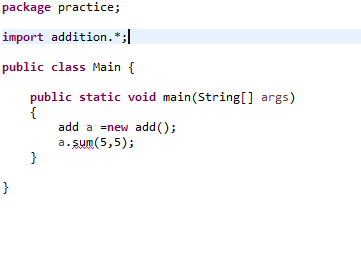
Here in above example , in the package addition the method add is “private” therefore in our main file it can not be used and it gives an error.

1. default access modifier:

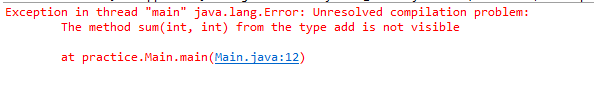
default modifier can be used within a class and within a package only.

Example:





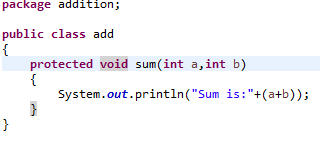
Output:

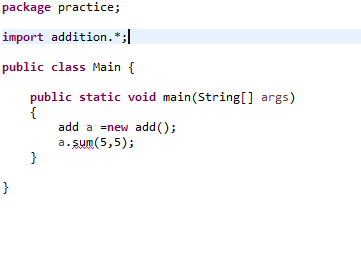


Here in above example , in the package addition the method add is “default” therefore in our main file it can not be used and it gives an error.

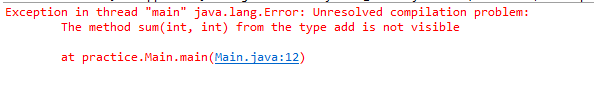
1. protected access modifier:

protected modifier can be used anywhere accept outside the package.





Output:



Here in above example , in the package addition the method add is “protected” therefore in our main file it can not be used and it gives an error.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only** | **outside package** |
| **Private** | Yes | No | No | No |
| **Default** | Yes | Yes | No | No |
| **Protected** | Yes | Yes | Yes | No |
| **Public** | Yes | Yes | Yes | Yes |

***Encapsulation***

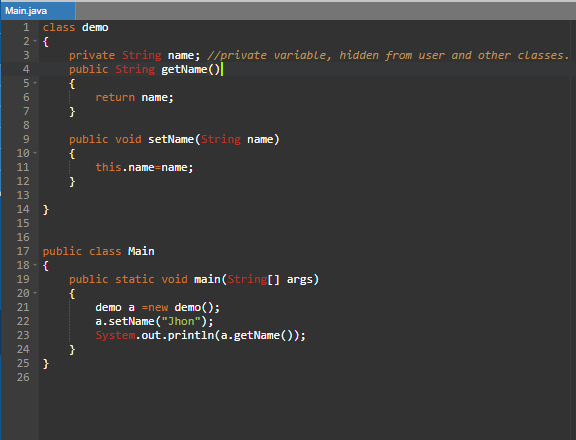
* Encapsulation in java is used to bind a similar kind of class under one program.
* Consider an example capsule which consist of many tiny medicines.

Here , Capsule – Package.

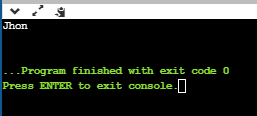
Tiny Medicines – classes

* Data which is should be hidden from user are encapsulated.
* Here are some example of Encapsulation in java using java getter and setter methods.

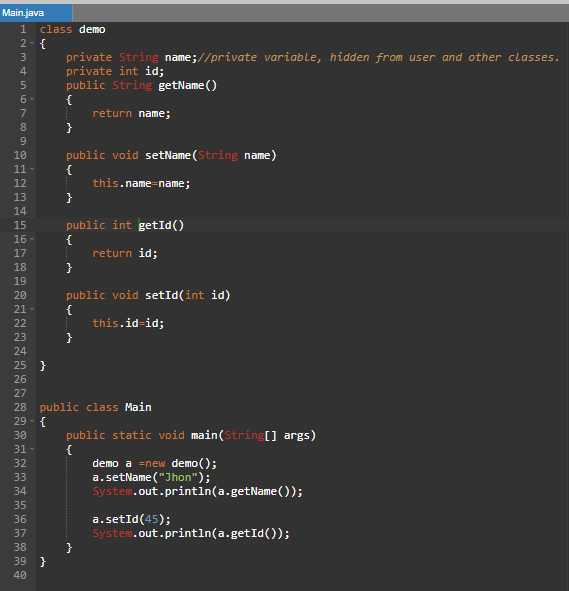
Example:



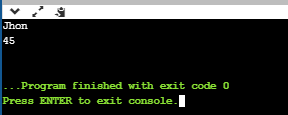
Output:



Example2:



Output:



***Strings***

* Strings in java is an array of a characters
* Here is an Example of a character array : Char[] a={‘a’ ,’b’, ‘c’ , ‘d’}
* String constants are stored in a special memory which is called as “String constant pool”

There are two methods for creation of Strings:

1. Using String literal:

Example:

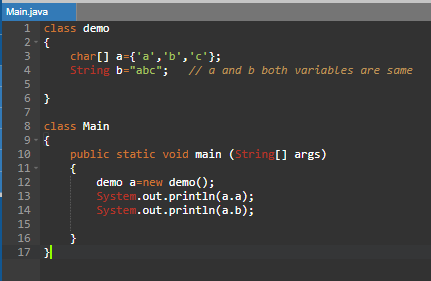
**String** a= “Jhon”;

* If You create any other object which also contains the same string than it does not create other object it just refers to the same string in the String constant pool.

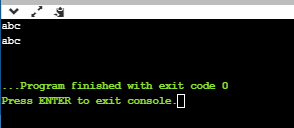
1. Using new keyword:

String a =new String(“Jhon”);

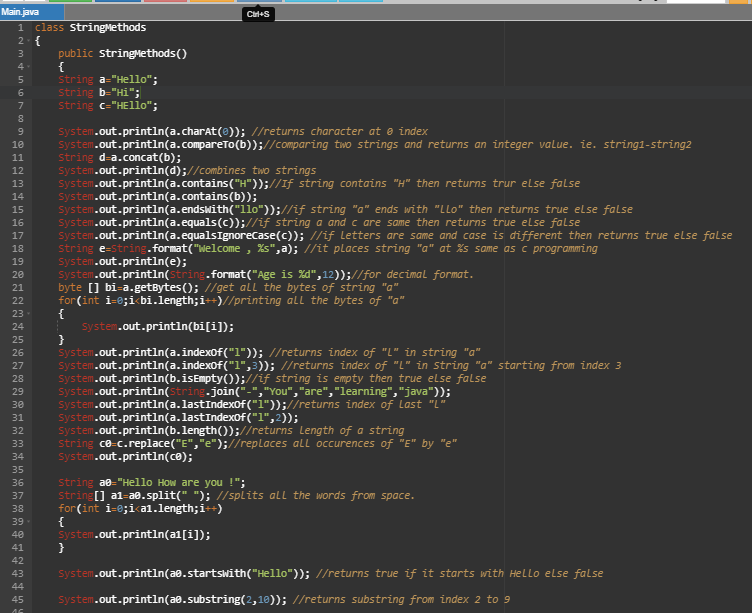
Example:

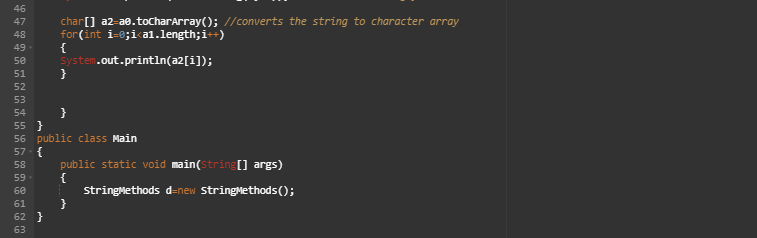


Output:

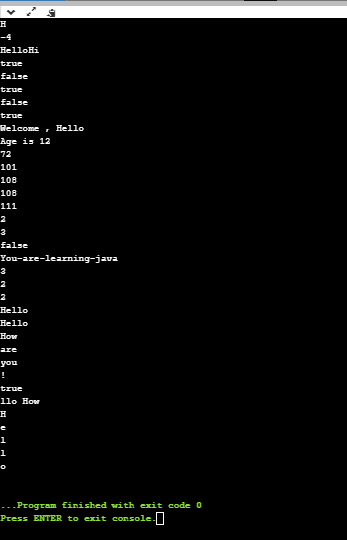


* There are some methods which are used in Strings





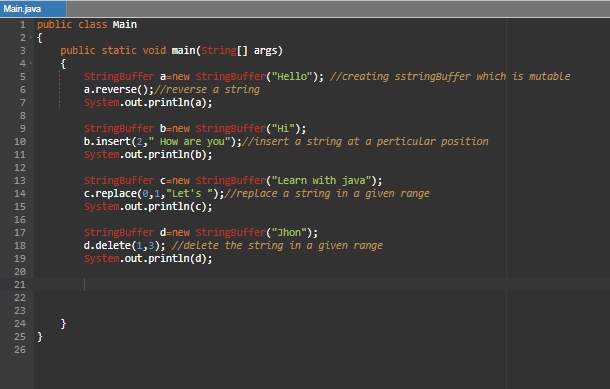
Output:



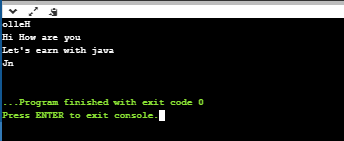
These are the methods for strings available in java.

* Above all strings are immutable that is we can not change those strings.
* If you want mutable string than you can use “StringBuffer” class or “StringBuilder” class
* StringBuilder class is similar to StringBuffer class. The difference between them is that StringBuffer is synchronized and StringBuilder is non-Synchronized

Example:



Output:



***Arrays***

* Multiple values of same data type is stored under one variable then it is called as Array.
* Arrays are used for random access of an element.
* Arrays are of two types:

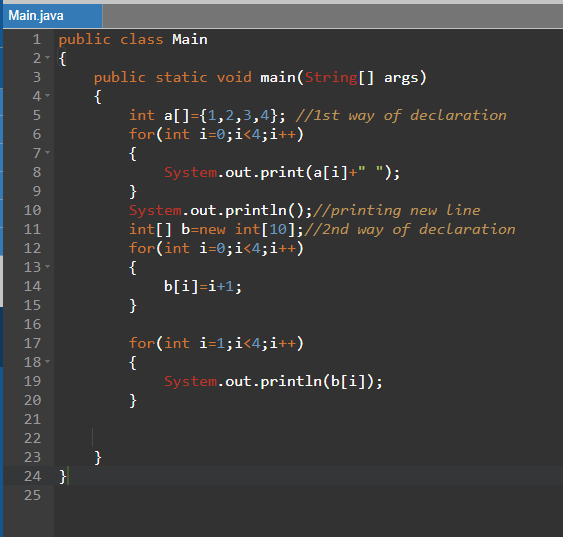
1. Single dimensional
2. Multi dimensional

* Single dimensional Array:

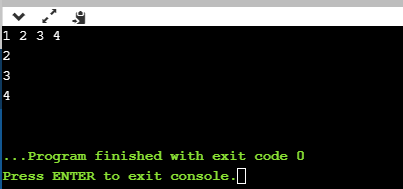
Syntax:

Data type[] arrayName = new Data type[size];

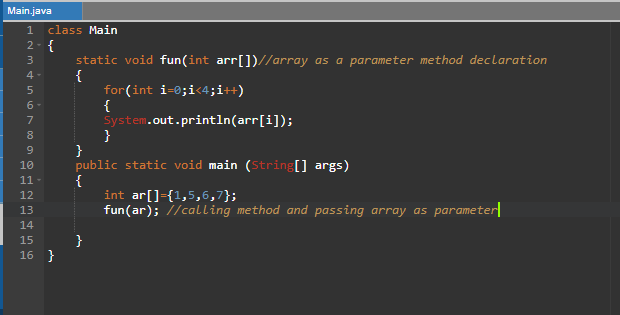
Example:



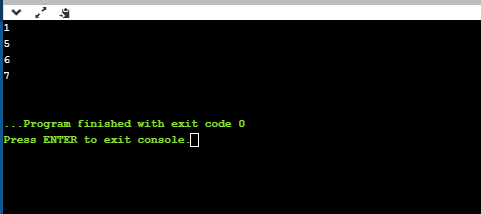
Output:



Example2:



Output:



* Multi-dimensional Array:

Multi dimensional arrays are nothing but the matrix.

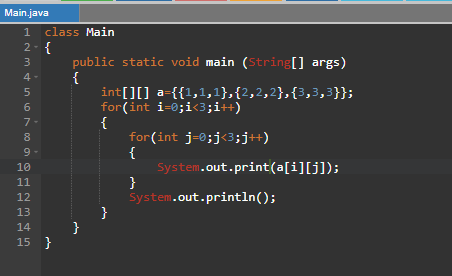
Syntax:

Data type[][] arrayName = new Data type[size1][size2];

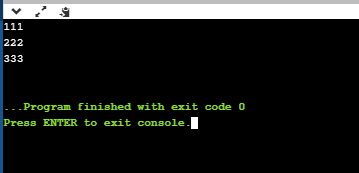
Where size1=number of rows

size2=number of columns

Example:

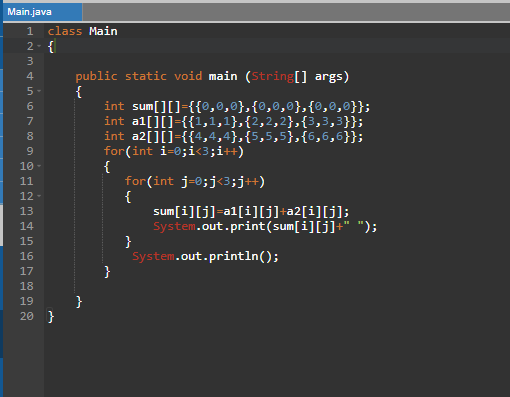


Output:



All the operations of single dimensional array and Multi dimensional array are similar.

Example2: addition of two matrix



Output:



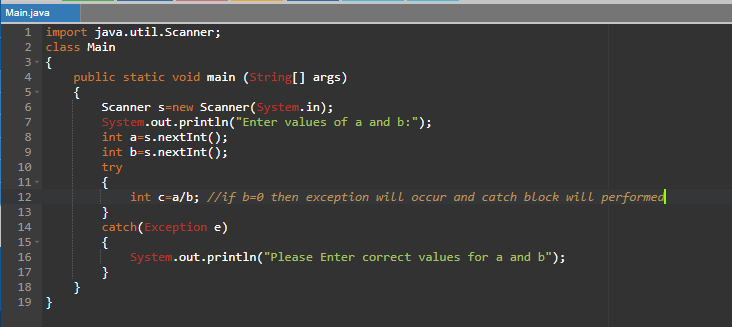
***Exception Handling***

* When the flow of your program get disturbed due to some exceptions then to handle that exception you can use exception handling.
* There are 5 keywords which are used in exception handling

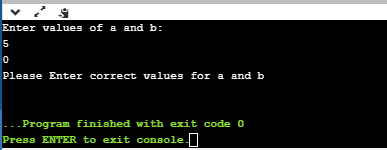
1. try
2. catch
3. throw
4. throws
5. finally

* try-catch block:

Examples:



Output:

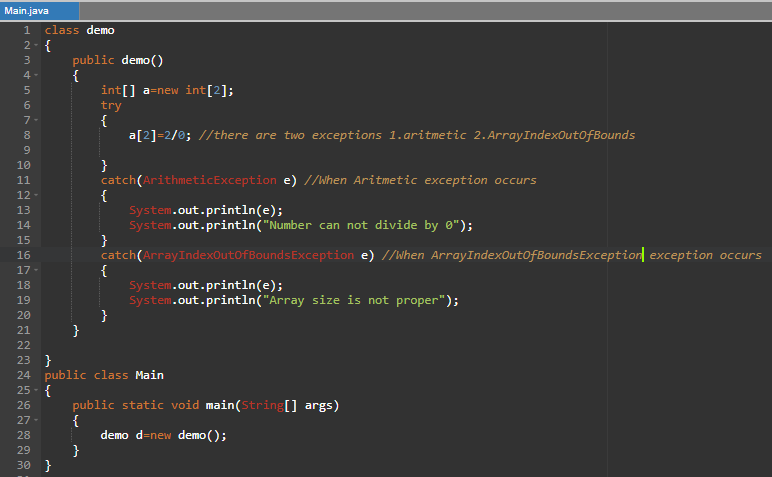


In try block if exception occurs then rest of the code in try block is not executed.

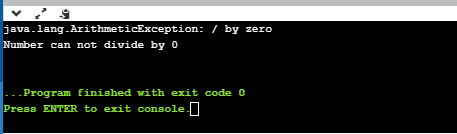
* Multiple catch blocks:

If there are two exceptions in a try block then you have to provide two catch blocks in java.

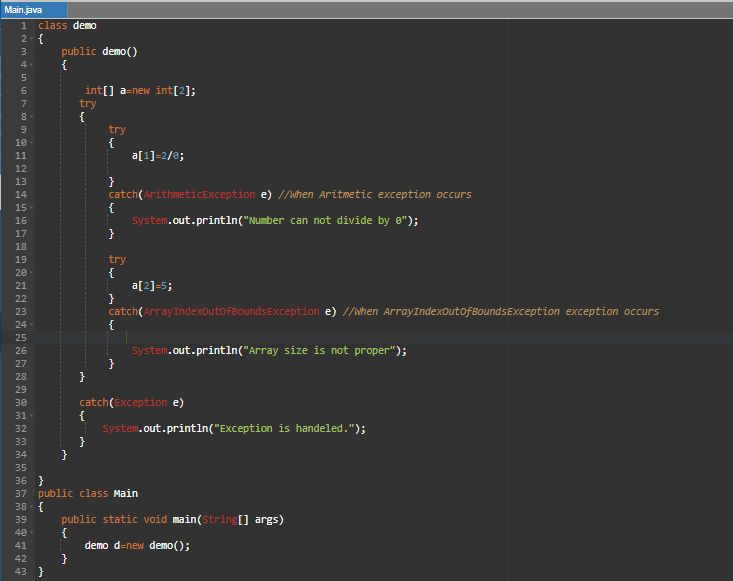
Example:



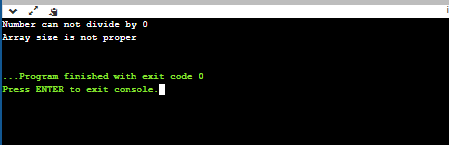
Output:



* Nested “try catch” block:



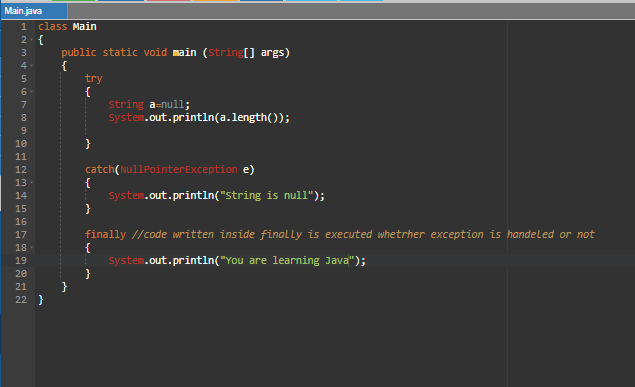
Output:



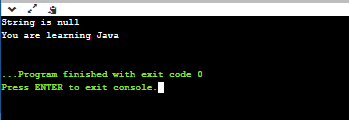
* “finally” block in java:

The code written inside the finally block is executed whether an exception occur or not.

Example:



Output:

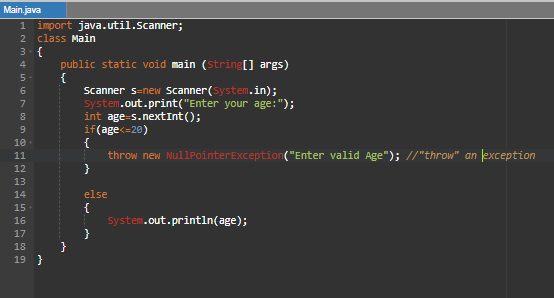


Note : For every try catch block there can be only one “finally” block.

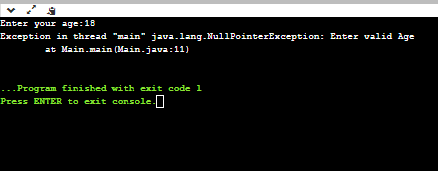
* “throw” keyword for exception:

“throw” keyword in java is used to create an object for an exception.

Example:



Output:



***Regular Expressions***

* Regular expressions in java are used to math a particular string with another String.
* Use:

It is used to validate any email , phone number etc.

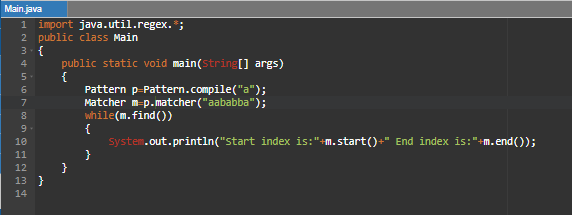
* Syntax:

Import java.util.regex.\*;

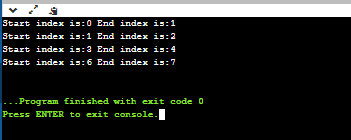
Pattern p=Pattern.compile(“Pattern to match”);

Matcher m=m.macther(“Target String”);

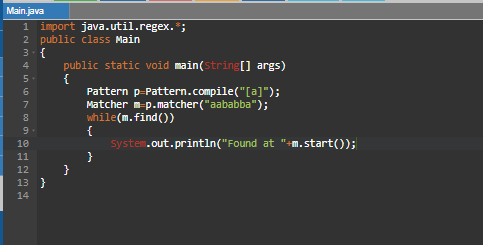
Example:



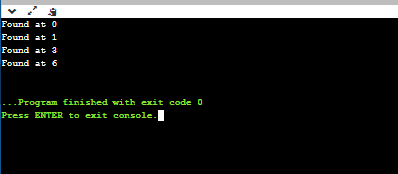
Output:

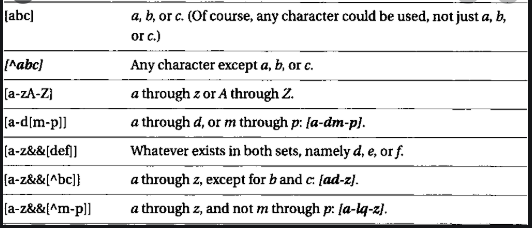


Example2:



Output:





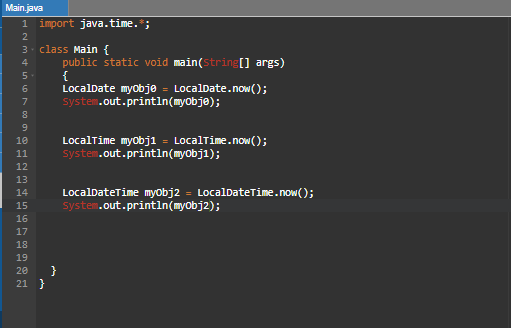
From above table , anything can be used for matching. Using all these regular expressions we can match any strings.

***Date and time***

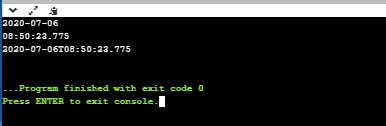
* Java does not contain date package but it contains “java.time” package through which we can access date and time
* There are many methods used in java.time package

|  |  |
| --- | --- |
| * LocalDate -Represents a date (year, month, day (yyyy-MM-dd)) * LocalTime- Represents a time (hour, minute, second and nanoseconds (HH-mm-ss-ns)) * LocalDateTime- Represents both Date and time (yyyy-MM-dd-HH-mm-ss-ns) * DateTimeFormatter- Formatter for displaying and parsing date-time objects |  |

Example:

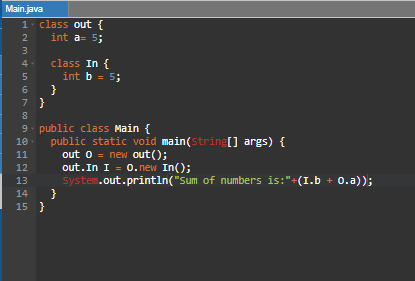


Output:

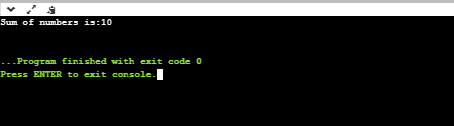


***Inner class***

* Inner classes are classes which are defined under the class.
* Example:



Output:



***Data Structures using java***

* The way of storing the data in memory is called as data structures.

1. Stack:

Stack is a LIFO data structure.

That is the element which is inserted last is removed first.

Program:

class Stack {

static final int MAX = 1000;

int top;

int a[] = new int[MAX]; // Maximum size of Stack

boolean isEmpty()

{

return (top < 0);

}

Stack()

{

top = -1;

}

boolean push(int x)

{

if (top >= (MAX - 1)) {

System.out.println("Stack Overflow");

return false;

}

else {

a[++top] = x;

System.out.println(x + " pushed into stack");

return true;

}

}

int pop()

{

if (top < 0) {

System.out.println("Stack Underflow");

return 0;

}

else {

int x = a[top--];

return x;

}

}

int peek()

{

if (top < 0) {

System.out.println("Stack Underflow");

return 0;

}

else {

int x = a[top];

return x;

}

}

}

class Main {

public static void main(String args[])

{

Stack s = new Stack();

s.push(30);

s.push(40);

s.push(50);

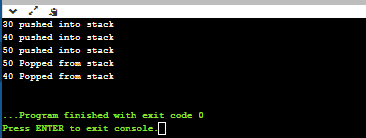
System.out.println(s.pop() + " Popped from stack");

System.out.println(s.pop() + " Popped from stack");

}

}

Output:



Queue:

Queue is a FIFO data structure where First In First Out data structure.

That is the element which is entered first is removed first.

Program:

import java.util.\*;

class Queue

{

private int arr[];

private int front;

private int rear;

private int capacity;

private int count;

Queue(int size)

{

arr = new int[size];

capacity = size;

front = 0;

rear = -1;

count = 0;

}

public void dequeue()

{

if (isEmpty())

{

System.out.println("UnderFlow\nProgram Terminated");

System.exit(1);

}

System.out.println("Removing " + arr[front]);

front = (front + 1) % capacity;

count--;

}

public void enqueue(int item)

{

if (isFull())

{

System.out.println("OverFlow\nProgram Terminated");

System.exit(1);

}

System.out.println("Inserting " + item);

rear = (rear + 1) % capacity;

arr[rear] = item;

count++;

}

public int peek()

{

if (isEmpty())

{

System.out.println("UnderFlow\nProgram Terminated");

System.exit(1);

}

return arr[front];

}

public int size()

{

return count;

}

public Boolean isEmpty()

{

return (size() == 0);

}

public Boolean isFull()

{

return (size() == capacity);

}

}

class Main

{

public static void main (String[] args)

{

Queue q = new Queue(5);

q.enqueue(1);

q.enqueue(2);

q.enqueue(3);

System.out.println("Front element is: " + q.peek());

q.dequeue();

System.out.println("Front element is: " + q.peek());

System.out.println("Queue size is " + q.size());

q.dequeue();

q.dequeue();

if (q.isEmpty())

System.out.println("Queue Is Empty");

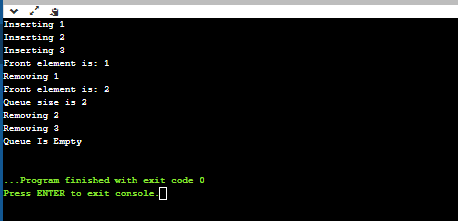
else

System.out.println("Queue Is Not Empty");

}

}

Output:



List:

List is a data structure in which user can not jump from one any element to another directly.

One element is linked with other element.

Every element is having two things,

Program:

class LinkedList {

StackNode root;

static class StackNode {

int data;

StackNode next;

StackNode(int data)

{

this.data = data;

}

}

public boolean isEmpty()

{

if (root == null) {

return true;

}

else

return false;

}

public void push(int data)

{

StackNode newNode = new StackNode(data);

if (root == null) {

root = newNode;

}

else {

StackNode temp = root;

root = newNode;

newNode.next = temp;

}

System.out.println(data + " pushed to stack");

}

public int pop()

{

int popped = Integer.MIN\_VALUE;

if (root == null) {

System.out.println("Stack is Empty");

}

else {

popped = root.data;

root = root.next;

}

return popped;

}

public int peek()

{

if (root == null) {

System.out.println("Stack is empty");

return Integer.MIN\_VALUE;

}

else {

return root.data;

}

}

}

class Main

{

public static void main(String[] args)

{

LinkedList sll = new LinkedList();

sll.push(40);

sll.push(80);

sll.push(20);

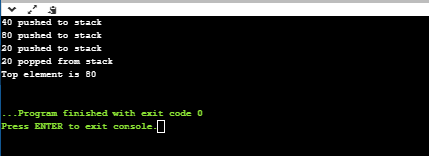
System.out.println(sll.pop() + " popped from stack");

System.out.println("Top element is " + sll.peek());

}

}

Output:

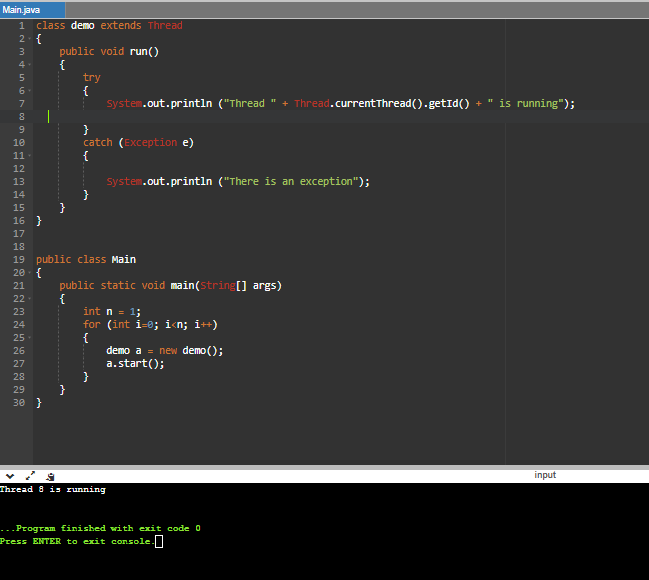


***Multithreading***

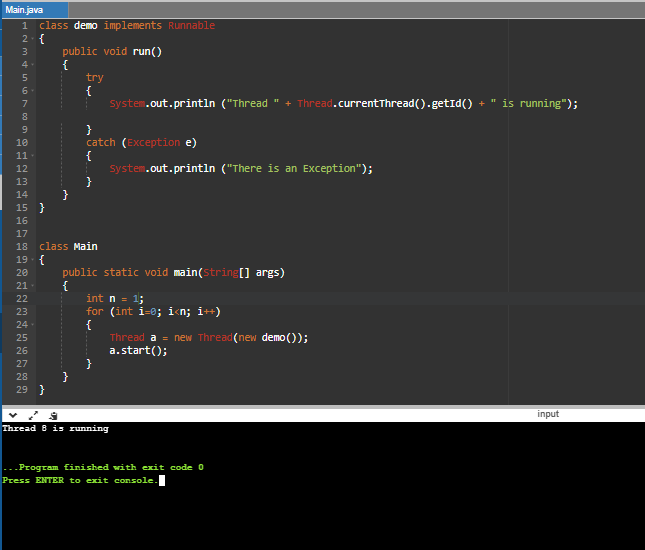
* Multithreading in java is used to perform two treads simultaneously.
* It increases the speed of your program.
* There are two ways to implement multithreading that is

1. By Inheritance of thread class
2. By implementing runnable interface

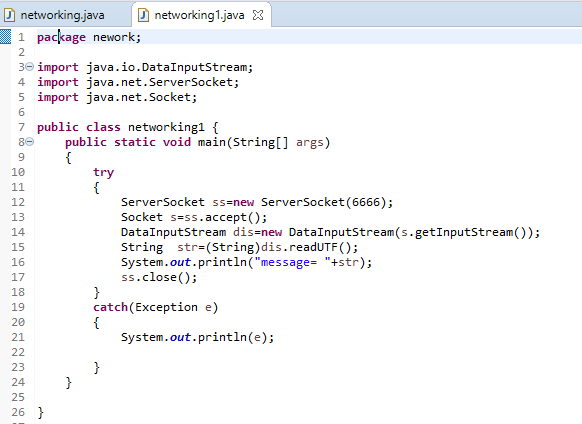
Example1:

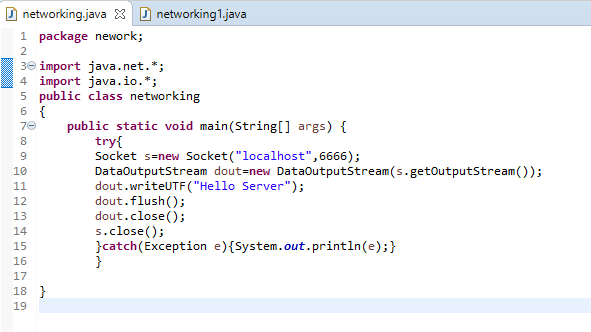


Example2: Using Runnable interface

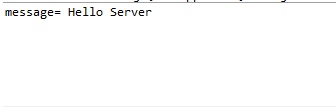


***Networking in java***

* Java networking is used for communication between client and server.
* There are two terms used in java networking UDP and TCP.
* The main difference between UDP and TCP is that UDP does not require any connection but TCP requires the connections.
* TCP requires “IP Address and Port number”
* Example:



Output:



***Sending Email***

* Using java we can send email. For sending email we require two files that is “activation.jar” and “mail.jar” file.
* In these .jar files there are many interfaces and classes are present using which we can send emails.
* First step Is to create an object of email class
* Then for sending email use inbuilt methods which are listed below;
* setFrom(“EmailID” , “Name of company”);
* setSubject(“EmailId”,”Subject”);
* setContent(“content” ,“text/html”);
* addRecipients(“EmailId”);
* send();

Example:

import **java**.util.Properties;

import javax.**mail**.\*;

import javax.**mail**.internet.\*;

class Email

{

Public static void main(String args[])  
 {

try{

Email e=new Email(“EmailId”, “Password”);

e. setFrom(“EmailID” , “Name of company”);

e. setSubject(“EmailId”,”Subject”);

e. setContent(“content” ,“text/html”);

e. addRecipients(“EmailId”);

e.send();

System.out.print(“Send Successfully”);

}

catch(exception e)

{

}

}

}

Using above example we can send emails in java.

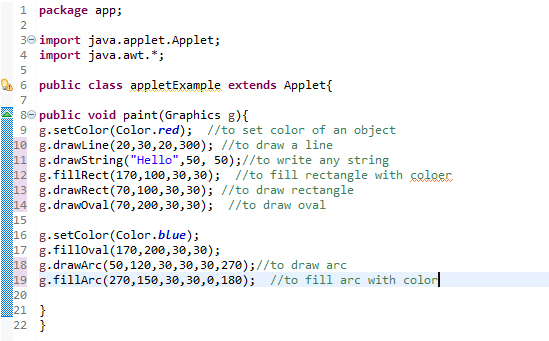
***Applets***

* Applets in java are used for creation of GUI(Graphical User Interface).
* Uses of Applets:

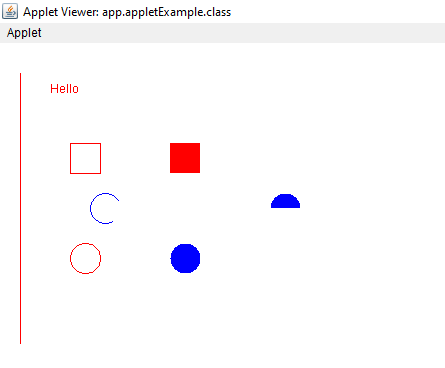
It is used for creation of shapes in java.

It is used for creation of Any GUI frame.

Example:

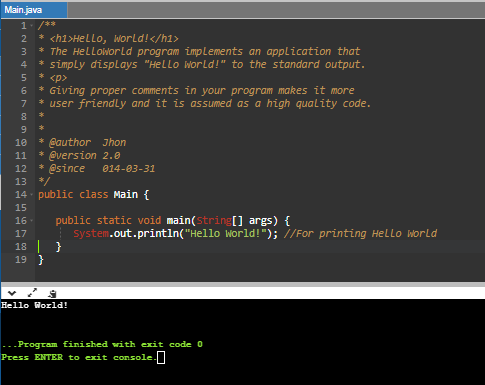


Output:



***Documentation***

* Documentation in java is used for writing comments in a program.
* There are two types of comments in java one is Single line and other is Multiline comments.
* Example1:



For more information about comments refer to the comments part of operators.

***Collections and generics***

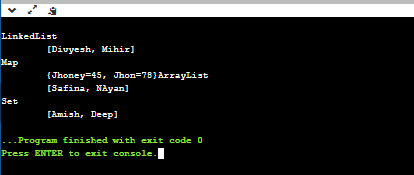
* Collection in java is a framework which provides interface
* It provides all the information about interfaces

1. Collection interface
2. List interface
3. Set interface
4. Sorted set interface
5. Map interface
6. Map Entry interface
7. Sorted Map interface
8. Enumeration

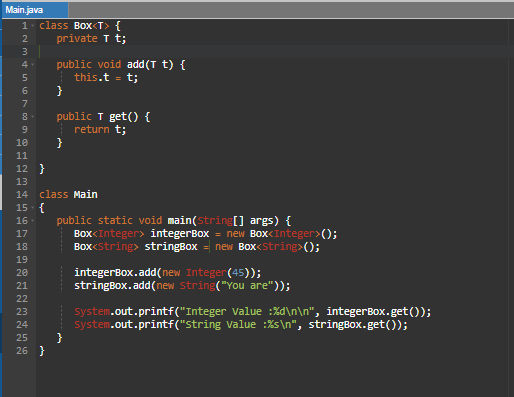
Example:



Output:



* ***Generics in java:***
* The main difference between java Collections and generics is that we can define data type of an element.
* For example, If we define a arraylist as an integer then we can not add any other data type value to that list
* ***Example:***

******

***Output:***

******

References:

Youtube.com

Udemy

javatpoint