

## Introduction To Computer

=====

**Computer** - Electronic device which reduce human effort

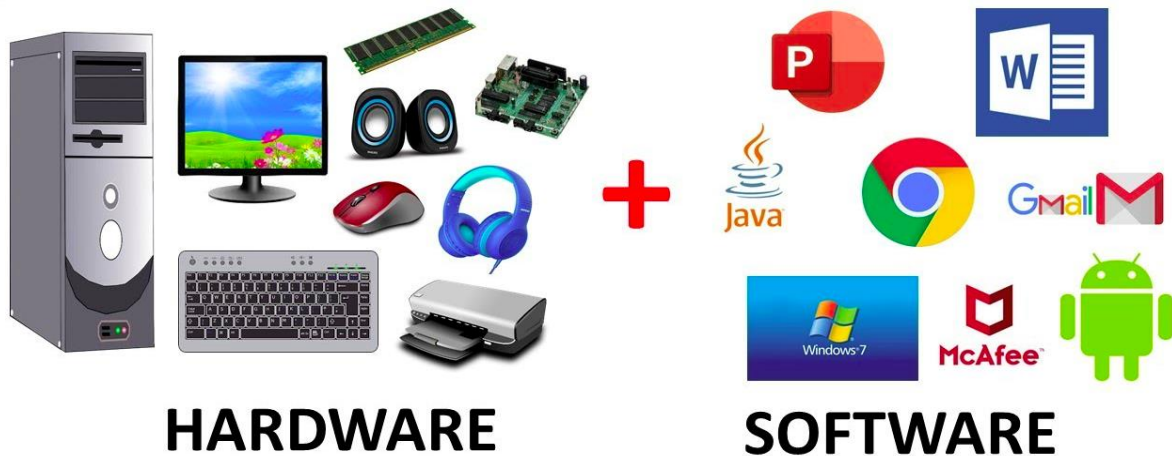
apple mac, lenovo, hp, dell etc...

**Computer Hardware** is defined as the physical part or component of a computer system which can be seen and touched.

Eg. Monitor, keyboard, mouse , CPU, printer etc

**Computer software** is defined as a set of instructions or collection of programs which are designed and developed to perform specific tasks.

Used to reduce human efforts



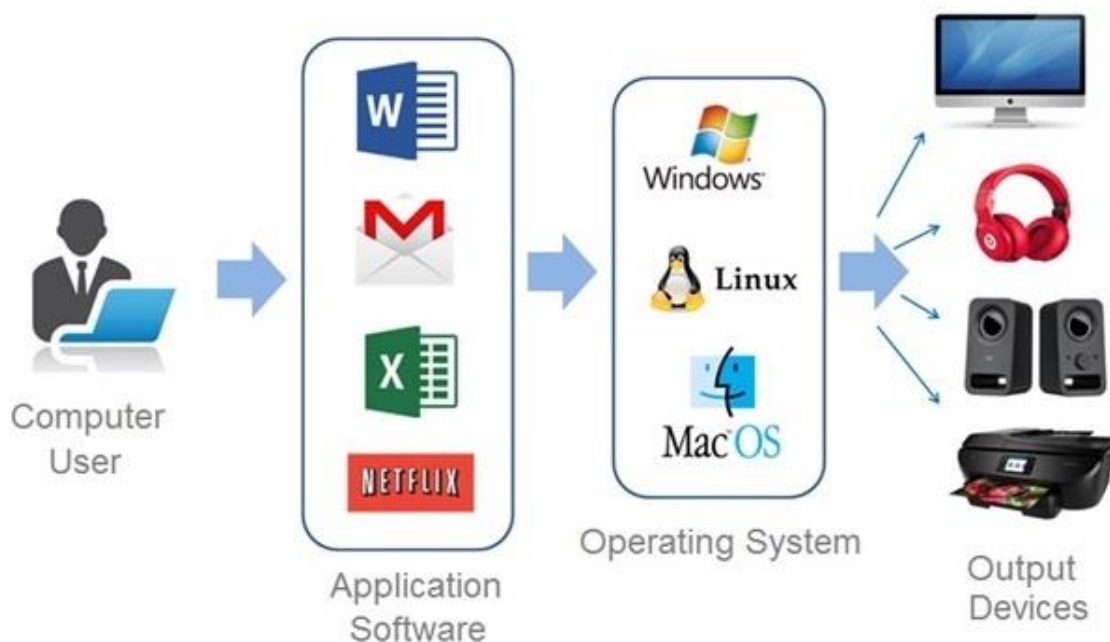
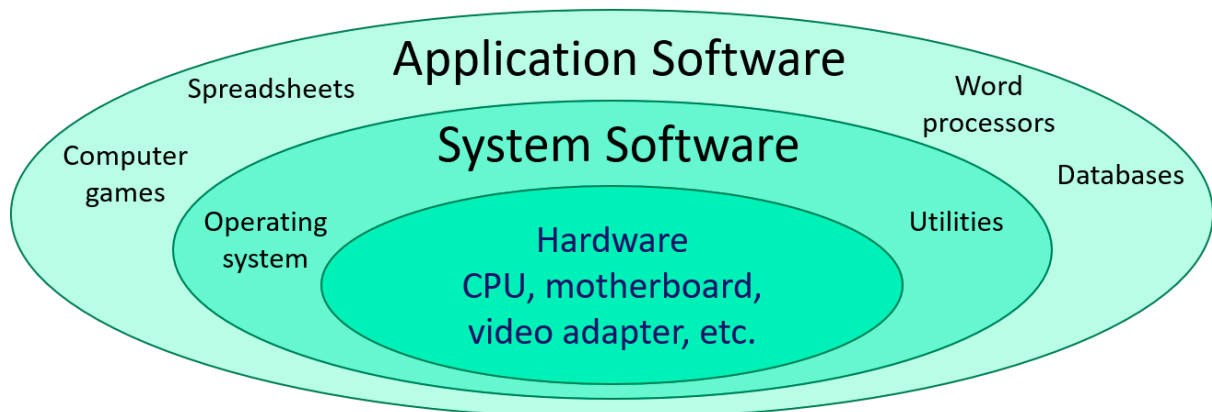
Types of Software

### **System Software -**

- Also called as Platforms or Operating testing
- The **system software** is software which controls the hardware so that any application software can run and can be executed to perform various task mentioned by programmers.
- The primary examples of system software's are operating system such as Microsoft Windows, Linux, Mac, Unix, etc

## Application Software -

- Application Software is a program that is designed and developed for specific purposes and used by user
- application software such as MS-OFFICE SUITE comes with MS Word, Excel, PowerPoint & Access and Adobe include Adobe Photoshop and Image ready together.



As we know, to communicate with a person, we need a specific language, similarly to communicate with computers, programmers also need a language is called Programming language.

## What is Language?

Language is a mode of communication that is used to **share ideas, opinions with each other.**

## What is a Programming Language?

A programming language is a **computer language** is used by **programmers (developers) to communicate with computers.**

What are different programming languages -

C, C++, Java, Python etc

So, it can also be defined. It is a set of instructions written in any specific language ( C, C++, Java, Python) to perform a specific task.

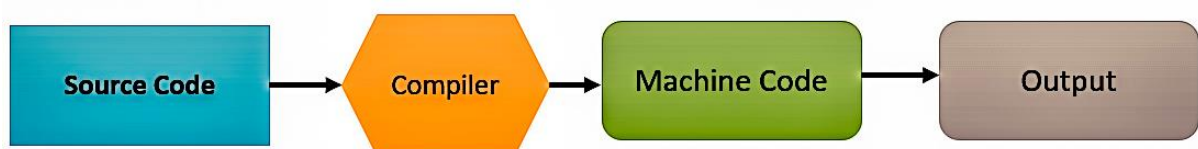
- A computer is a machine that can perform tasks according to the instructions provided by the user.
- Which is provided by user in the form of source code.

## What is Source Code and machine code or byte code?

Source code is set of instructions those are human-readable that a programmer writes to perform specific task.

Machine code Byte code or Binary code - The codewhich is converted by compiler which will be udertstaabkle by computer

## How Program get execute



# JAVA

# What is Java?

- Java is general purpose programming language that is class based, object-oriented, platform independent and platform to develop an application.
- It provides software platform that runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices and many others.

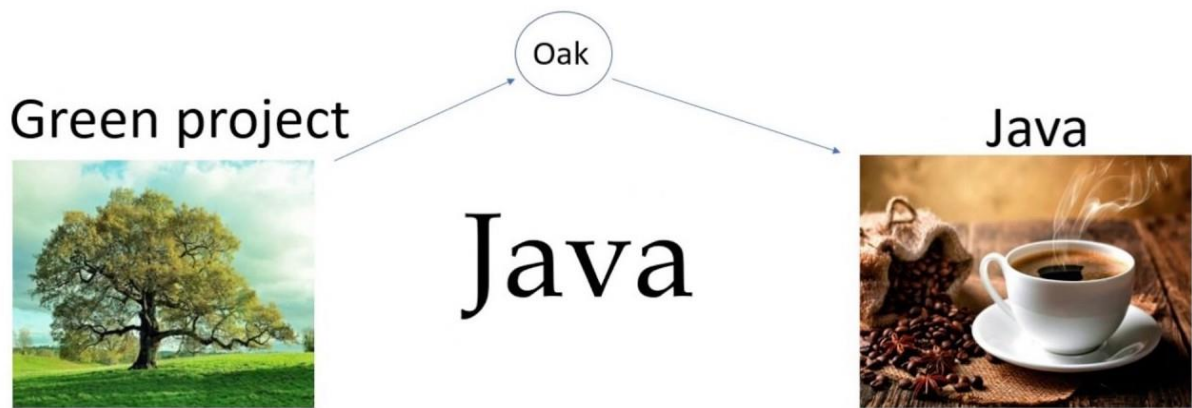


## History of JAVA :-

1. Java was originally developed by James Gosling's at sun microsystem (which has been acquired by oracle in 2010) and released in 1995.
2. James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. The small team of sun engineers called Green Team.
3. Initially it was designed for small, embedded systems in electronic appliances like set-top boxes.
4. it was called Oak and was developed as a part of the Green project.

## Why was Java named as "Oak"?

1. Why Oak? Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.
2. In 1995, Oak was renamed as "Java" because it was already a trademark by Oak Technologies.



### Why is Java Programming named "Java"?

3. The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA", etc.
4. Java was so unique, most of the team members preferred Java over other names.
5. Java is an island in Indonesia where the first coffee was produced (called Java coffee). It is a kind of espresso bean. Java name was chosen by James Gosling while having a cup of coffee nearby his office.
6. In 1995, Time magazine called Java one of the Ten Best Products of 1995.

### Java Version History

JDK 1.0 was released on January 23, 1996. After the first release of Java, there have been many additional features added to the language. Each new version adds new features in Java.

Sr.No	Version	Release date
1	JDK Alpha and Beta	1995
2	JDK 1.0	23rd Jan 1996
3	JDK 1.1	19th Feb 1997
4	J2SE 1.2	8th Dec 1998
5	J2SE 1.3	8th May 2000
6	J2SE 1.4	6th Feb 2002

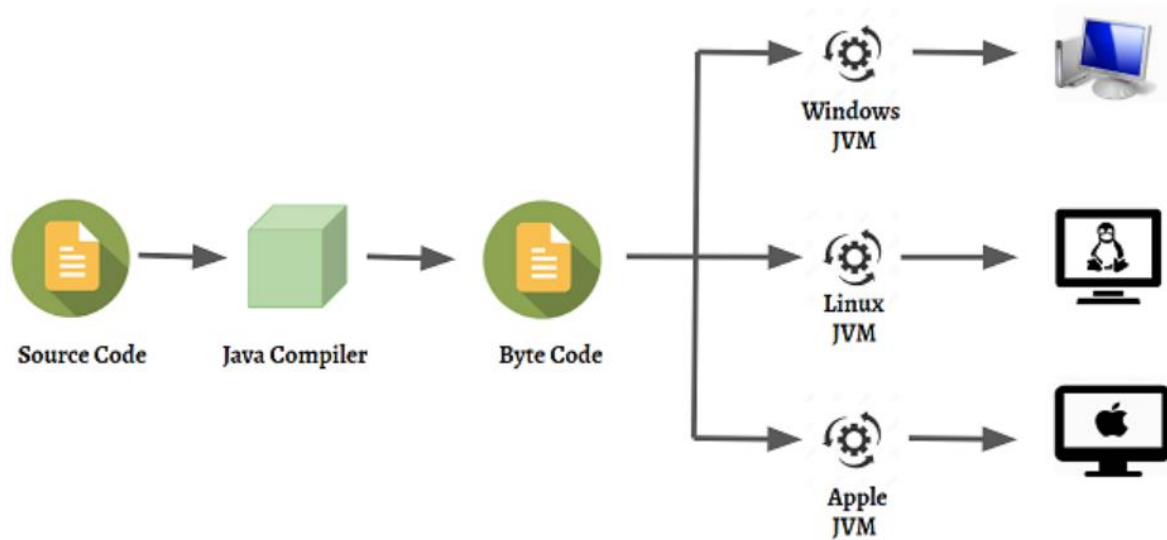
7	J2SE 5.0	30th Sep 2004
8	Java SE 6	11th Dec 2006
9	Java SE 7	28th July 2011
10	Java SE 8	18th Mar 2014
11	Java SE 9	21st Sep 2017
12	Java SE 10	20th Mar 2018
13	Java SE 11	September 2018
14	Java SE 12	March 2019
15	Java SE 13	September 2019
16	Java SE 14	Mar 2020
17	Java SE 15	September 2020
18	Java SE 16	Mar 2021
19	Java SE 17	September 2021
20	Java SE 18	to be released by March 2022

- a. It promises WORA = "**Write One Run Anywhere**"
- b. Since Java SE 8 release, the Oracle corporation follows a pattern in which every even version is release in March month and an odd version released in September month.

### How Java is WORA?

- **WORA**, which is abbreviated as **Write Once Run Anywhere**, is the feature applicable to those programs which will be run on any operating systems or any machine. Sun Microsystem gave this terminology for their programming language - Java.
- According to this concept, the same code must run on any machine, and hence the source code needs to be portable.
- So Java allows run Java bytecode on any computer irrespective of the machine or the hardware, using JVM (Java Virtual Machine).

- The bytecode generated by the compiler is not platform-specific and hence takes the help of JVM to run on a wide range of machines. So we can call Java programs as a write once and run on any computer residing anywhere.



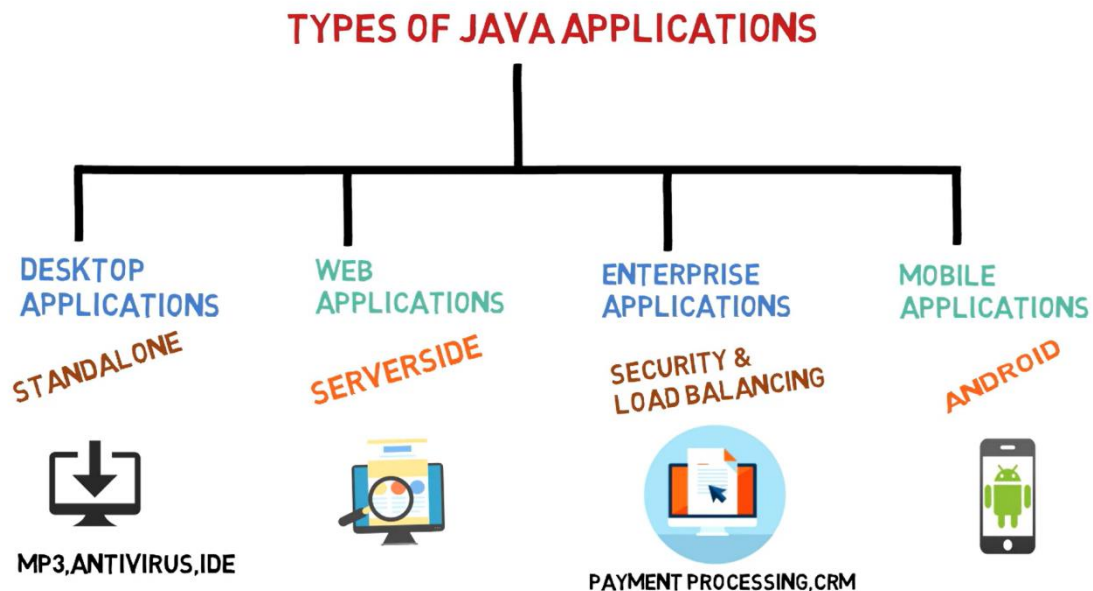
## Application

According to Sun, 3 billion devices run Java. There are many devices where Java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus, etc.
2. Web Applications such as irctc.co.in, javatpoint.com, etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games, etc.

## Types of Java Applications

There are mainly 4 types of applications that can be created using Java programming:



### 1. Standalone Application

- a. Also called as Desktop based or windows-based application. These are software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc.
- b. AWT and Swing are used in Java for creating standalone applications.

### 2. Web Application

- c. An application that runs on the server side and creates a dynamic page is called a web application. Examples Facebook, amazon etc.
- d. Servlet, JSP, Struts, Spring, Hibernate, JSF, etc. technologies are used for creating web applications in Java.

### 3. Enterprise Application

- e. An application that is distributed in nature, such as banking applications, etc. is called an enterprise application.
- f. It has advantages like high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.



#### 4. Mobile Application

- g. An application which is created for mobile devices is called a mobile application.
- h. Currently, Android and Java ME are used for creating mobile applications.

#### Java Platforms / Editions



#### 1. Java SE (Java Standard Edition)

- a. It is a Java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc.
- b. It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

#### 2. Java EE (Java Enterprise Edition)

- c. It is an enterprise platform that is mainly used to develop web and enterprise applications.
- d. It is built on top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA, etc.

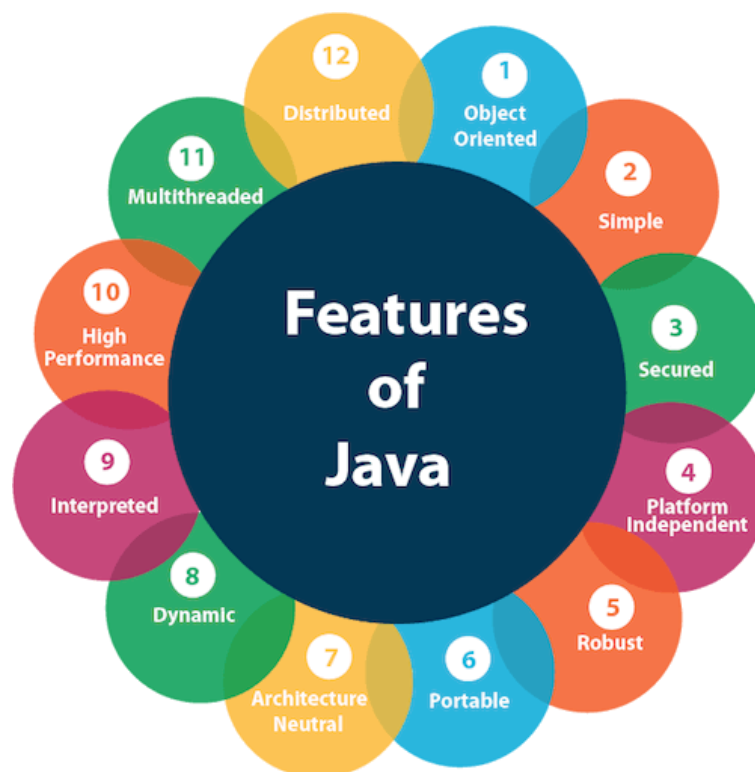
#### 3. Java ME (Java Micro Edition)

- e. It is a micro platform that is dedicated to mobile applications.

## Features of Java

A list of the most important features of the Java language is given below.

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Interpreted
9. High Performance
10. Multithreaded
11. Distributed
12. Dynamic



- ❖ **Object Oriented** – In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- ❖ **Simple** – Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- ❖ **Secure** – With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

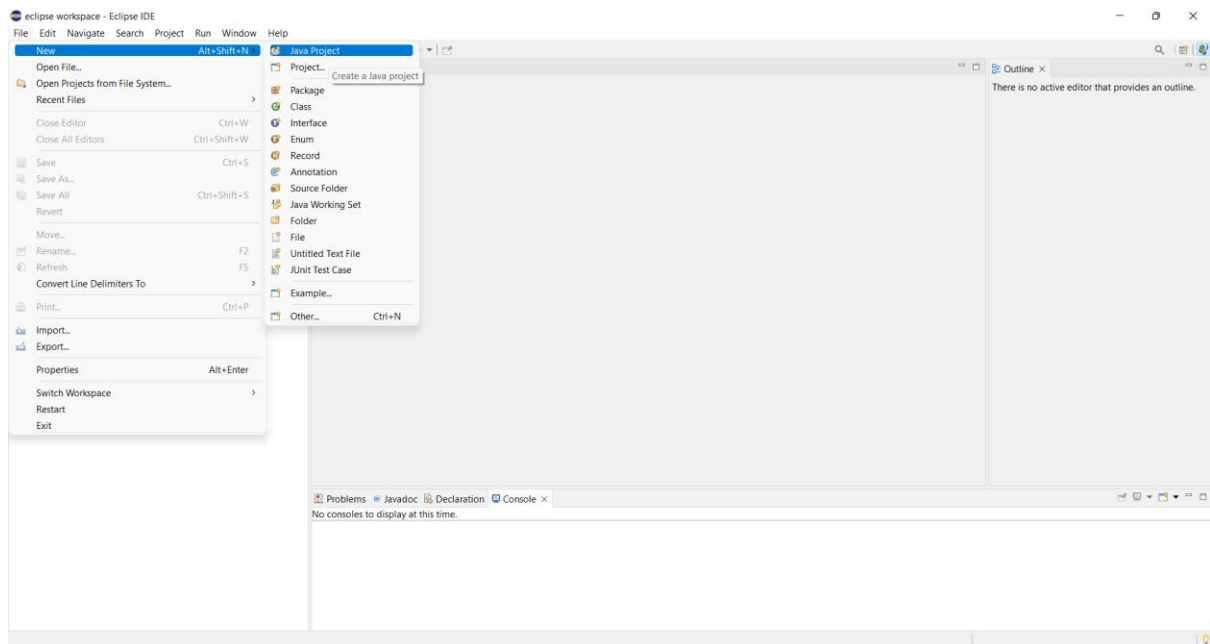
- ❖ **Portable** – Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- ❖ **Platform Independent** – Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- ❖ **Architecture-neutral** – Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- ❖ **Robust** – Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- ❖ **Interpreted** – Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
- ❖ **High Performance** – With the use of Just-In-Time compilers, Java enables high performance.
- ❖ **Multithreaded** – With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- ❖ **Distributed** – Java is designed for the distributed environment of the internet.
- ❖ **Dynamic** – Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

=====

## Description about java program structure

### 1. Project –

- a. Project is a collection of multiple packages, multiple classes, user defined folder (for test data), supporting libraries, jar files and so on.
- b. We can create many packages and classes inside a project.

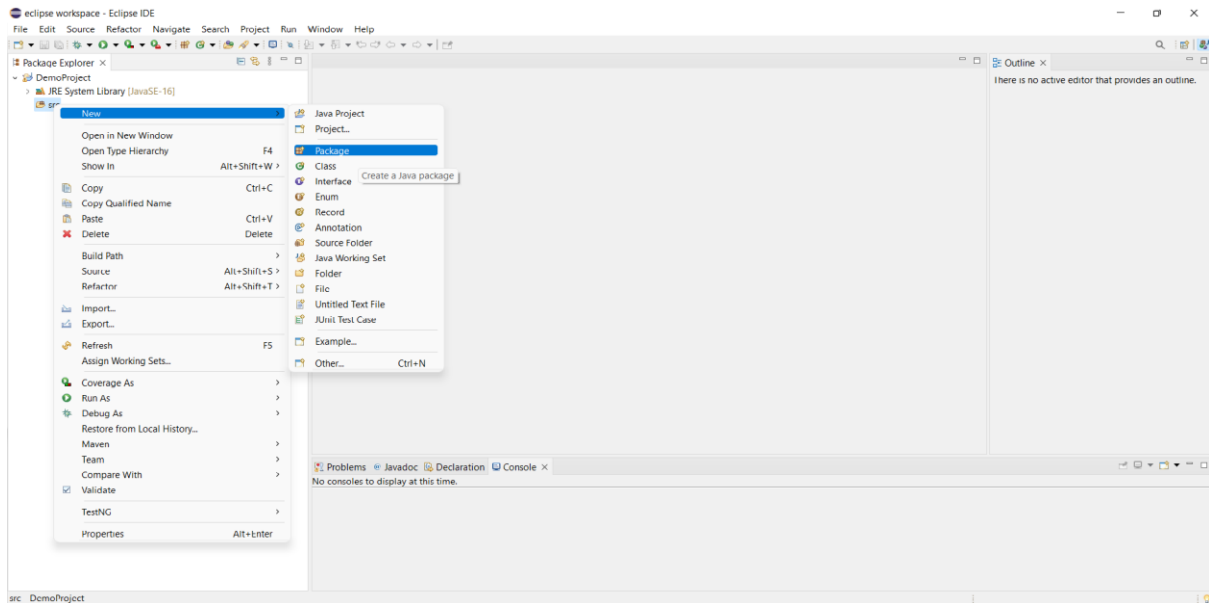


### 2. Package –

- c. Package is a collection of multiple classes
- d. There are two types of packages
  - i. User defined package – If we create package
  - ii. Default package – If we are created number of classes without generating package and shows under default package.
- e. Syntax

---

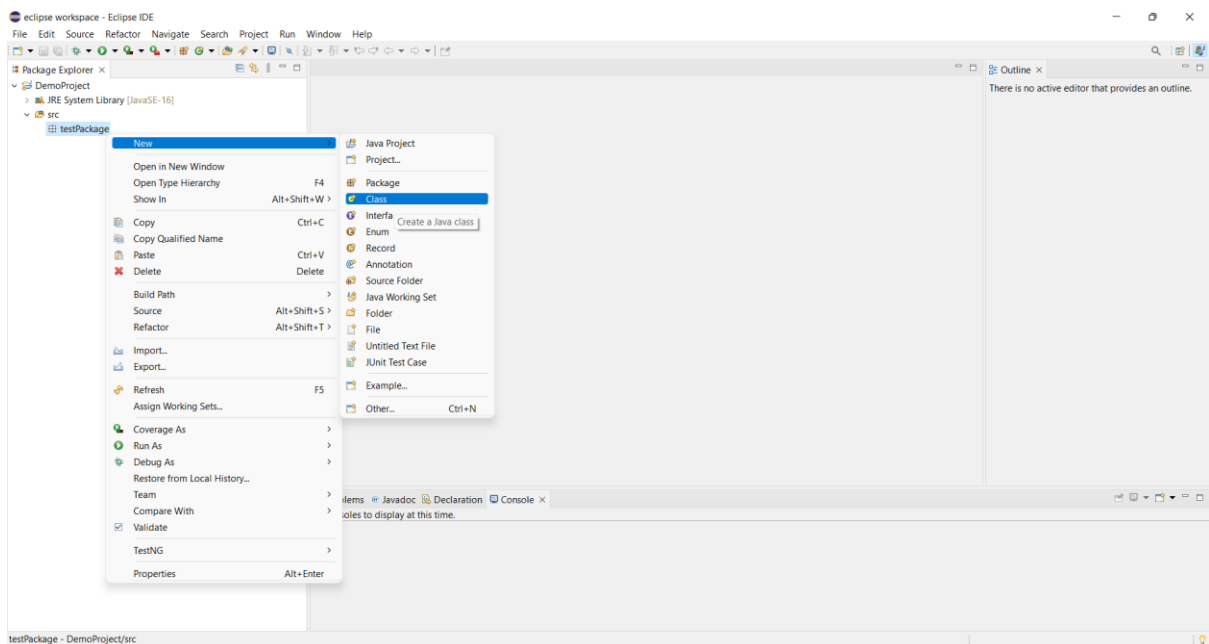
**package package\_name ;**
- f. If package does not contain class then symbol should be colourless
- g. If class created inside package the symbol should be colourful



### 3. Class

- A class is collection of member functions(methods), variable, datatypes, printing statements, scanning statement, object, constructor etc.
- For all class name the first letter should be uppercase
- If several words are used to form a name of the class each inner words first letter should be in upper case

**Eg. public class MyFirstJavaProgram**



## Creating a Java Project in Eclipse IDE

### ❖ How to create a project

- Open Eclipse IDE Click on file >> New >> Java Project >> { Project name }  
>> Click on finish >> Don't Create

### ❖ How to create package

- click on src >> right click >> new >> select package >> { package name }  
>> finish

### ❖ How to create class

- Click on specific package>> right click >> new >> select class >> provide  
name >> finish

=====

## Case sensitivity/Name Convention -

Java is a case sensitive which means identifier "Hello" and ""hello" which have different meaning in java

### 1. ProjectName

- All project name start with uppercase letter
- If have multi name project, then uppercase of each first letter of word

**Eg. DemoProject**

### 2. PackageName

- all method starts with lower case letter
- If have multi name package, then lowercase of first word and upper case of next words

**Eg. testPackage**

### 3. ClassName

- All classname start with uppercase letter
- If have multi name class, then uppercase of each first letter of word

**Eg. public class MyFirstJavaProgram**

### 4. methodName

- All method starts with lower case letter

- If have multi name method, then lowercase of first word and upper case of next words

**Eg. myMethodName()**

## **Basic Terms and Rules**

### **5. Comments(//)**

- These lines are used for comments or writing some statement that will be not considered during execution of programs

### **6. Signature Bracket ()**

- This bracket we called as signature bracket or argument bracket

### **7. + Sign**

- This + sign is used for concatenating or we can say that join the string or statements

### **8. JRE library**

- In JRE library there are supporting jar files to write java programs

### **9. { }**

- It is called as body or curly braces  
eg. class body, method body etc.

### **10.src**

- src is a source folder where the project source file content programs

### **11.System.out.println()**

- System - it is name of java utility class
- out - it is object which belongs to system class to call predefined method
- println - it is predefined method or utility method which is used to send any string to console

=====

## Method/Member Function in Java

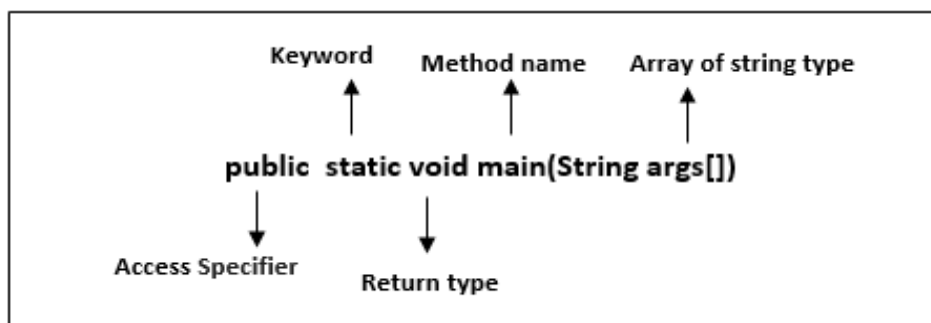
- Method in java is a collection of instructions that perform specific task. Without method we are not able to write program.
- It is also called member function
- It is memory location in JVM to store the data and information
- Why use methods? To reuse code: define the code once, and use it many times.
- A method must be declared within a class.

There are two types of methods

1. Business method – main method
2. Regular method

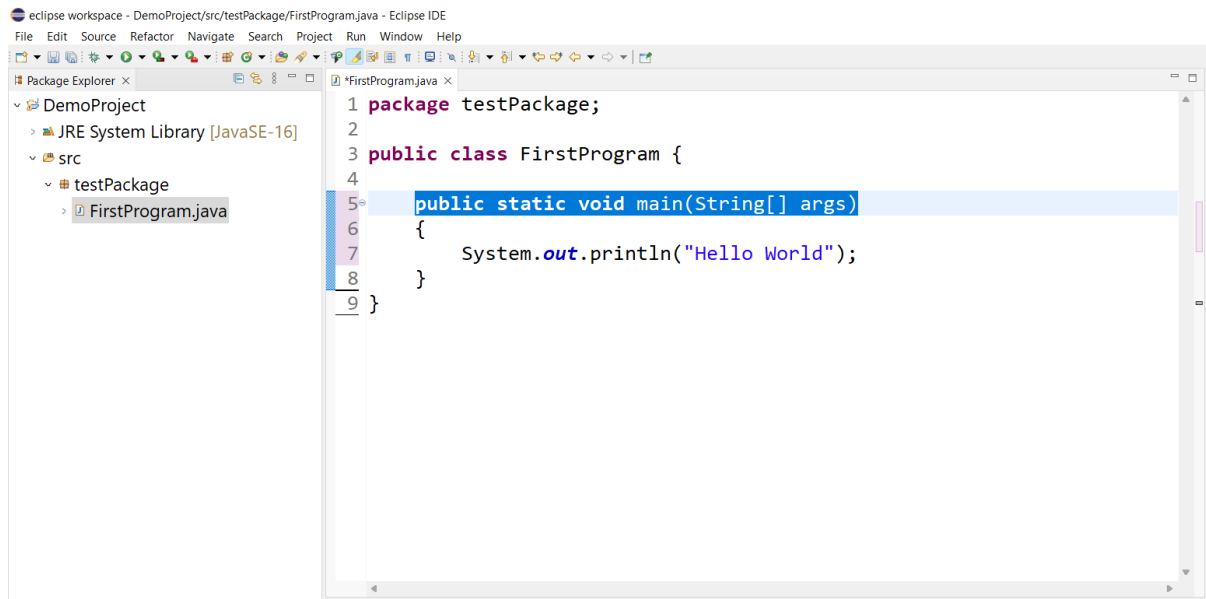
## Java main() method

- The main() is the starting point for JVM to start execution of a Java program. Without the main() method, JVM will not execute the program.
- The syntax of the main() method is:



- **public:** It is an access specifier. We should use a public keyword before the main() method so that JVM can identify the execution point of the program.
- **static:** You can make a method static by using the keyword static. We should call the main() method without creating an object.
- **void:** In Java, every method has the return type. Void keyword in main() method does not return any value.
- **main():** It is a default method name which is predefined in the JVM.
- **String args[]:** The main() method also accepts some data from the user. It accepts a group of strings, which is called a string array.





## Regular method

Two types of methods

1. Static method
2. Non-Static method

### 1. Static Method

- The method which contains static keyword and used to perform particular task
- A static method in Java is a method that is part of a class rather than an instance of that class. (To call static method there is no need to create object.)
- It is static in nature
- Main method is the best example of static method
- It can be represented as

**public static void methodName()**

**{**

**Body of the program for execution.**

**}**

- It can call as **ClassName.methodName();** or only **methodName();**
  - ❖ Class variables and methods can be accessed using the class name followed by a dot and the name of the variable or method.

## 2. Non-Static Method

- The method which don't have static keyword
- It is dynamic in nature
- To call non-static method we need to create object

**public void methodName()**

**{**

**Body of the program for execution.**

**}**

- It can call as object.methodName();

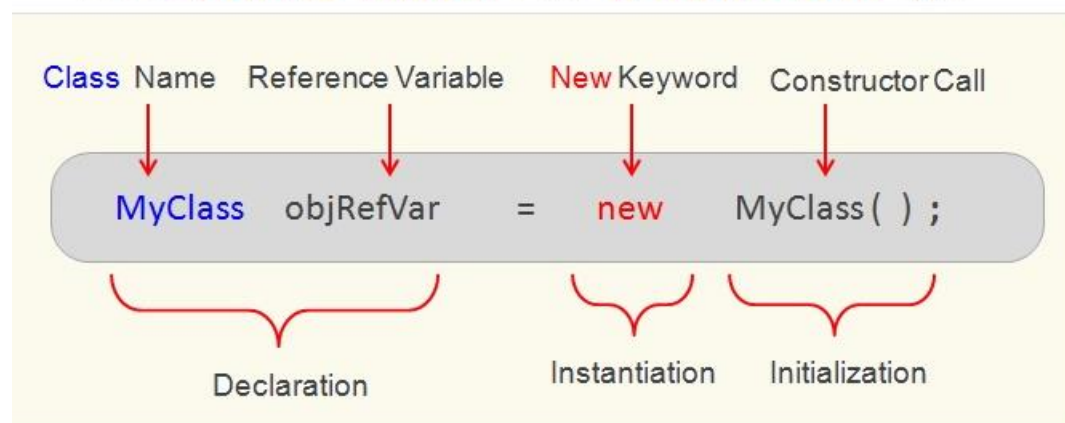
## Object

- It is instance of class/example of class and used to call non-static method.
- An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical.

How to Create Object in Java using new keyword

**ClassName object = new ClassName();**

Java Reference Variable And Reference Data Type



`objRefVar` is a Reference Variable of `MyClass` Type .

`new` is a Java Keyword used To Instantiate a `Class` .

## ClassName

- ❖ It is name of class at the time of object creation.
- ❖ Class is used as datatype to declare reference variable.

## ObjectReferenceVariable

- ❖ Used to provide the reference of object.

## New

- ❖ It is keyword used to create object of class.

## Constructor();

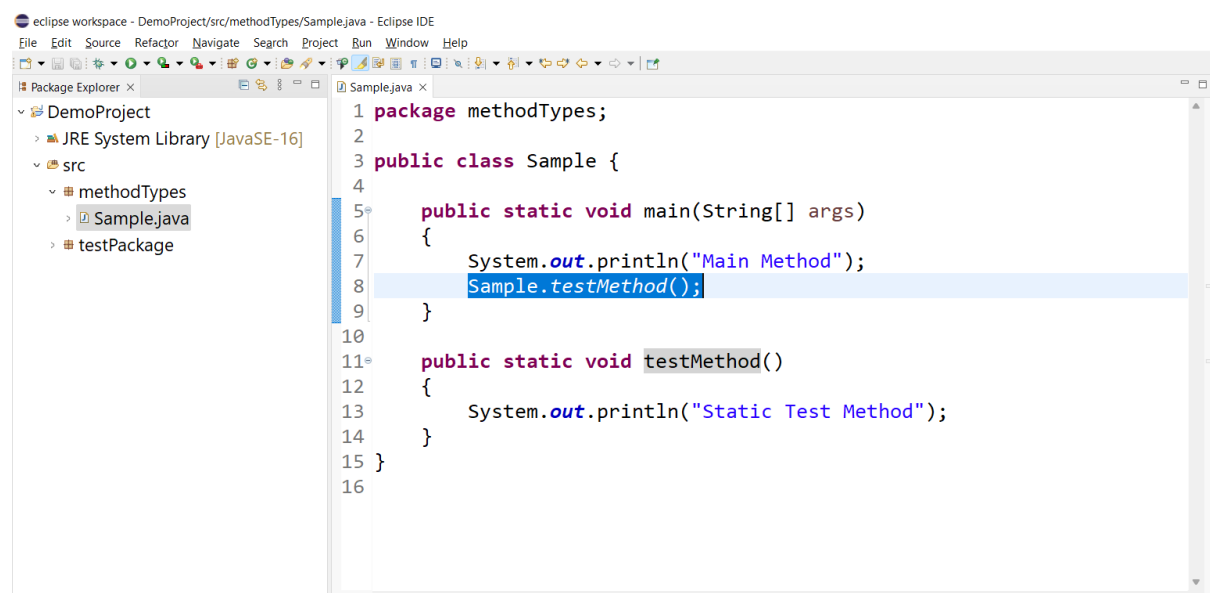
- ❖ Constructor name should be same as class name.
- ❖ Used to initialize the information of the particular class.

=====

## Why is the main method in Java static?

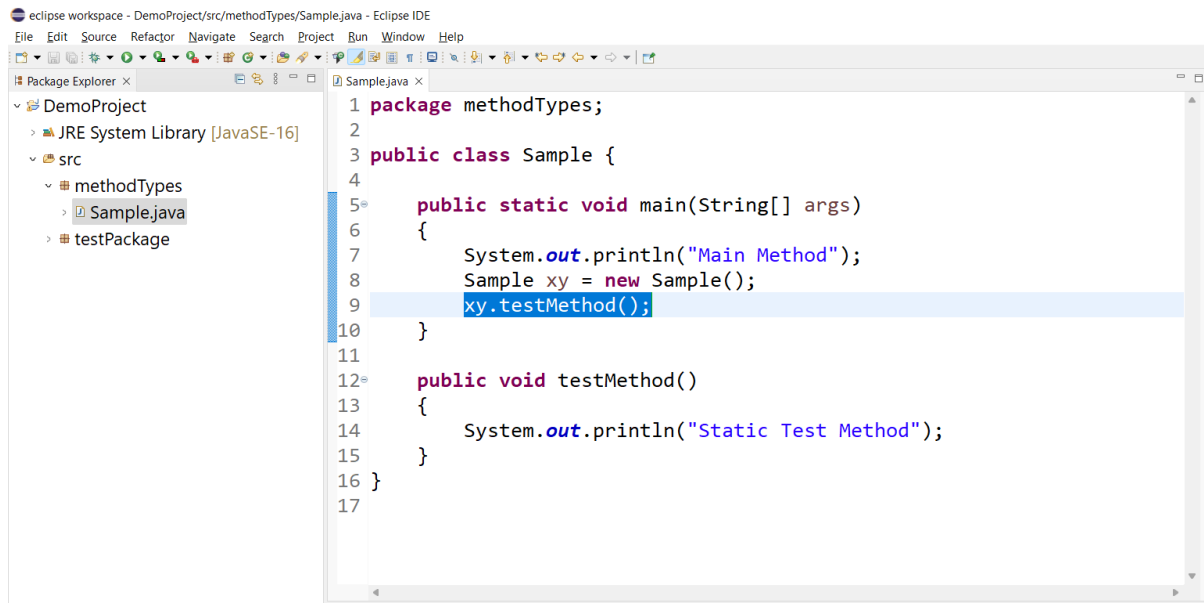
It's because calling a static method isn't needed of the object. If it were a non-static function, JVM would first build an object before calling the main() method, resulting in an extra memory allocation difficulty.

## Static Method Example -



```
1 package methodTypes;
2
3 public class Sample {
4
5     public static void main(String[] args)
6     {
7         System.out.println("Main Method");
8         Sample.testMethod();
9     }
10
11     public static void testMethod()
12     {
13         System.out.println("Static Test Method");
14     }
15 }
16
```

## Non Static Method Example -



```
1 package methodTypes;
2
3 public class Sample {
4
5     public static void main(String[] args)
6     {
7         System.out.println("Main Method");
8         Sample xy = new Sample();
9         xy.testMethod();
10    }
11
12    public void testMethod()
13    {
14        System.out.println("Static Test Method");
15    }
16 }
17
```

=====

## Data Types in Java

- ❖ Datatypes are used to represent types of data or information that we going to used in the programming.
- ❖ It is mandatory / compulsory to declare datatype before variable.

Types of data types in Java:

### 1. **Primitive data types:**

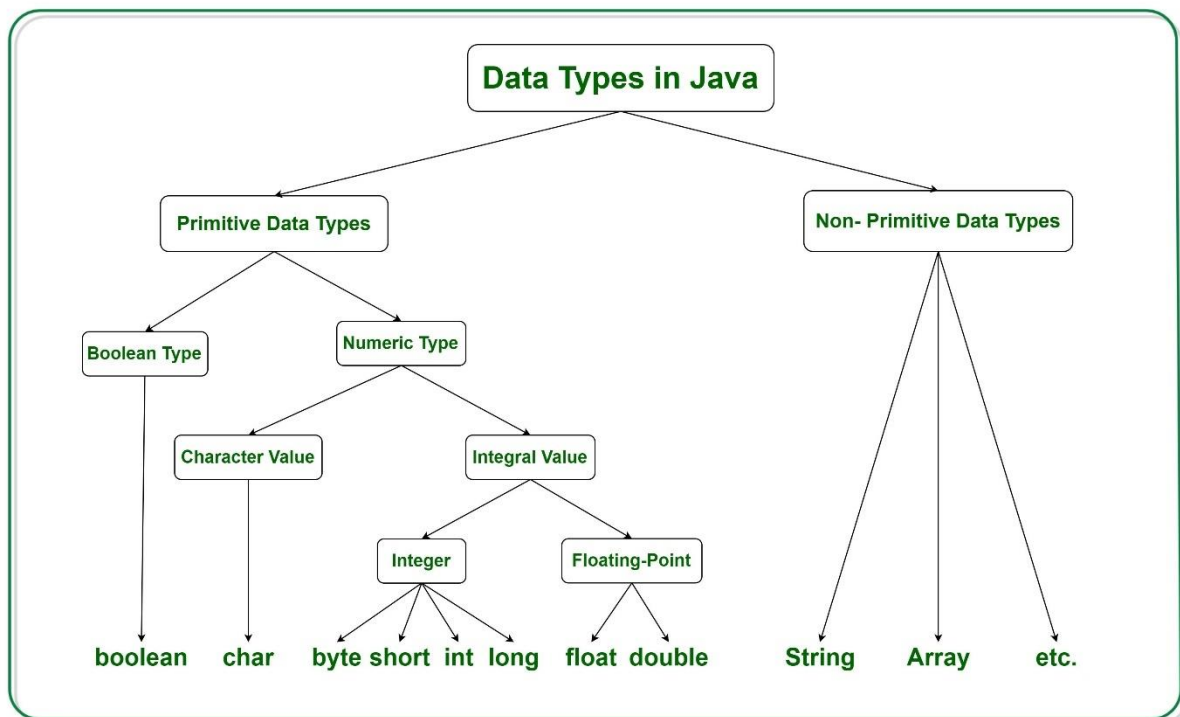
- The primitive data types include boolean, char, byte, short, int, long, float and double.
- Have fixed size

### 2. **Non-primitive data types:**

- The non-primitive data types include Classes, Interfaces, and Arrays.
- Not have fixed size

## Java Primitive Data Types

In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language.



There are 8 types of primitive data types:

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

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

## 1. Boolean Data Type

- The Boolean data type is used to store only two possible values: true and false.
- This data type is used for simple flags that track true/false conditions.

- The Boolean data type specifies one bit of information, but its "size" can't be defined precisely.

**Example:** Boolean one = **false**

## 2. Byte Data Type

- The byte data type is an example of primitive data type.
- Its value-range lies between -128 to 127
- Its minimum value is -128 and maximum value is 127. Its default value is 0.
- The byte data type is used to save memory in large arrays where the memory savings is most required.

**Example:** **byte** a = **10**, **byte** b = **-20**

## 3. Short Data Type

- Its value-range lies between -32,768 to 32,767
- Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0.
- The short data type can also be used to save memory just like byte data type.

**Example:** **short** s = **10000**, **short** r = **-5000**

## 4. Int Data Type

- Its value-range lies between - 2,147,483,648 ( $-2^{31}$ ) to 2,147,483,647 ( $2^{31} - 1$ )
- Its minimum value is - 2,147,483,648 and maximum value is 2,147,483,647. Its default value is 0.
- The int data type is generally used as a default data type for integral values unless if there is no problem about memory.

**Example:** **int** a = **100000**, **int** b = **-200000**

## 5. Long Data Type

- Its value-range lies between  $-9,223,372,036,854,775,808(-2^{63})$  to  $9,223,372,036,854,775,807(2^{63} - 1)$ .
- Its minimum value is  $-9,223,372,036,854,775,808$  and maximum value is  $9,223,372,036,854,775,807$ . Its default value is 0.
- The long data type is used when you need a range of values more than those provided by int.

**Example:** `long a = 100000L, long b = -200000L`

## 6. Float Data Type

- The float data type is a single-precision 32-bit IEEE 754 floating point.
- Its value range is unlimited.
- It is recommended to use a float (instead of double) if you need to save memory in large arrays of floating point numbers.
- The float data type should never be used for precise values, such as currency.
- Its default value is 0.0F.

**Example:** `float f1 = 234.5f`

## 7. Double Data Type

- The double data type is a double-precision 64-bit IEEE 754 floating point.
- Its value range is unlimited.
- The double data type is generally used for decimal values just like float.
- The double data type also should never be used for precise values, such as currency. Its default value is 0.0d.

**Example:** `double d1 = 12.3`

## 8. Char Data Type

- The char data type is a single 16-bit Unicode character.
- Its value-range lies between `'\u0000'` (or 0) to `'\uffff'` (or 65,535).



- The char data type is used to store characters.

**Example:** `char` letterA = 'A'

Why char uses 2 byte in java and what is \u0000 ?

It is because java uses Unicode system not ASCII code system. The \u0000 is the lowest range of Unicode system. To get detail explanation about Unicode visit next page.

## 9. String

- String is a sequence of characters. But in Java, string is an object that represents a sequence of characters.
- A String variable contains a collection of characters surrounded by double quotes:

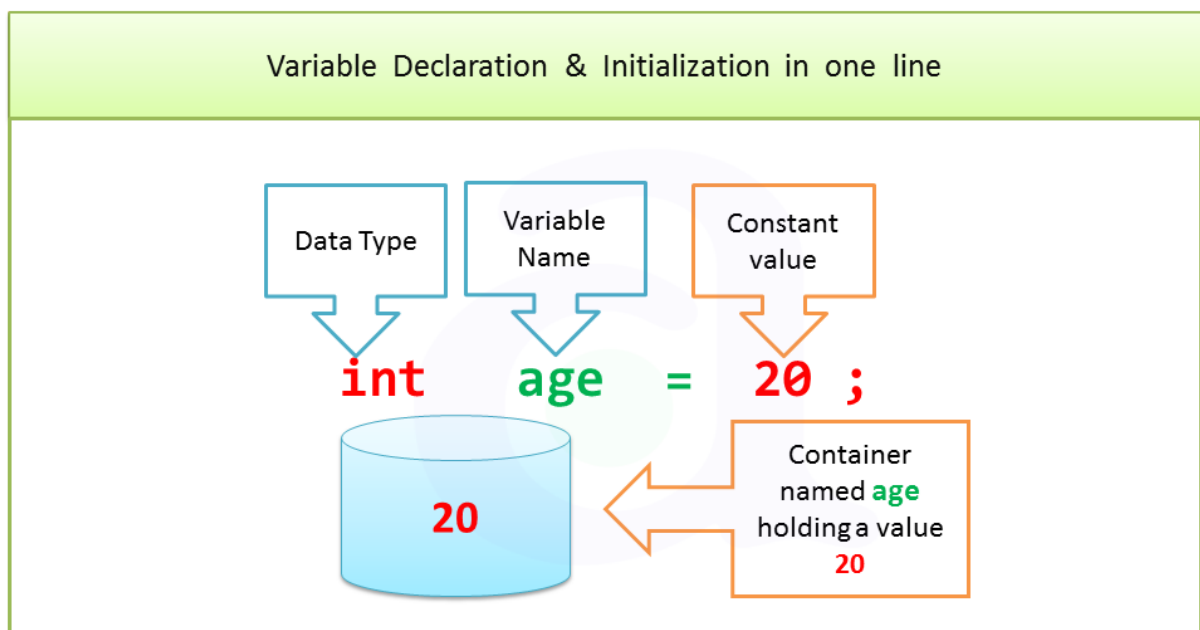
Eg. String s="Welcome To Velocity";

=====

## Variables

- Variable is nothing but piece of memory used to store information.
- A variable is a name given to a memory location. It is the basic unit of storage in a program.
- A variable is a container which holds the value while the Java program is executed. A variable is assigned with a data type.
  1. According to all programming language we cannot declare information directly, so variables are introduced to declare information and to store it.
  2. In java programming variable can't be declare directly to store info so we have declared datatype before it.
  3. **Reusable** – It help us the info will use again and again

### How to declare variables?

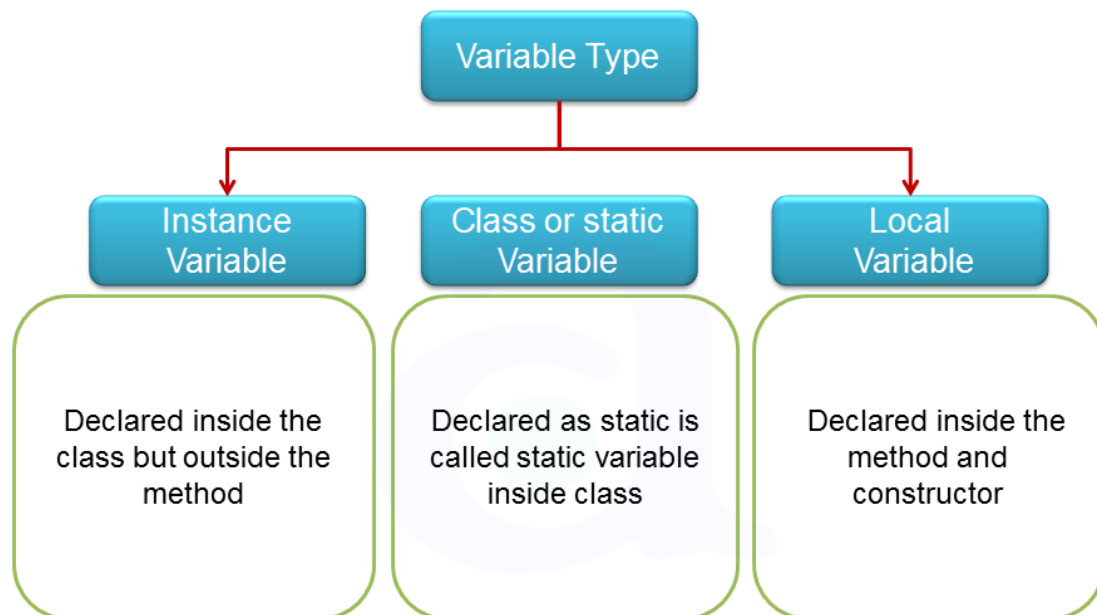


**datatype variable = information;**

- ❖ **datatype:** Type of data that can be stored in this variable.
- ❖ **Variable name:** Name given to the variable.
- ❖ **value:** It is the initial value stored in the variable.

## Types of Variable

1. Local Variable
2. Static Variable/Class Variable
3. Global Variable/Instance Variable



### 1. Local Variable

- The variables which are declared inside method body, method block or constructor called as local variables
- We can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
- These variables are temporary variable
- It cannot be defined by static keyword
- Initialization of the local variable is mandatory before using it

### 2. Static Variable

- Static variables also known as class variable
- Static variables are declared in class body but outside of method/block with static keyword before it.
- We have only one copy of it.

- Initialization of static variable is not necessary
- Static variables are directly accessible to any method
- We have default values

int, byte, short, long	0
float double	0.0
String	null
boolean	false

### 3. Global Variable

- Also called as Instance/non-static variable
- Global variables are declared in class body but outside of method/block
- Initialization of global variable is not necessary
- Global variables are only accessible in non-static method, not able to access in static method. But if we want to call in main method we can call it with object.
- We have default values as

int, byte, short, long	0
float double	0.0
String	null
boolean	false

=====

## Constructor

- =====
- Constructor are used to initialize of data members(variables) of class and to load non-static member/methods into object
  - Constructor is a block of code similar to method
  - Constructors are special member of class
  - All classes have constructors, whether you define one or not, because Java automatically provides a default constructor that initializes all member variables to default values. However, once you define your own constructor, the default constructor is no longer used.
  - At the time of constructor declaration below are points need to followed
    - Constructor name should be same as class name
    - We should not declare any return type for Constructor
    - Any number of Constructor can be declared in class but name should same as class name but having different argument.

### Ways to call a constructor

1. By creating object of a class
  - Constructor executes automatically when we create an object.
2. By using 'new' keyword with constructor name with method signature followed by semicolon

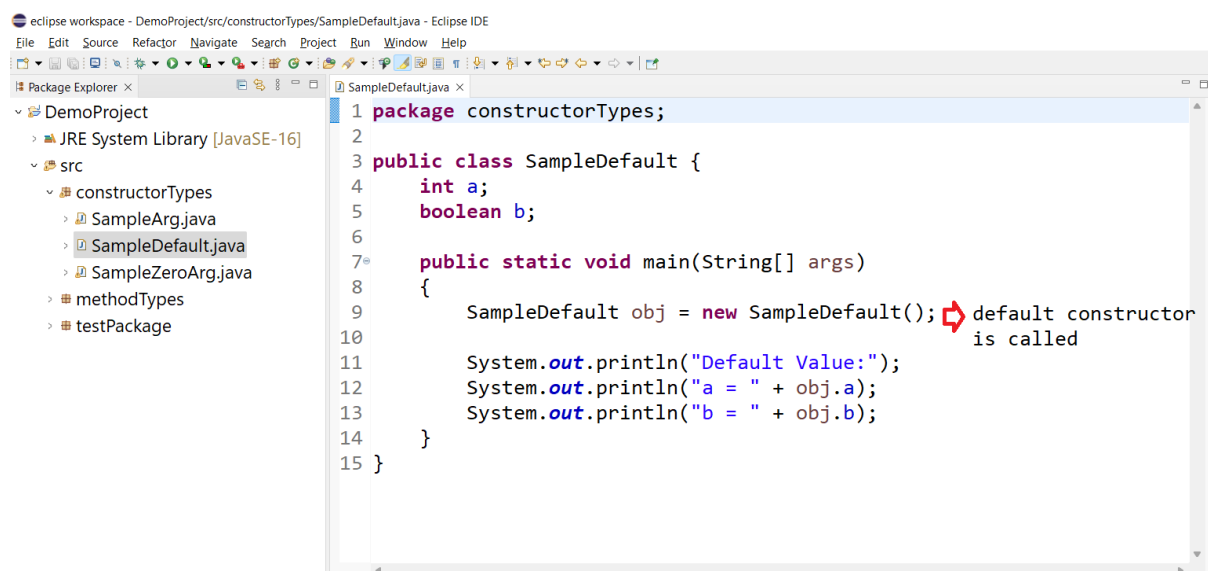
**new ConstructorName();**

### There are 2 types of constructor

1. Default Constructor
2. User defined Constructor
  - a. Zero Argument/ Non Argument Constructor
  - b. Parameterized/ Argument Constructor

## 1. Default Constructor

- If Constructor is not declared in class, then at the time compilation compiler will provide/consider the constructor.
- If we do not create any constructor, the Java compiler automatically create a no-arg constructor during the execution of the program. This constructor is called default constructor.
- It initializes all member variables to their default values
- The best example of default constructor is main method



```
1 package constructorTypes;
2
3 public class SampleDefault {
4     int a;
5     boolean b;
6
7     public static void main(String[] args)
8     {
9         SampleDefault obj = new SampleDefault(); default constructor
10                                           is called
11
12         System.out.println("Default Value:");
13         System.out.println("a = " + obj.a);
14         System.out.println("b = " + obj.b);
15     }
16 }
```

## 2. User defined constructor

- If programmer declared constructor in Java class, then it is considered to be user defined constructor
- User defined constructor is divided into two types
  1. Zero argument constructor and
  2. parameterised constructor

### 1. Zero argument constructor

- It can be used to initialise data member of a class
- Constructor with no argument is known as zero argument constructor or non-argument constructor

```
1 package constructorTypes;
2
3 public class SampleZeroArg {
4
5     static int num;           // declaration
6     String name;
7
8     SampleZeroArg()
9     {
10         num=20;
11         name="Ganesh";        // initiation
12
13         System.out.println(num); // usage
14         System.out.println(name);
15     }
16     public static void main(String args[])
17     {
18         SampleZeroArg xyz = new SampleZeroArg();
19     }
20 }
21
```

## 2. Parametrize constructor

- Constructor with arguments in signature bracket is called as parameterised constructor
- We can declare multiple constructors having different arguments (distinct parameters) in class

```
1 package constructorTypes;
2
3 public class SampleArg {
4
5     String city;
6     SampleArg()
7     {
8         city="Pune";
9         System.out.println(city);
10    }
11
12    SampleArg(int a)
13    {
14        System.out.println(a);
15    }
16
17    SampleArg(String name)
18    {
19        System.out.println(name);
20    }
21
22    public static void main(String args[])
23    {
24        SampleArg obj1=new SampleArg();
25        SampleArg obj2=new SampleArg(852681256);
26        SampleArg obj3=new SampleArg("Shivlila");
27    }
28 }

```

### **Important Notes –**

- ❖ Access scope of constructors will be same as a class access, it means if class is a public then constructor is a public, if class is private then constructor is also private or vice versa
- ❖ Constructors are going to invoke/use at the time of object creation at the time of object creation
- ❖ A constructor cannot be abstract or static or final.
- ❖ A constructor can be overloaded but cannot be overridden.

### **How Constructors are different from Methods in Java?**

- ❖ Constructors must have the same name as the class within which it is defined while it is not necessary for the method in Java.
- ❖ Constructor does not have any return type while method have the return type or void which does not return any value.
- ❖ Constructors are called only once at the time of Object creation while method(s) can be called any number of times.

### **Constructor Overloading -**

- ❖ The class allows to define multiple constructors by providing different types of arguments in simple words constructors with argument is known as parameters constructors where constructor overloading is placed
- ❖ **Def –** In a class if we have multiple constructors with different argument but having same name is known as constructor overloading.

=====



## Operators in Java

- Operators in Java are symbols that are used to perform operations on variables and values.

### Types:-

1. Unary Operator,
2. Arithmetic Operator,
3. Relational Operator,
4. Shift Operator,
5. Bitwise Operator,
6. Logical Operator,
7. Ternary Operator and
8. Assignment Operator

### Java Operator Precedence

Operator	Category	Precedence
Unary	postfix	<i>expr++ expr--</i>
	prefix	<i>++expr --expr</i>
Arithmetic	multiplicative	<i>* / %</i>
	additive	<i>+ -</i>
Shift	shift	<i>&lt;&lt; &gt;&gt; &gt;&gt;&gt;</i>
Relational	comparison	<i>&lt; &gt; &lt;= &gt;=</i>
	equality	<i>== !=</i>
Bitwise	AND	<i>&amp;</i>
	OR	<i> </i>
	XOR	<i>^</i>
Logical	AND	<i>&amp;&amp;</i>
	OR	<i>  </i>
Ternary	ternary	<i>? :</i>
Assignment	assignment	<i>= += -= *= /= %= &amp;= ^=  = &lt;&lt;= &gt;&gt;=</i>

## 1. Java Unary Operator

- The Java unary operators require only one operand.
- Unary operators are used to perform operations like
  - incrementing/decrementing a value by one
  - negating an expression

Operator	Meaning	Work
a++ a--	postfix	Print + operation
++a --a	prefix	Operation + print

## 2. Java Arithmetic Operators

- Java arithmetic operators are used to perform addition, subtraction, multiplication, and division.
- They act as basic mathematical operations.

Operator	Meaning	Work
+	Addition	To add two operands.
-	Subtraction	To subtract two operands.
*	Multiplication	To multiply two operands.
/	Division	To divide two operands.
%	Modulus	To get the area of the division of two operands.

## 3. Relational operators

- The **Java Relational operators** compare between operands and determine the relationship between them.
- The output of the relational operator is (true/false) boolean value

Operator	Meaning
==	Is equal to
!=	Is not equal to
>	Greater than
<	Less than
>=	Greater than or equal to

<=	Less than or equal to
----	-----------------------

#### 4. Shift Operator

- It is used to shift all bits in value to left/right side of specified number in times
- **Left Shift Operator**
  - The Java left shift operator << is used to shift all of the bits in a value to the left side of a specified number of times.
- **Java Right Shift Operator**
  - The Java right shift operator >> is used to move the value of the left operand to right by the number of bits specified by the right operand.

#### 5. Logical Operators

- These operators are used to perform logical "AND", "OR" and "NOT" operation.
- They are used to combine two or more conditions.

Operator	Meaning	Working
&&	AND	True --- All conditions need true False --- any one condition will be false
	OR	True --- any one condition is True False -- all condition false
!	Not	Invert conditions

#### 6. Bitwise Operators

Operator	Meaning	Work
&	AND Operator	True - All condition true False - Any one condition false
	OR Operator	True - any one condition is true False - all condition false
^	XOR Operator	False - Both condition same / T or F True - Have different values of conditions

## 7. Assignment Operators

- The **Java Assignment Operators** are used when you want to assign a value to the expression. The assignment operator denoted by the single equal sign `=`.
- In a Java assignment statement, any expression can be on the right side and the left side must be a variable name.

Operator	Example
<code>=</code>	<code>C = A + B</code> will assign value of <code>A + B</code> into <code>C</code>
<code>+=</code>	<code>C += A</code> is equivalent to <code>C = C + A</code>
<code>-=</code>	<code>C -= A</code> is equivalent to <code>C = C - A</code>
<code>*=</code>	<code>C *= A</code> is equivalent to <code>C = C * A</code>
<code>/=</code>	<code>C /= A</code> is equivalent to <code>C = C / A</code>
<code>%=</code>	<code>C %= A</code> is equivalent to <code>C = C % A</code>
<code>&lt;&lt;=</code>	<code>C &lt;&lt;= 2</code> is same as <code>C = C &lt;&lt; 2</code>
<code>&gt;&gt;=</code>	<code>C &gt;&gt;= 2</code> is same as <code>C = C &gt;&gt; 2</code>
<code>&amp;=</code>	<code>C &amp;= 2</code> is same as <code>C = C &amp; 2</code>
<code>^=</code>	<code>C ^= 2</code> is same as <code>C = C ^ 2</code>
<code> =</code>	<code>C  = 2</code> is same as <code>C = C   2</code>

## 8. Java Ternary Operator

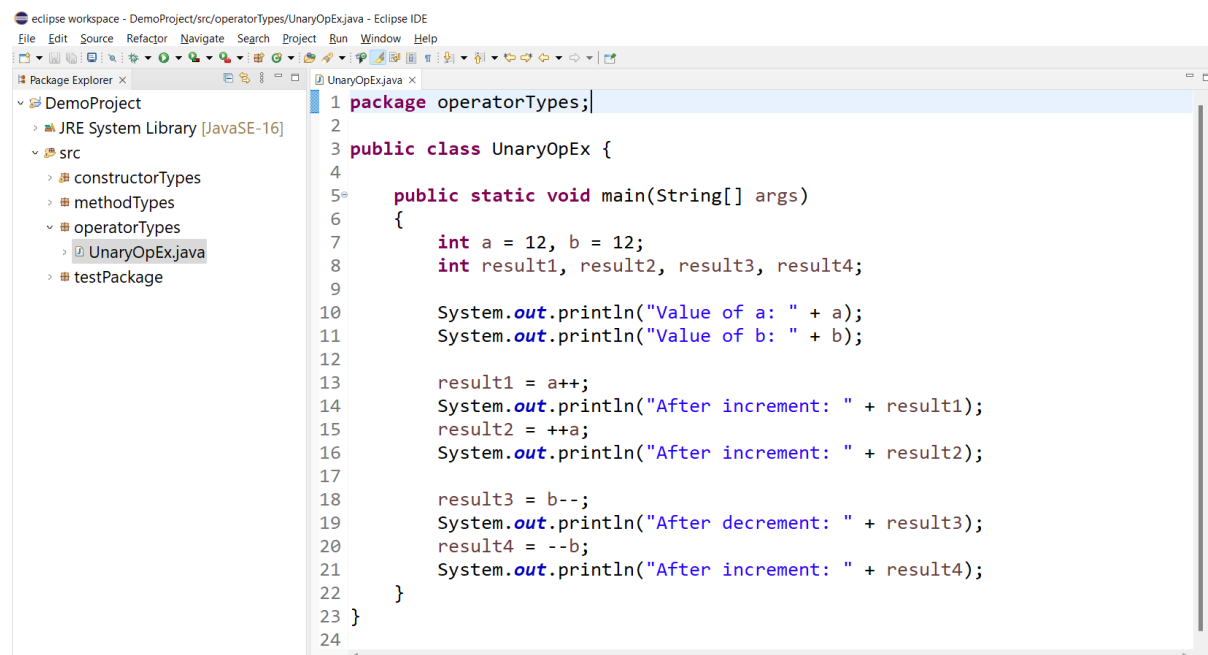
- The ternary operator (conditional operator) is shorthand for the if-then-else statement.

**variable = Expression ? expression1 : expression2**

Here's how it works.

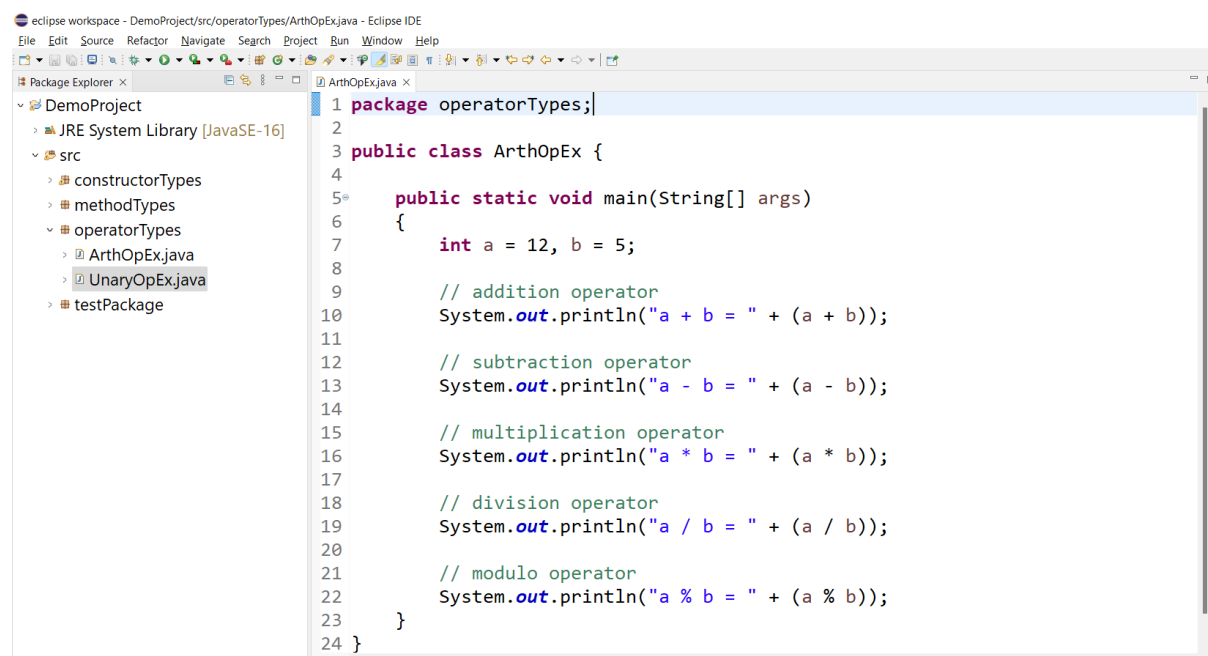
- If the Expression is true, expression1 is assigned to the variable.
- If the Expression is false, expression2 is assigned to the variable

## Unary Operator



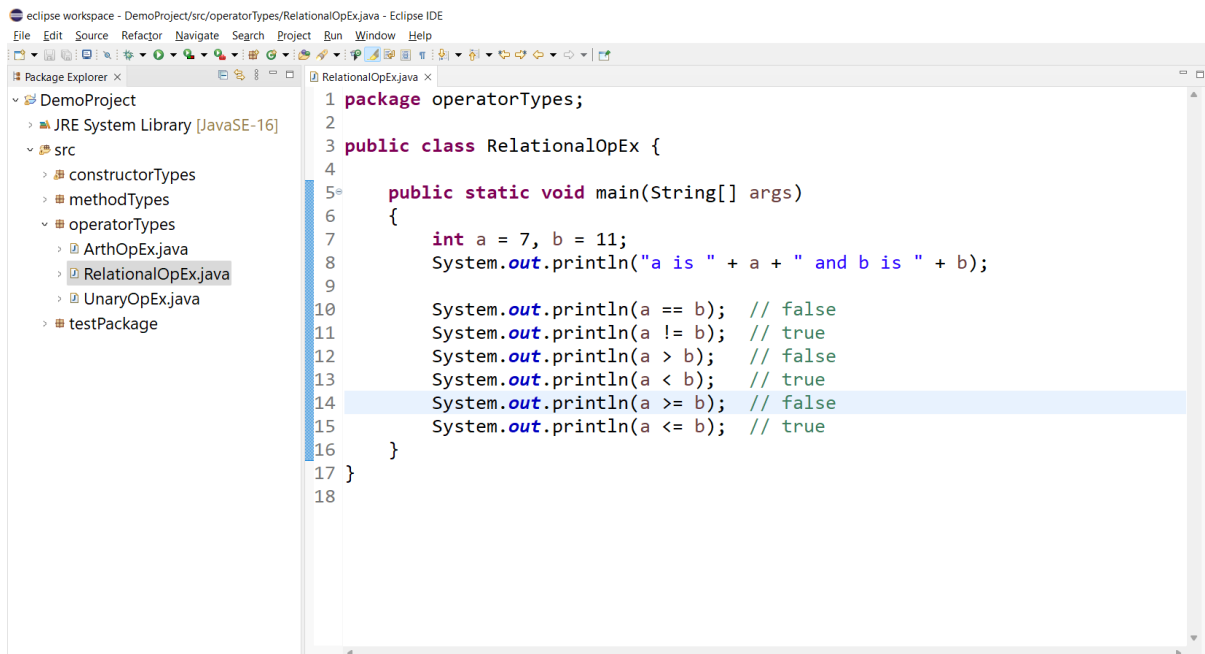
```
1 package operatorTypes;
2
3 public class UnaryOpEx {
4
5     public static void main(String[] args)
6     {
7         int a = 12, b = 12;
8         int result1, result2, result3, result4;
9
10        System.out.println("Value of a: " + a);
11        System.out.println("Value of b: " + b);
12
13        result1 = a++;
14        System.out.println("After increment: " + result1);
15        result2 = ++a;
16        System.out.println("After increment: " + result2);
17
18        result3 = b--;
19        System.out.println("After decrement: " + result3);
20        result4 = --b;
21        System.out.println("After decrement: " + result4);
22    }
23 }
24
```

## Arithmetic Operator



```
1 package operatorTypes;
2
3 public class ArthOpEx {
4
5     public static void main(String[] args)
6     {
7         int a = 12, b = 5;
8
9         // addition operator
10        System.out.println("a + b = " + (a + b));
11
12        // subtraction operator
13        System.out.println("a - b = " + (a - b));
14
15        // multiplication operator
16        System.out.println("a * b = " + (a * b));
17
18        // division operator
19        System.out.println("a / b = " + (a / b));
20
21        // modulo operator
22        System.out.println("a % b = " + (a % b));
23    }
24 }
```

## Relational Operator



```
1 package operatorTypes;
2
3 public class RelationalOpEx {
4
5     public static void main(String[] args)
6     {
7         int a = 7, b = 11;
8         System.out.println("a is " + a + " and b is " + b);
9
10        System.out.println(a == b); // false
11        System.out.println(a != b); // true
12        System.out.println(a > b); // false
13        System.out.println(a < b); // true
14        System.out.println(a >= b); // false
15        System.out.println(a <= b); // true
16    }
17 }
18
```

Logical Operator

Bitwise Operator

Shift Operator

Ternary Operator

Assignment Operator