Sumago Infotech Pvt. Ltd.

Strive With Technology...!

JAVA

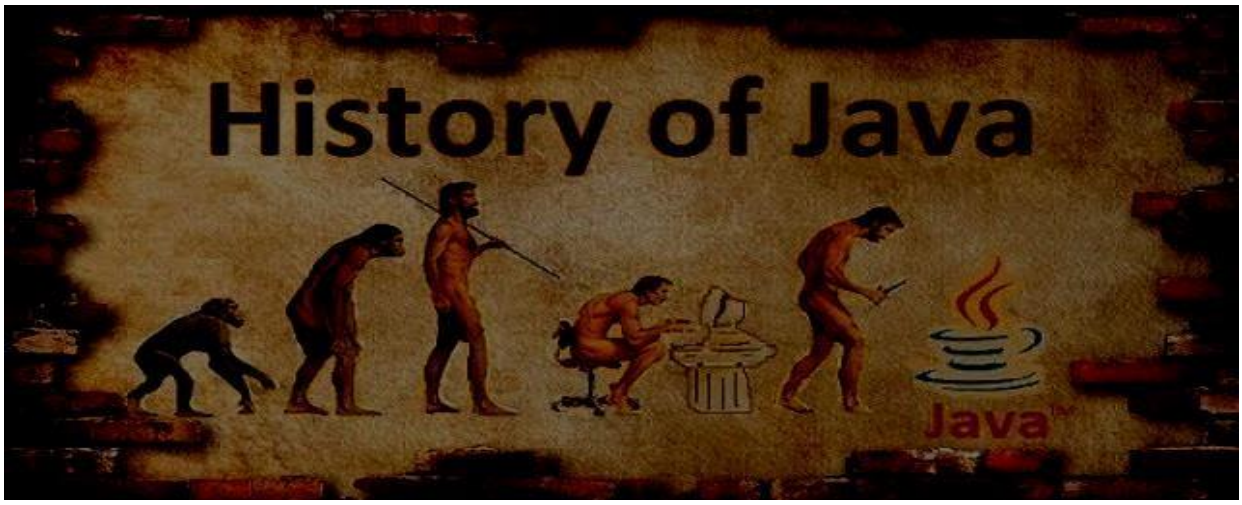
INDEX

- Important Programs
- Oops Concept

1. What is JAVA?

- Java is a widely used object-oriented programming language and platform independent language used 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.

2. History of JAVA:



- In 1991, Sun Microsystem had a requirement and they want to prepare new programming Language.
- By using this "new programming language they want to prepare software's for simple electronic consumer like cable TV switch boxes,

Remote Controls...

- As per requirement Sun Microsystem collect/gather 30 members from their company as team and this team lead by **James Gosling** and

Patrick Naughton. With this 30 members Sun Microsystem started project and name given to this project is **GREEN**.

- Before introducing the new programming language they want to introduce three features in the new language.

What are these three features?

1. Simple programming language. (means more performance)
2. Tightly coded language. (means less cost product)
3. Architectural Neutral programming language. (means low maintenance product)

Finally, Sun Microsystem started preparing their own programming language. They took almost 2 years to prepare Solution for above three features.

3. Why it is called JAVA?

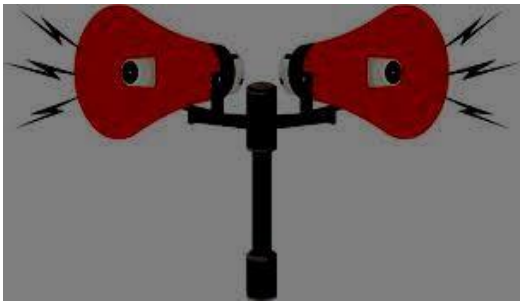
- In 1992, December month Sun Microsystem prepare new programming language and want to give name to new programming language. James Gosling everyday sited in cabin and he always like to see type of tree through his cabin window and the name of tree is "OAK". But Sun Microsystem rejected name because OAK is already existed programming language that time.

- Meanwhile, preparing the new programming language "**GREEN**"" project people used to go Cafe at Sun Microsystem. They used to take

one brand label coffee and name of Coffee is **"JAVA",** everyone decided and given the same coffee name JAVA to new programming language.

- JAVA is one of place name in **Indonesia** that place famous for coffee and they provided the coffee powder with name JAVA. Unfortunately this JAVA coffee going at Sun Microsystem. Because of that our new programming language is JAVA.

4. When JAVA introduced?



- The development of Java language is started in mid-1991. The exact date is not available but it's in the month of June originally the language was to be used for Interactive Televisions. But digital cable television Industries in 1991 found this language more advanced for them. James Gosling, Mike Sheridan, and Patrick Naughton were the key persons in the development of JAVA. **James Gosling, who is also known as the Founder of Java**, started "THE GREEN PROJECT" that led to the release of the language in **1995**

- People come to know Sun Microsystem created new language JAVA and popularity is more in market after that they decided to introduced JAVA language in market as Open Source on date 1996 January 23rd with 1st version name as: **JDK 1.0**

5. JAVA Versions:

- 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.

Managed by Sun Microsystem: (JAVA Born in Sun Microsystem)

JDK Alpha and Beta (1995)
JDK 1.0 (23rd Jan 1996)
JDK 1.1 (19th Feb 1997)
J2SE 1.2 (8th Dec 1998)
J2SE 1.3 (8th May 2000)
J2SE 1.4 (6th Feb 2002)
J2SE 5.0 (30th Sep 2004)
Java SE 6 (11th Dec 2006)

Managed by Oracle Corporation (JAVA growing in Oracle now)

Java SE 7 (28th July 2011)
Java SE 8 (18th Mar 2014)
Java SE 9 (21st Sep 2017)
Java SE 10 (20th Mar 2018)
Java SE 11 (September 2018)
Java SE 12 (March 2019)
Java SE 13 (September 2019)
Java SE 14 (Mar 2020)
Java SE 15 (September 2020)
Java SE 16 (Mar 2021)
Java SE 17 (September 2021)
Java SE 18 (to be released by March 2022)

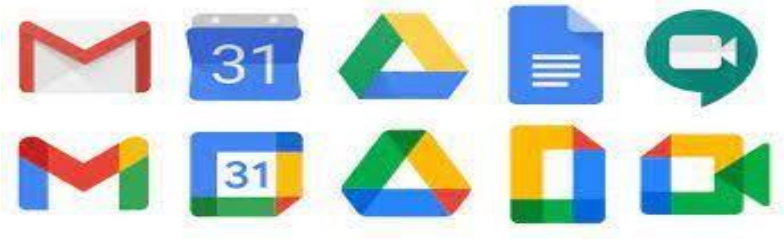
- Since Java SE 8 release, the Oracle Corporation follows a pattern in which every even version is release in March month and an odd version released in September month.

6. How Java is WORA?

- In traditional programming languages like C, C++ when programs were compiled, they Used to be converted into the code understood by the particular underlying hardware, so If We try to run the same code at another machine with different hardware, which understands different code will cause an error, so you have to re-compile the code to be understood by the new hardware.

- In Java, the program is not converted to code directly understood by Hardware, rather it is converted to bytecode(class file), which is interpreted by JVM, so once compiled it generates bytecode file, which can be run anywhere (any machine) which has JVM(Java Virtual Machine) and hence it gets the nature of Write Once and Run Anywhere.

Application



Application is a package to perform specific task.

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

- Desktop Applications such as acrobat reader, media player, antivirus, etc.
- Web Applications such as irctc.co.in, javatpoint.com, etc.
- Enterprise Applications such as banking applications.
- Mobile
- Embedded System
- Smart Card
- Robotics
- Games, etc.

Types of Java Applications

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

1. Standalone Application

- 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, Paint, Microsoft office etc.
- AWT and Swing are used in Java for creating standalone applications.

- Standalone application do not required internet

2. Web Application

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

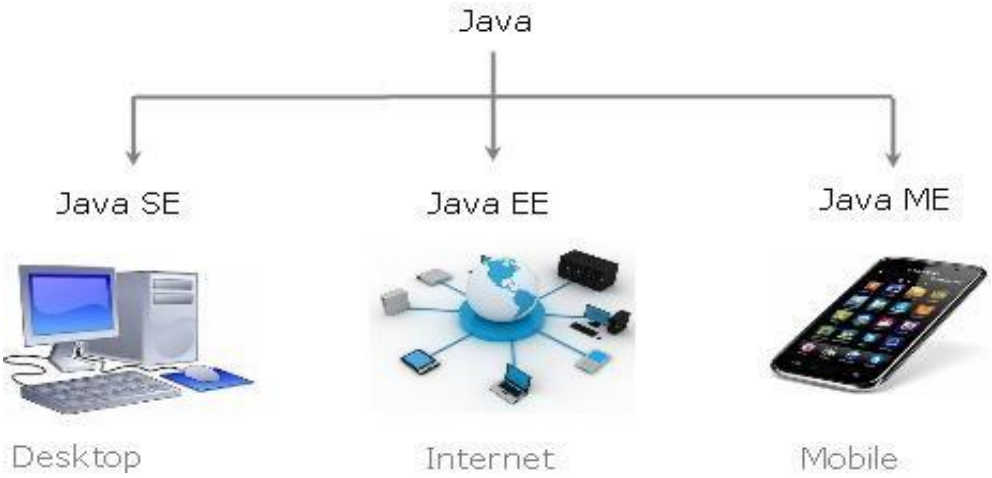
3. Enterprise Application

- An application that is distributed in nature, such as banking applications, etc. is called an enterprise application.
- It has advantages like high-level security, load balancing. In Java, EJB is used for creating enterprise applications.
- Examples of Enterprice applications are Google Pay, PhonePe, Billing System.

4. Mobile Application

- An application which is created for mobile devices is called a mobile application.
- Currently, Android and Java ME are used for creating mobile applications.
- Examples of mobile application are Games like (PubG, Candy crush, Talking Tom etc.,), Share it, Instagram.

Java Classification / Platforms / Editions



1. Java SE (Java Standard Edition)

- It is mainly used for programming of standalone/Desktop applications
- 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.
- It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

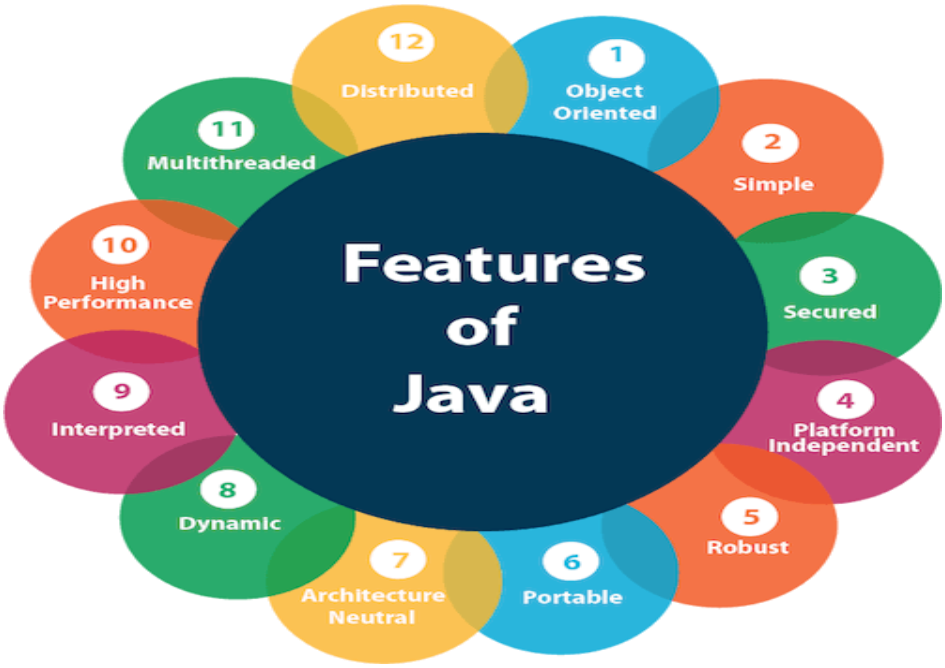
2. Java EE (Java Enterprise Edition)

- It is an enterprise platform that is mainly used to develop web and enterprise applications.
- It is built on top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA, etc.
- Mostly interact with server. Example, banking, e-commerce, telecom appliaction.

3. Java ME (Java Micro Edition)

- It is a micro platform that is dedicated to embeded system development such as mobile applications.

Features of Java



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

1. Simple

- Java is Easy to learn, Java having simple syntax, clean and easy to understand.
- In java no need to remove unreference object because java having automatic garbage collection.

2. Object-Oriented

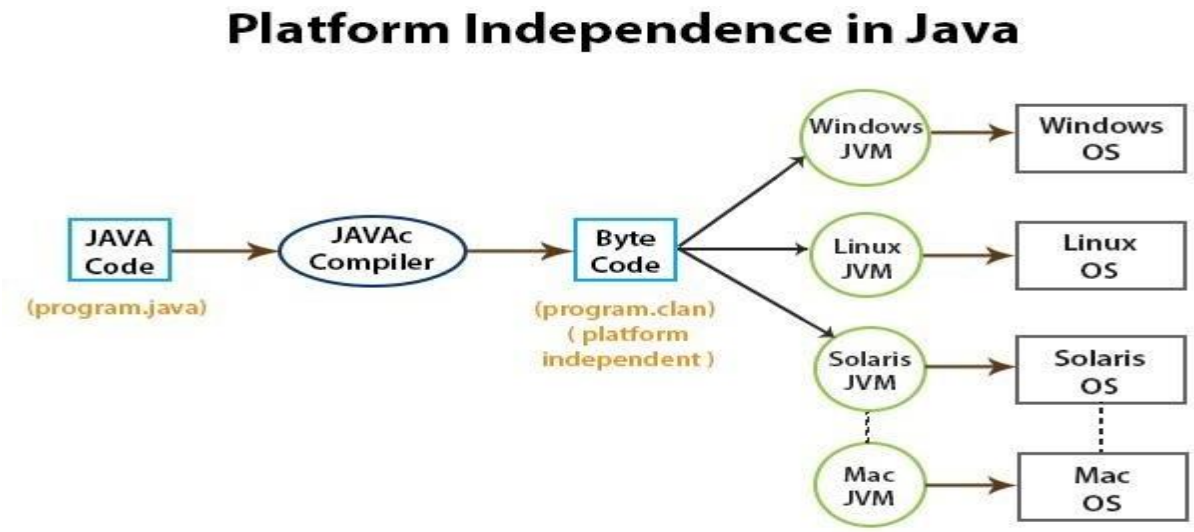
- Java is object oriented programming language, everything in java is a object.
- Object-oriented development includes concepts of Objects and Classes and many more.
- This enables developers to have a wide variety of options for designing their software.

3. Portable

- Java is portable because it facilitates to carry the java byte code to any platform it dosent required any implementation.

4. Platform independent

- Being platform-independent means a program compiled on one machine can be executed on any machine in the world without any change. Java achieves platform independence by using the concept of the BYTE code.
- This is where the "Write Once, run anywhere" (WORA) slogan for Java comes in, which means that we can develop applications on one environment (OS) and run on any other environment without doing any modification in the code.



5. Secured

- Java is the best known for Security.
- By using java we can develop virus free system or software.

6. Robust

- Java is robust as it is capable of handling run-time errors, supports automatic garbage collection and exception handling.
- Java has a strong memory management system. It helps in eliminating errors as it checks the code during both compile and runtime.

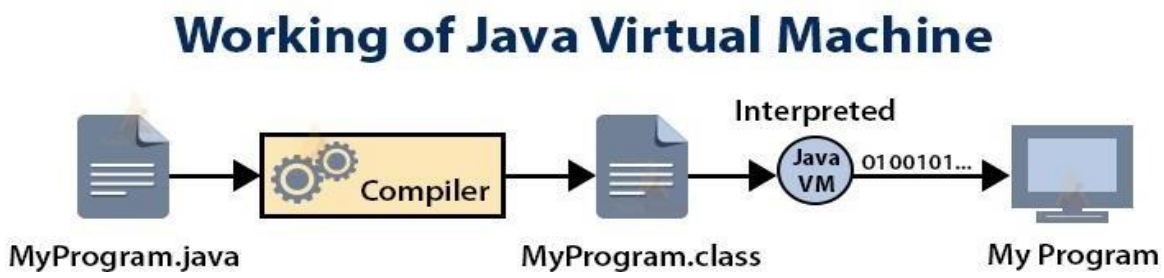
7. Architecture neutral

- Architectural neutral means that the program written on one platform or OS is independent of other platforms or environments and can run on any other Operating System without recompiling them.
- For Example, In C programming, the datatype occupies 2byte for 32-bit architecture and 4byte for 64-bit architecture but in java occupie 4byte for both 32 and 64 bit architecture.

8. Interpreted

- Usually, a computer language can be either compiled or interpreted. Java integrates the power of Compiled Languages with the flexibility of Interpreted Languages.
- Java compiler (javac) compiles the java source code into the bytecode.
- Java Virtual Machine (JVM) then executes this bytecode which is executable on many operating systems and is portable.

The diagram below shows the above process:

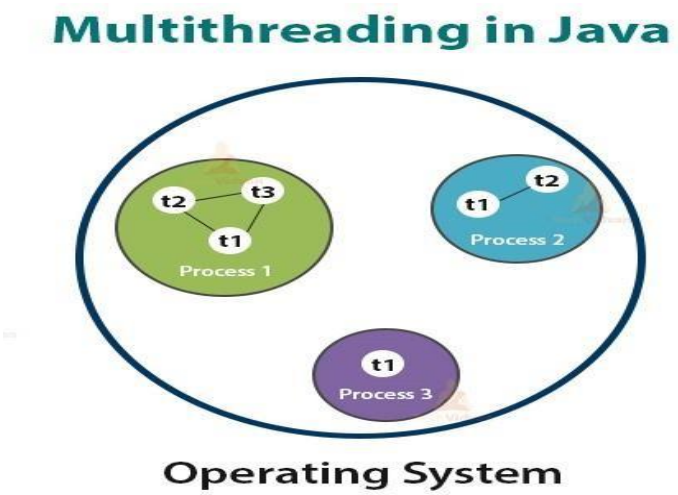


9. High Performance

- Java provides high performance with the use of "JIT – Just In Time compiler", in which the compiler compiles the code on-demand basis, that is, it compiles only that method which is being called. This saves time and makes it more efficient.

10. Multithreaded

- We can write java program that deals with many task at once by defining multiple thread.
- Advantage of multithreaded is that it does not occupie memory of each thread. It shared common memory.



11. Distributed

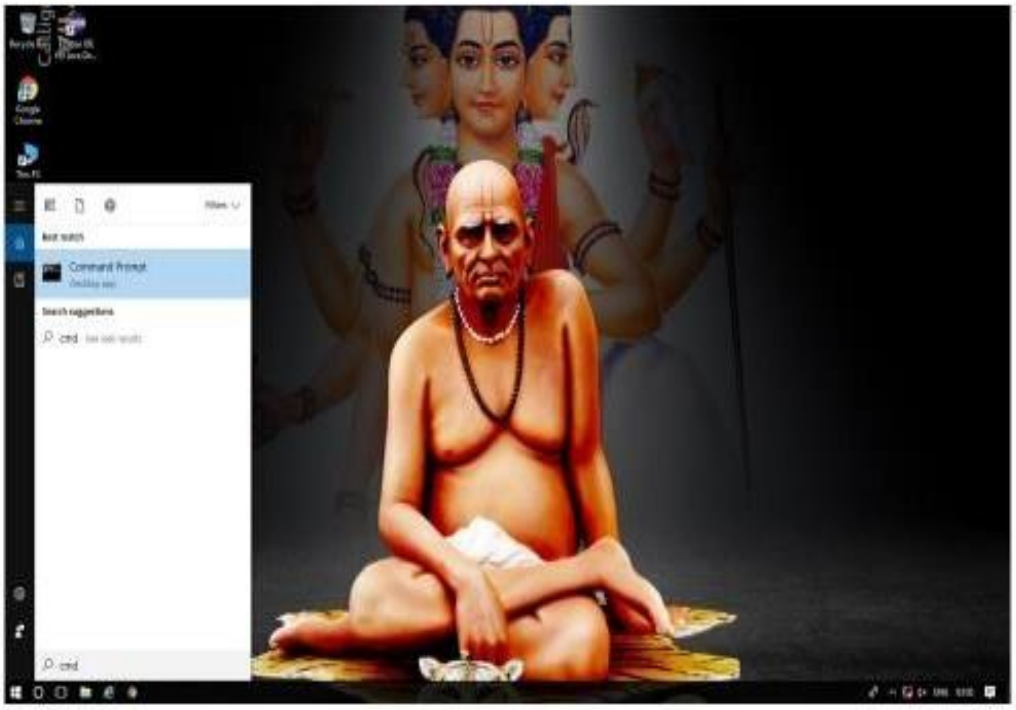
- In Java, we can split a program into many parts and store these parts on different computers. A Java programmer sitting on a machine can access another program running on the other machine.
- This feature in Java gives the advantage of distributed programming, which is very helpful when we develop large projects.

12. Dynamic

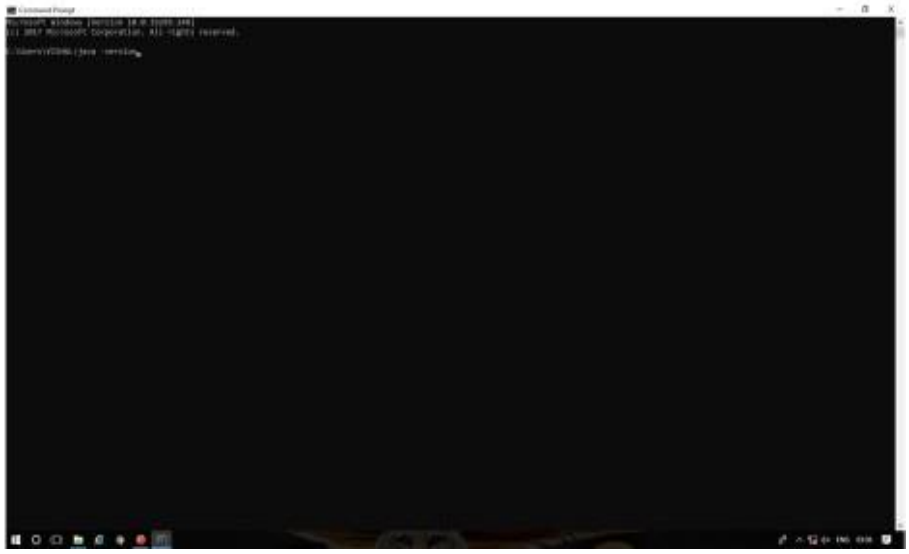
- Java is dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand.

Installation of JDK:

1)Check availability of java in our system for that
go on search box type cmd and click commandpromot



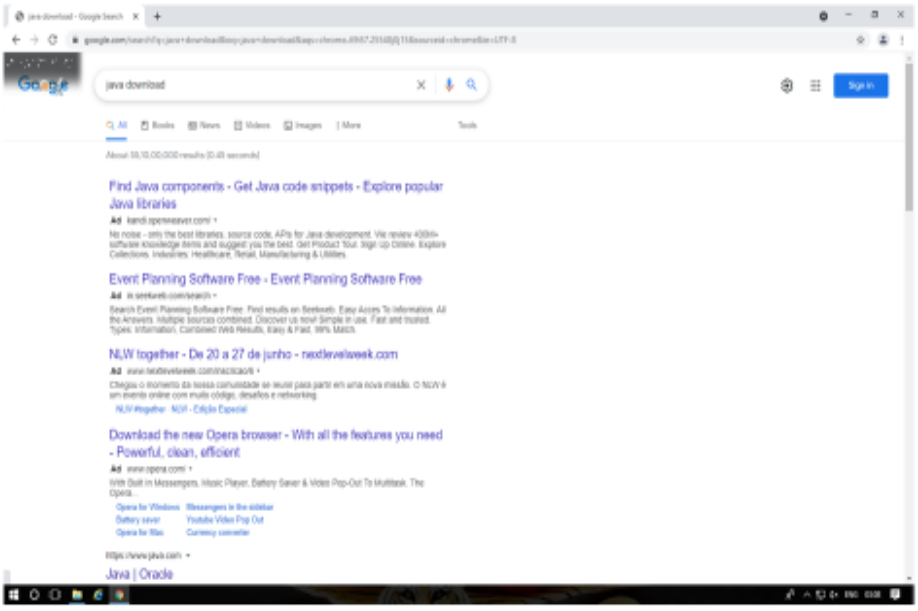
Prompt opens as shown . Now type in command prompt *java -version*



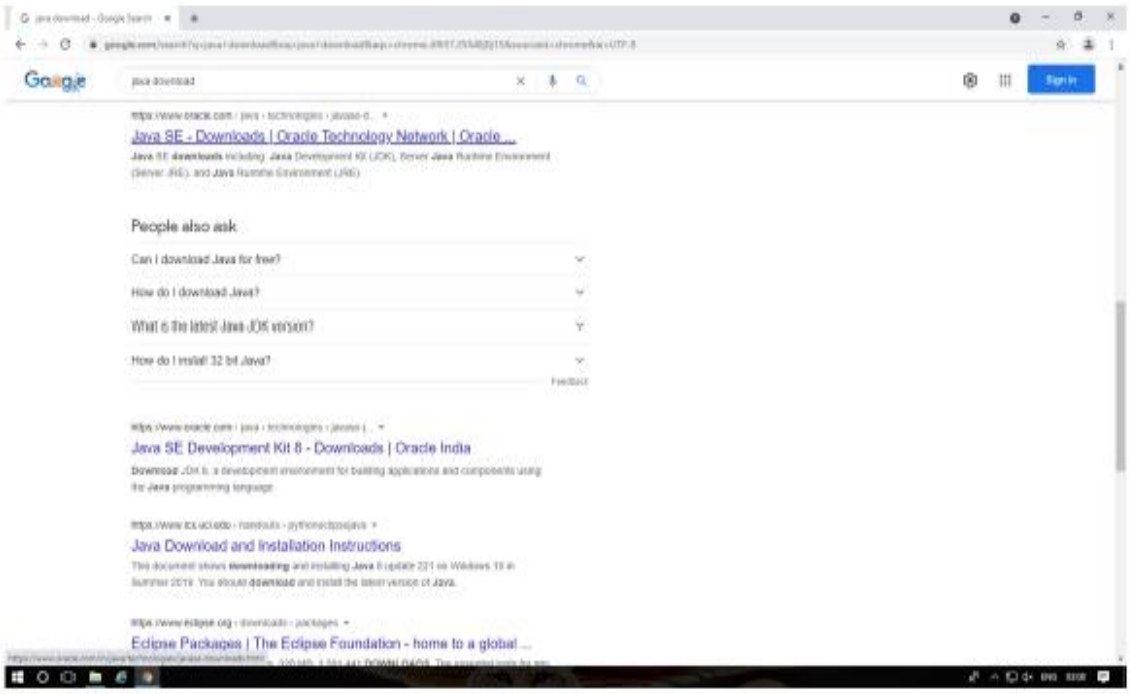
After giving command press enter .if it showing error like *Not Recognized....*

- Then we have not installed java
- We have to be download java from internet.

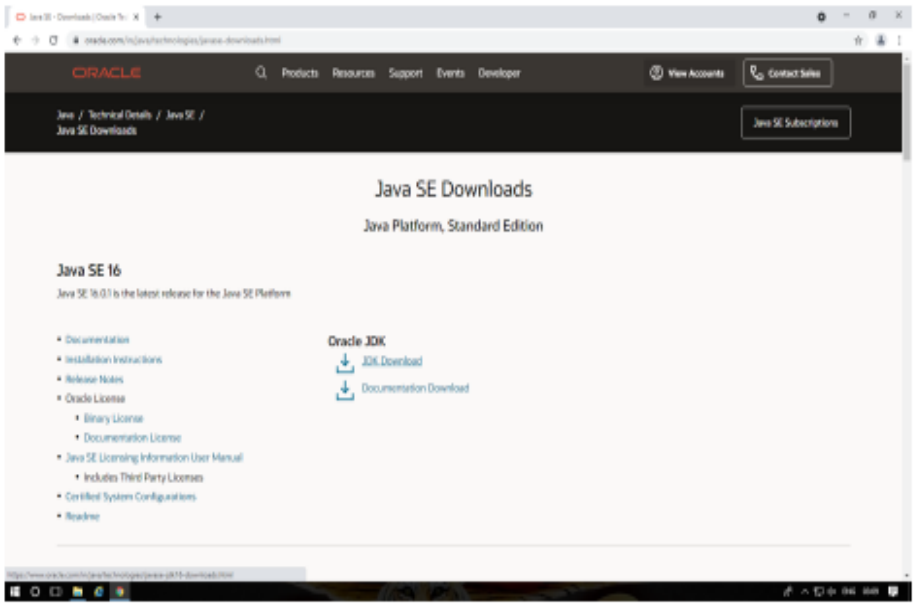
Search on google **Java Download**



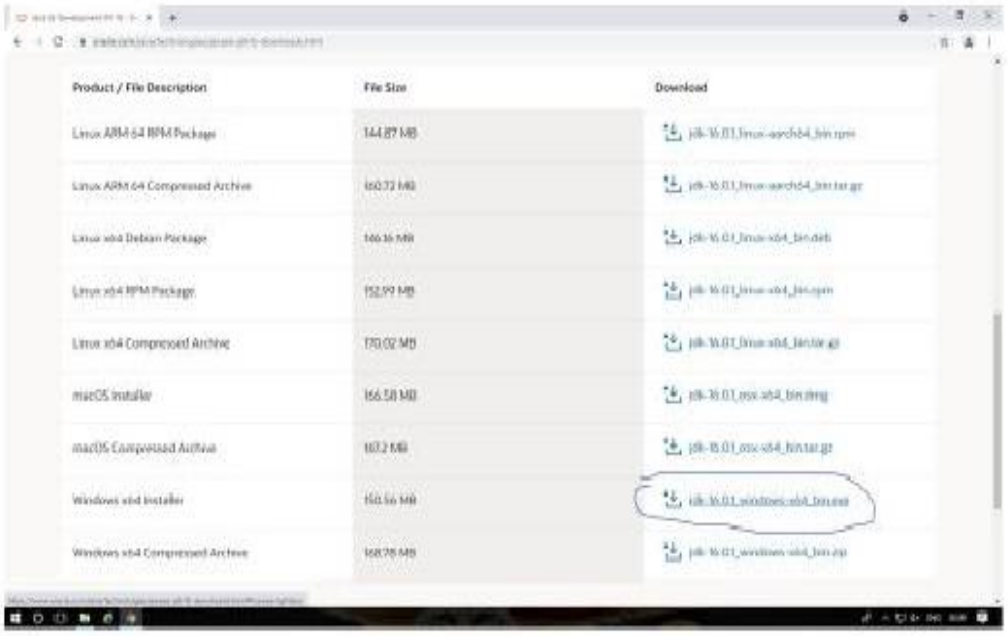
Click on official website of www.Oracle.com as shown



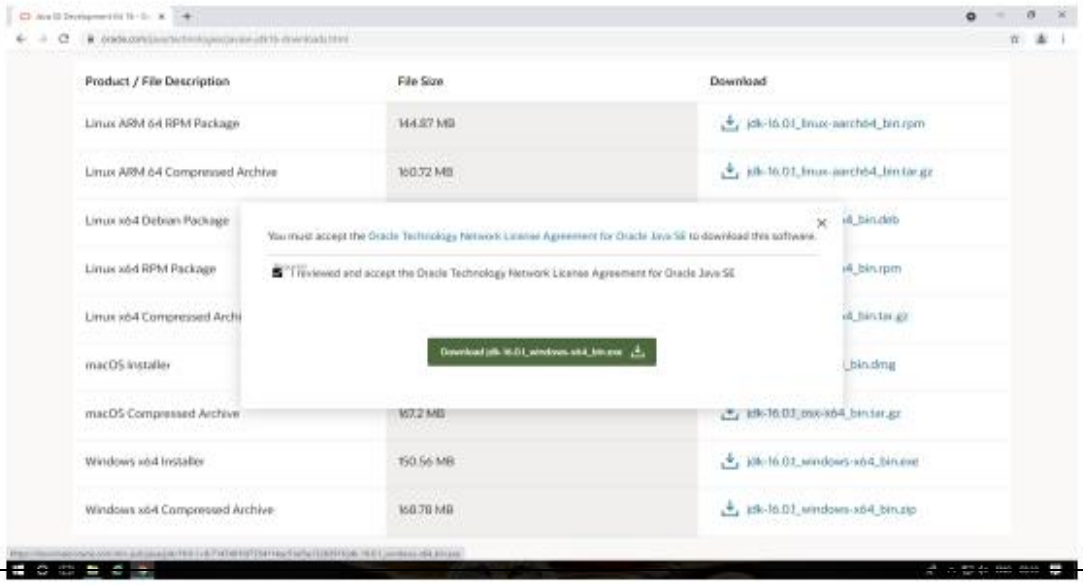
After opening website showing SE 16 latest version . Now click on ***JDK Download***



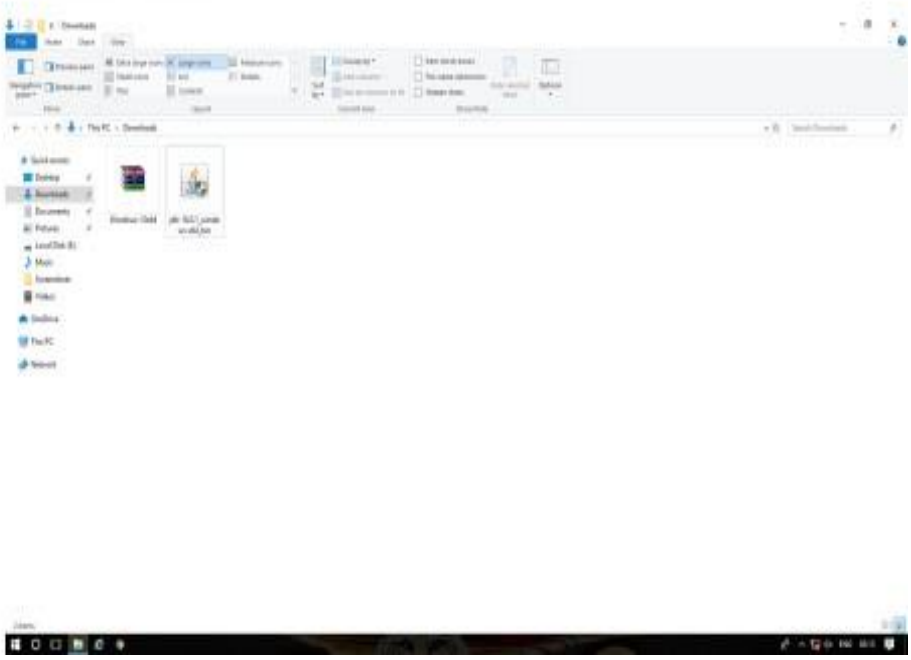
New window opens now click
Windows x64 installer .exe file



New window opens
mark tick in checkbox and click download key



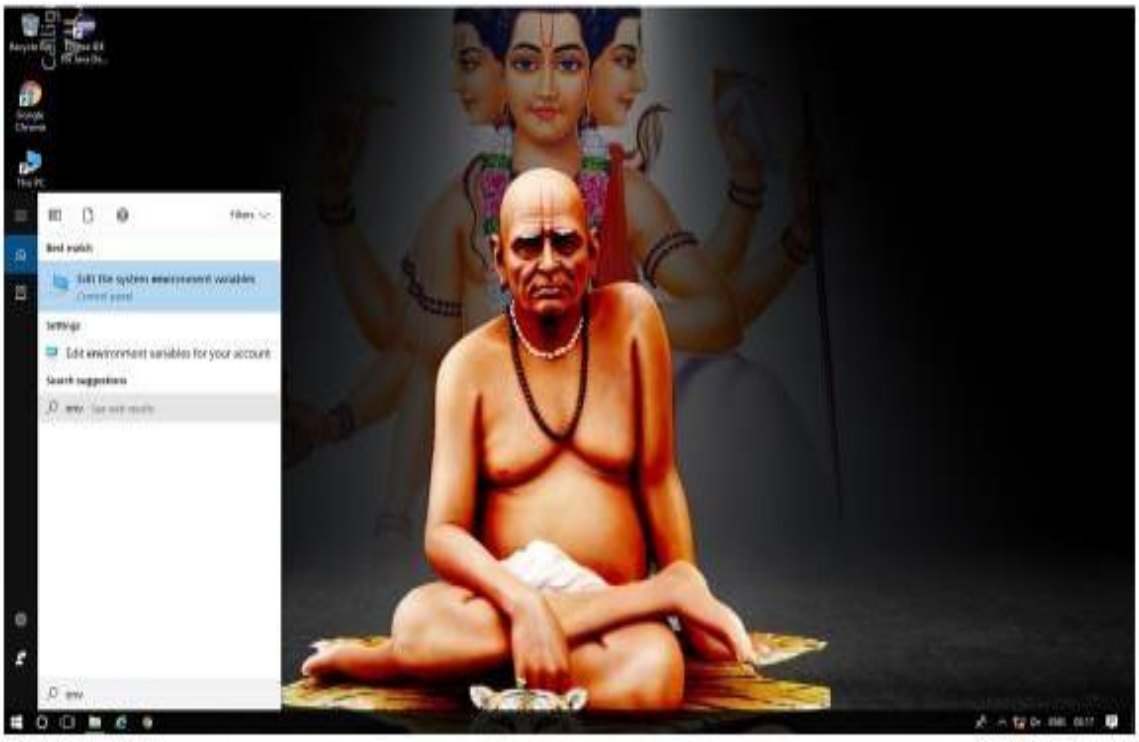
After downloading your file showing in downloads



Now copy that file and paste in your drive D or E

- Don't paste in c
- Now double click on downloaded file
- And simply complete the installation process.

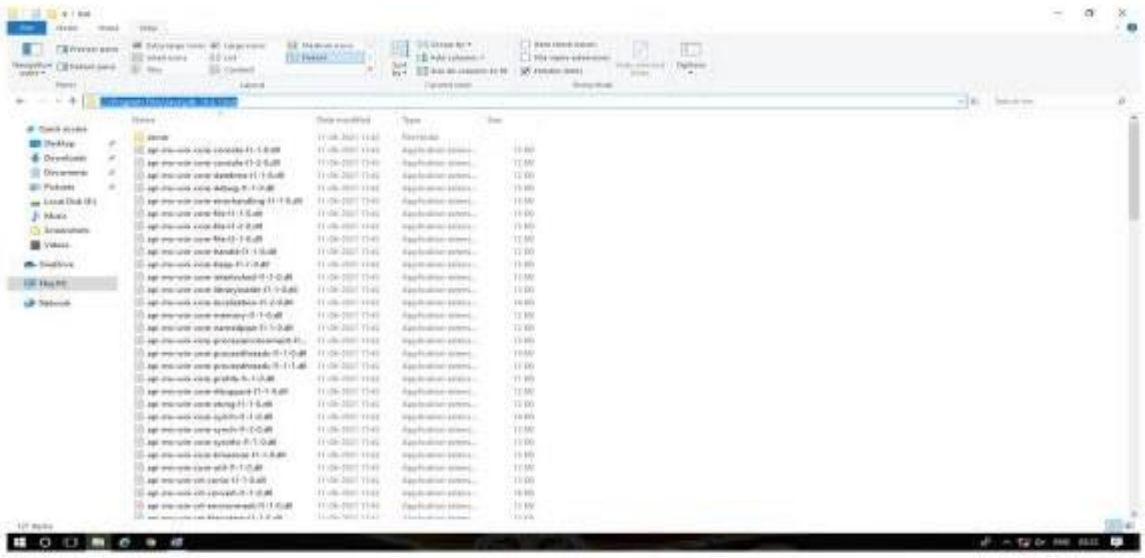
Now we have to set variables for our application
go to search box and type *env* now click "*edit the system environment variables* "



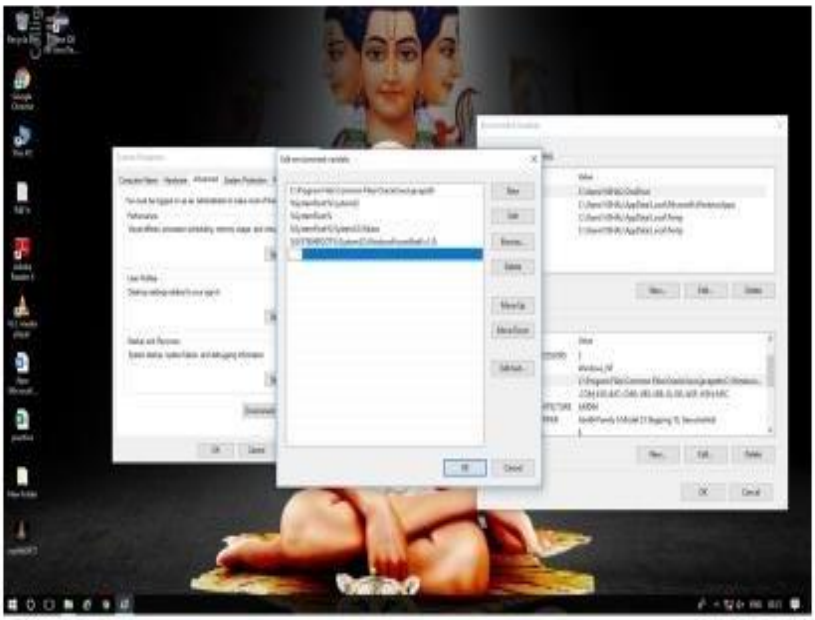
New window opens now click on **Environment variables**

Now in system variable section click on *path* and press *Edit* button

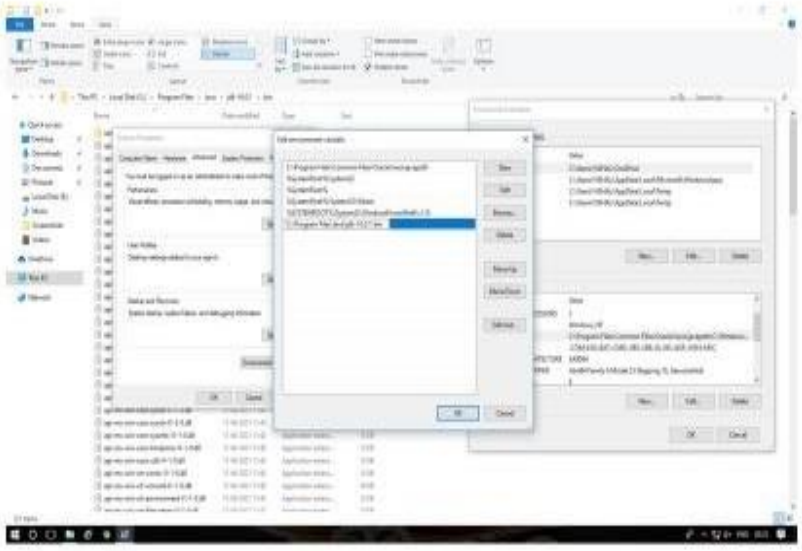
C:\Program Files\Java\jdk-16.0.1\bin
this is our path now copy this path



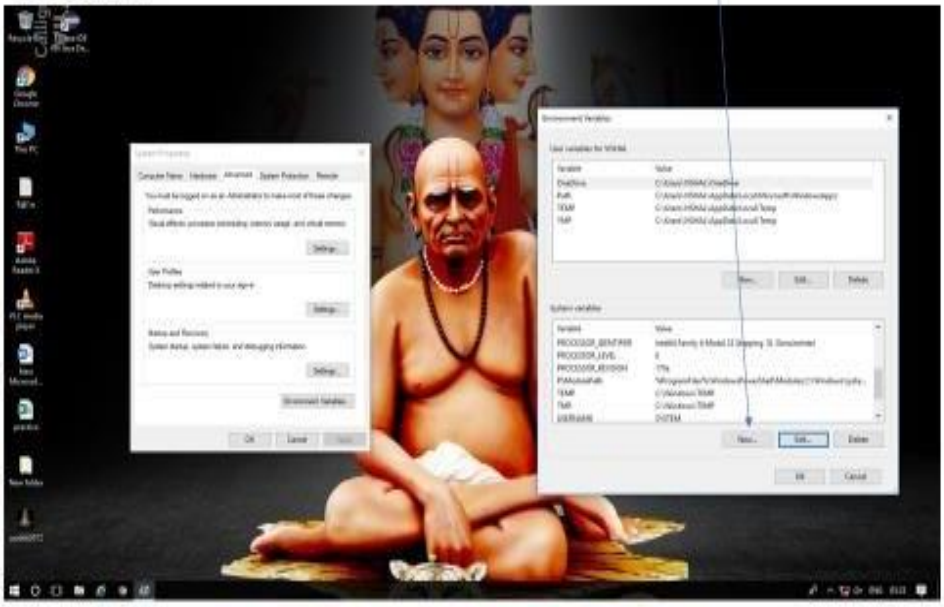
Now click on new button . New row is created. and simply paste the copied path in this section



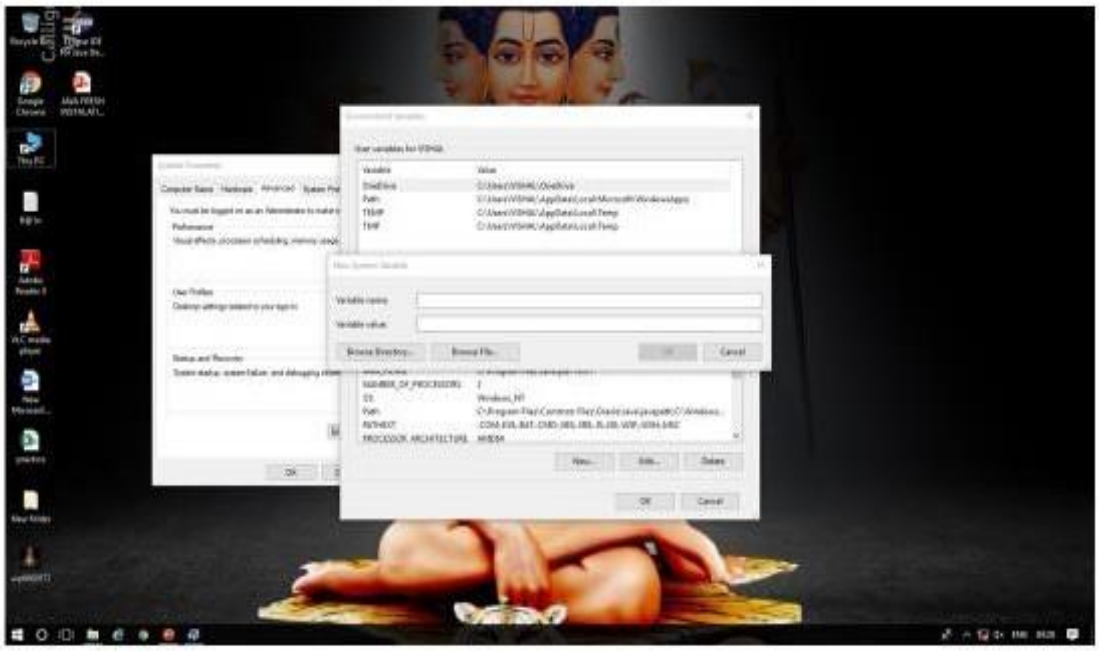
After pasting the path, Now press ok



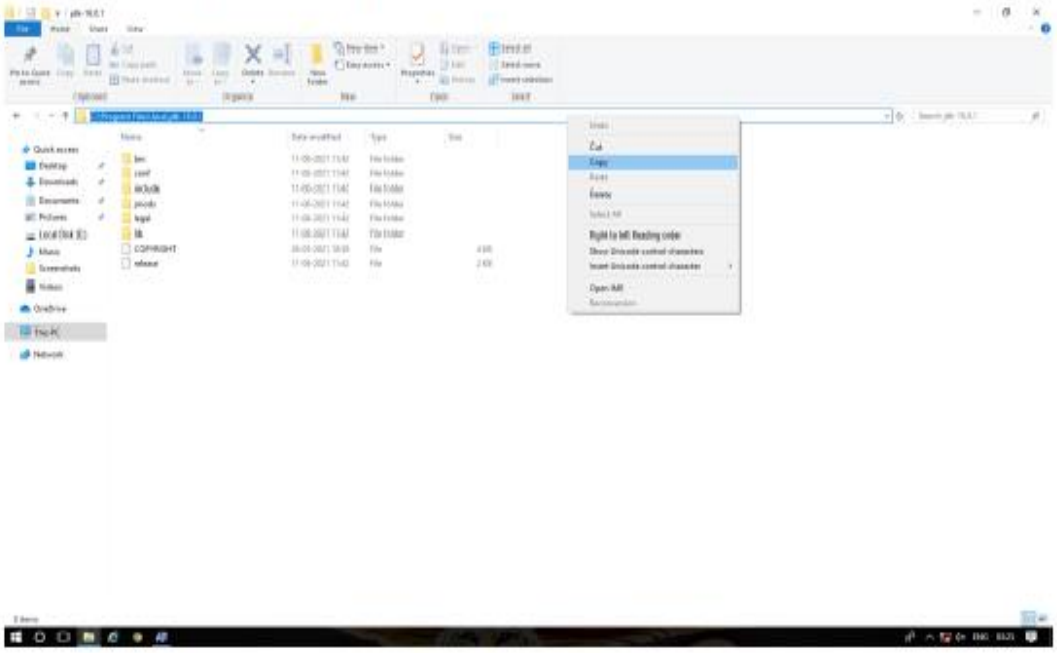
Now find the JAVA_HOME in system variable section if not found then press ***New*** button as shown



New window opens as we press **new**



Now type in vaviable name : JAVA_HOME
In variable value copy the path as shown
C:\Program Files\Java\jdk-16.0.1

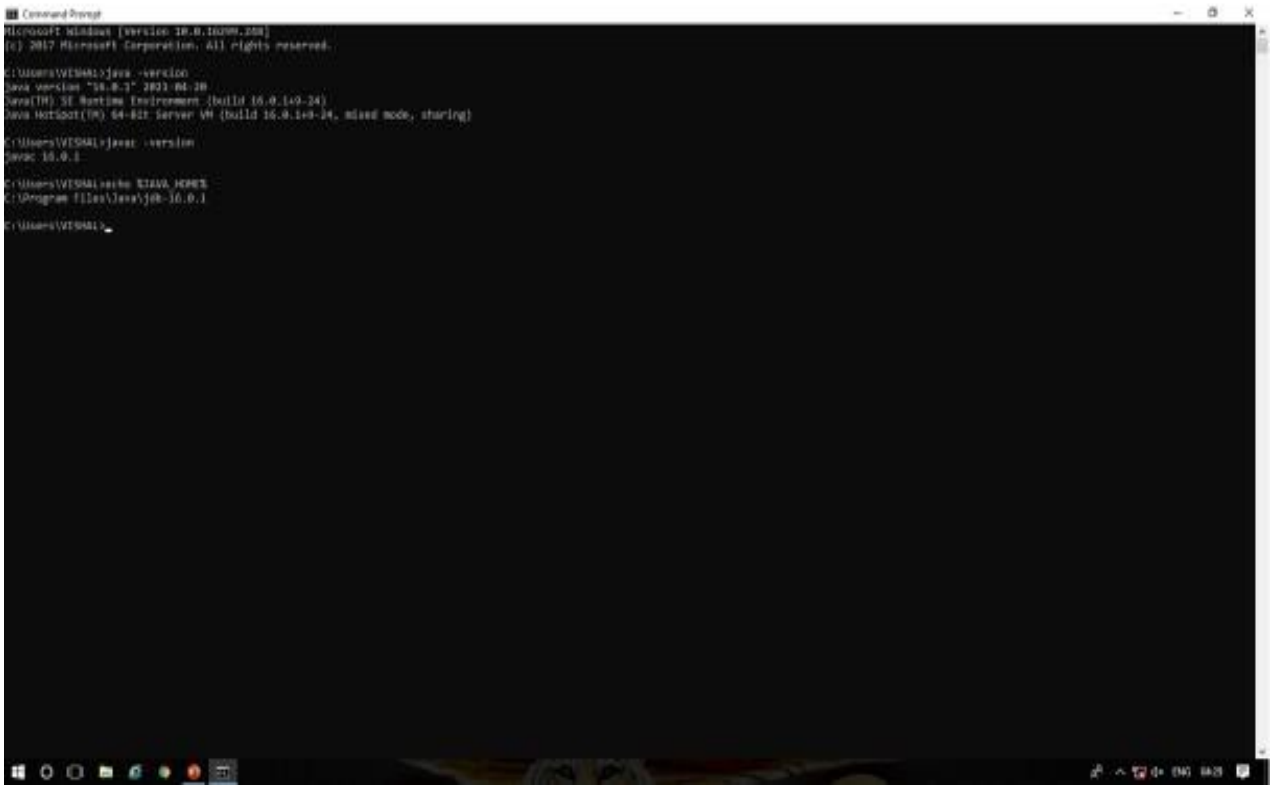


Now pess **ok**

Now go to command prompt and type 3 commands as shown below:

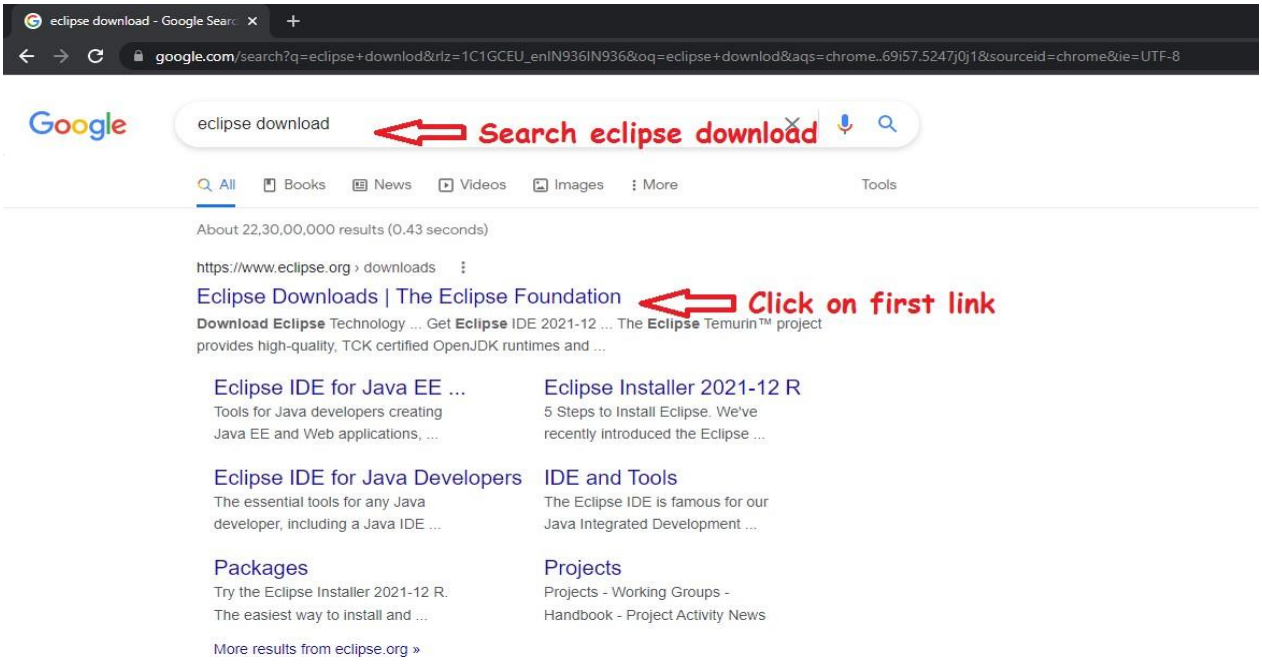
- java –version
- javac –version
- echo %JAVA_HOME%

If this results are found then java installed successfully.

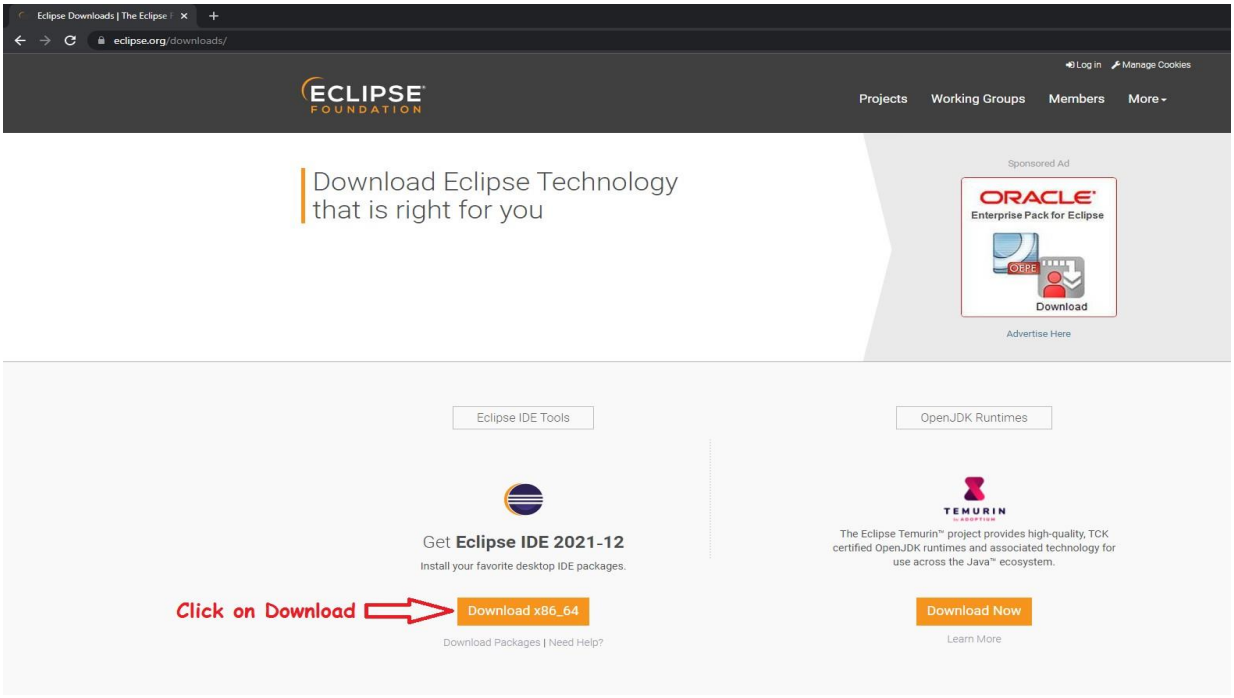


Eclipse Installation :

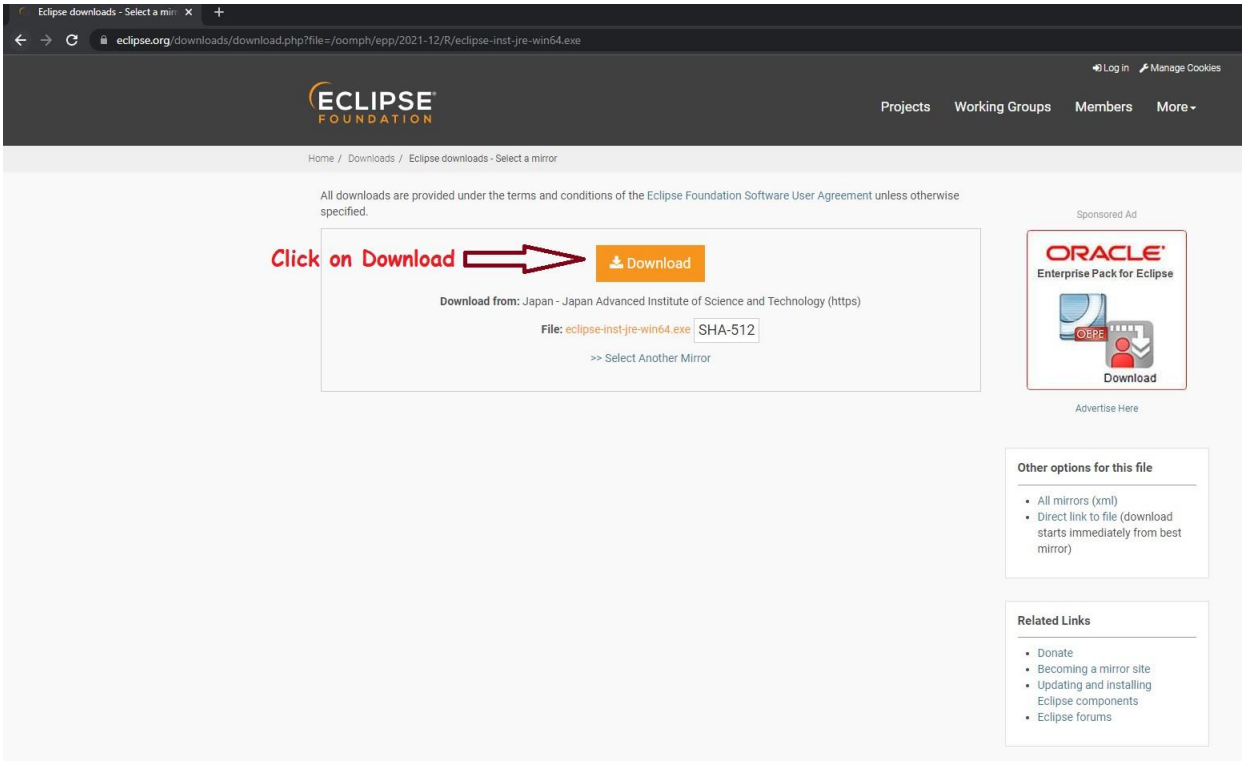
Step 1: Goto chrome browser and Serach **Eclipse Download** as shown in below screenshot



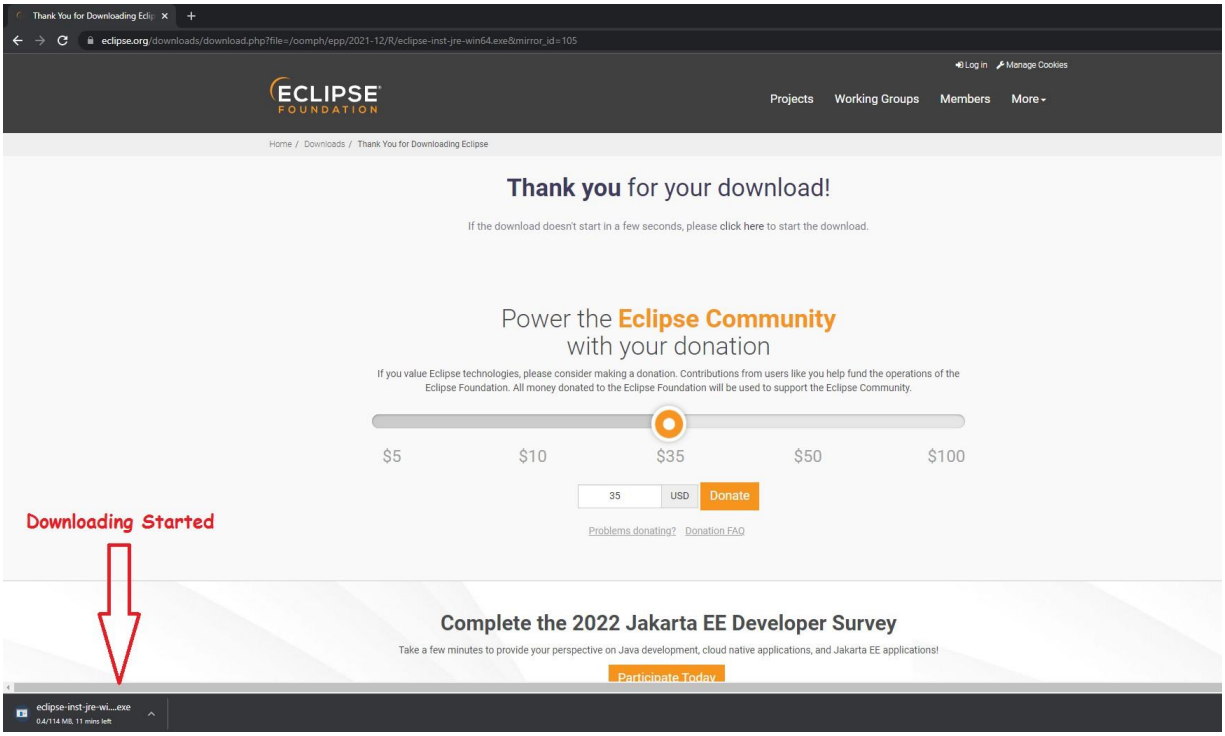
Step 2: Now you can see eclipse website open just click on Download button


Eclipse Downloads | The Eclipse
eclipse.org/downloads/
Log in
Manage Cookies
ECLIPSE FOUNDATION
Projects
Working Groups
Members
More
Download Eclipse Technology that is right for you
Sponsored Ad
ORACLE
Enterprise Pack for Eclipse
Download
Advertise Here
Eclipse IDE Tools
OpenJDK Runtimes
Get Eclipse IDE 2021-12
Install your favorite desktop IDE packages.
Click on Download
Download x86_64
Download Packages | Need Help?
TEMURIN
The Eclipse Temurin™ project provides high-quality, TCK certified OpenJDK runtimes and associated technology for use across the Java™ ecosystem.
Download Now
Learn More

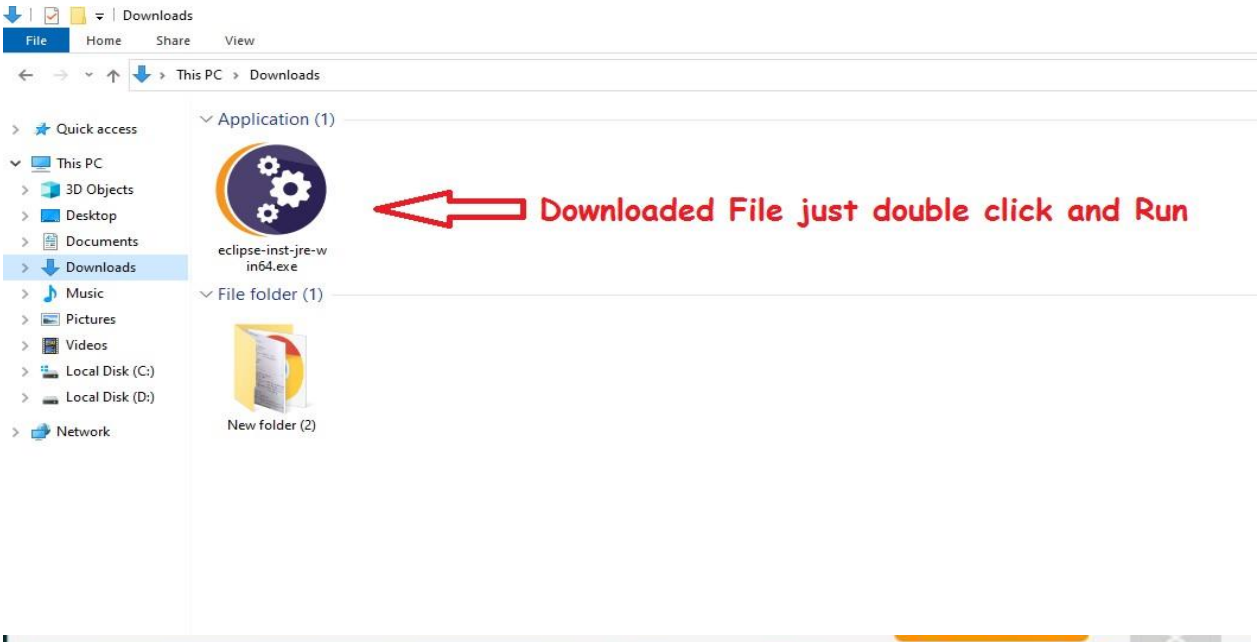
Step 3: You will see again next download window Simply click on **Download**



Step 4: Now your downloading started


Thank You for Downloading Eclip
eclipse.org/downloads/download.php?file=/oomph/epp/2021-12/R/eclipse-inst-jre-win64.exe&mirror_id=105
Log in
Manage Cookies
ECLIPSE FOUNDATION
Projects
Working Groups
Members
More
Home / Downloads / Thank You for Downloading Eclipse
Thank you for your download!
If the download doesn't start in a few seconds, please click here to start the download.
Power the Eclipse Community with your donation
If you value Eclipse technologies, please consider making a donation. Contributions from users like you help fund the operations of the Eclipse Foundation. All money donated to the Eclipse Foundation will be used to support the Eclipse Community.
$5
$10
$35
$50
$100
35
USD
Donate
Problems donating?
Donation FAQ
Downloading Started
Complete the 2022 Jakarta EE Developer Survey
Take a few minutes to provide your perspective on Java development, cloud native applications, and Jakarta EE applications!
Participate Today
eclipse-inst-jre-wi....exe
0.4/114 MB, 11 mins left

Step 5: Goto Download folder and click on downloaded file. Installing Started.

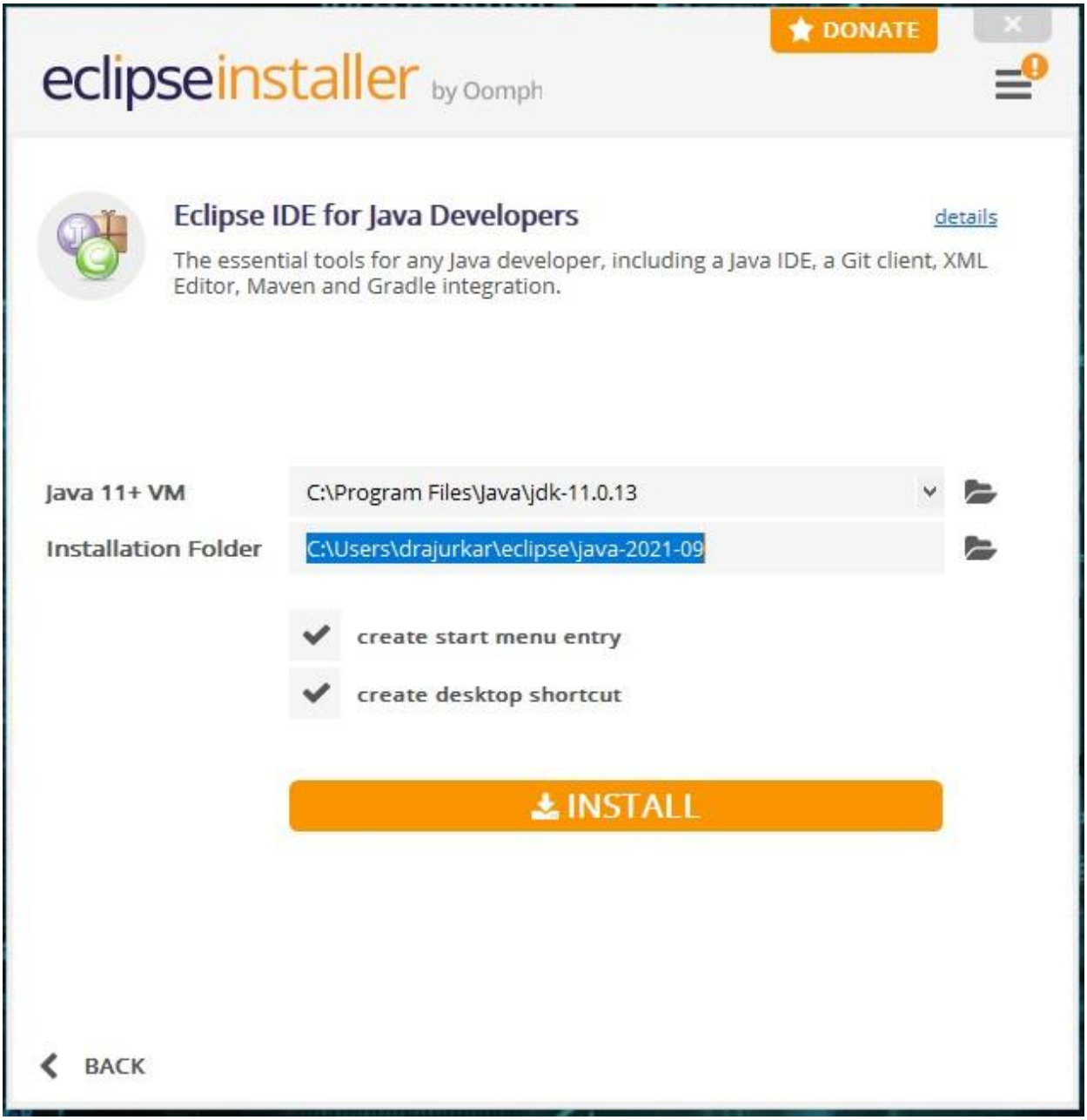


Step 6: Now, you will see the eclipseinstaller by Oomph window as below.

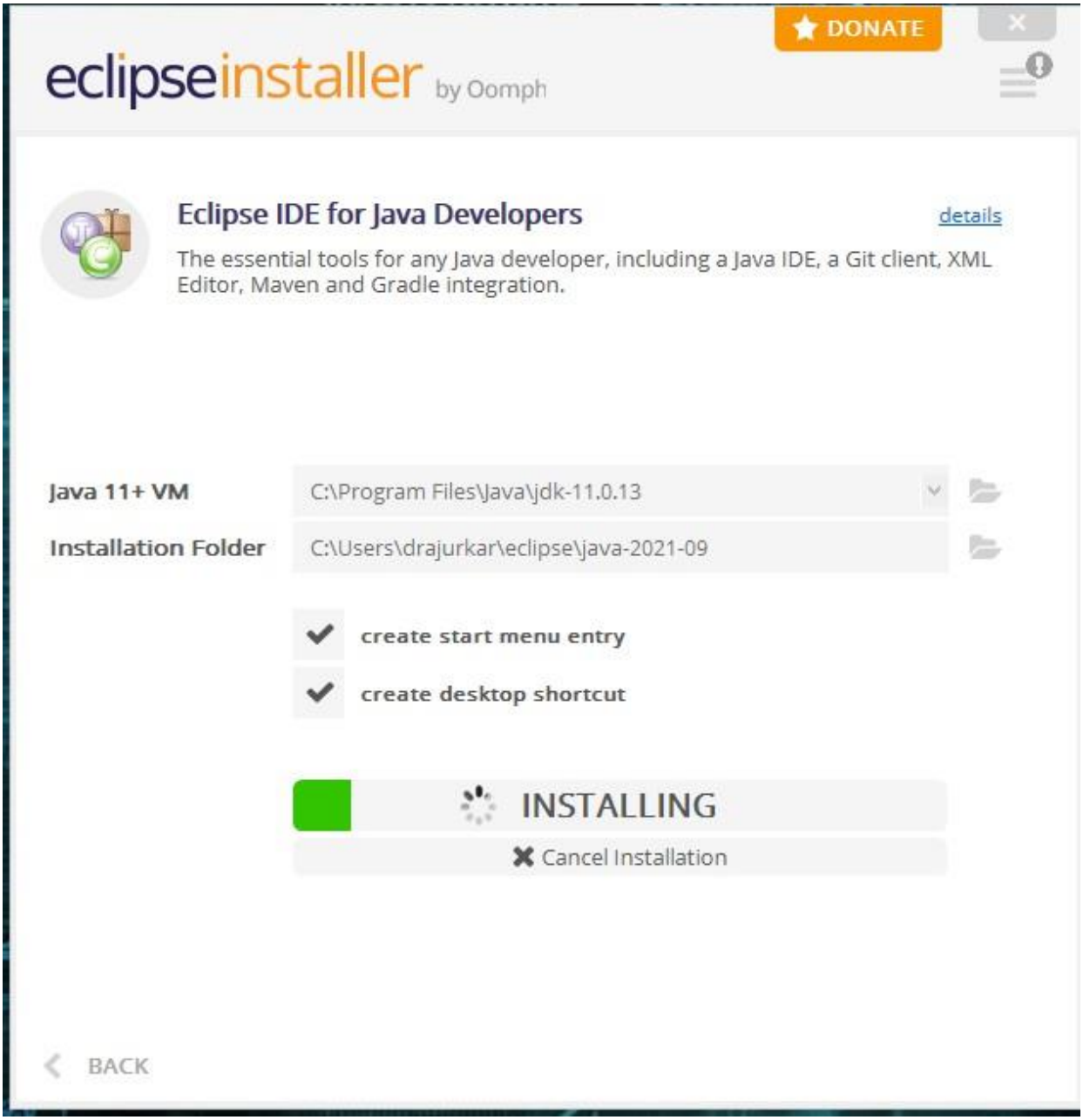
Click on Eclipse IDE for Java Develpers

Step 7: Specify the folder where you want to installed. The default folder will be in your user directory.

Select the Install button to begin installation

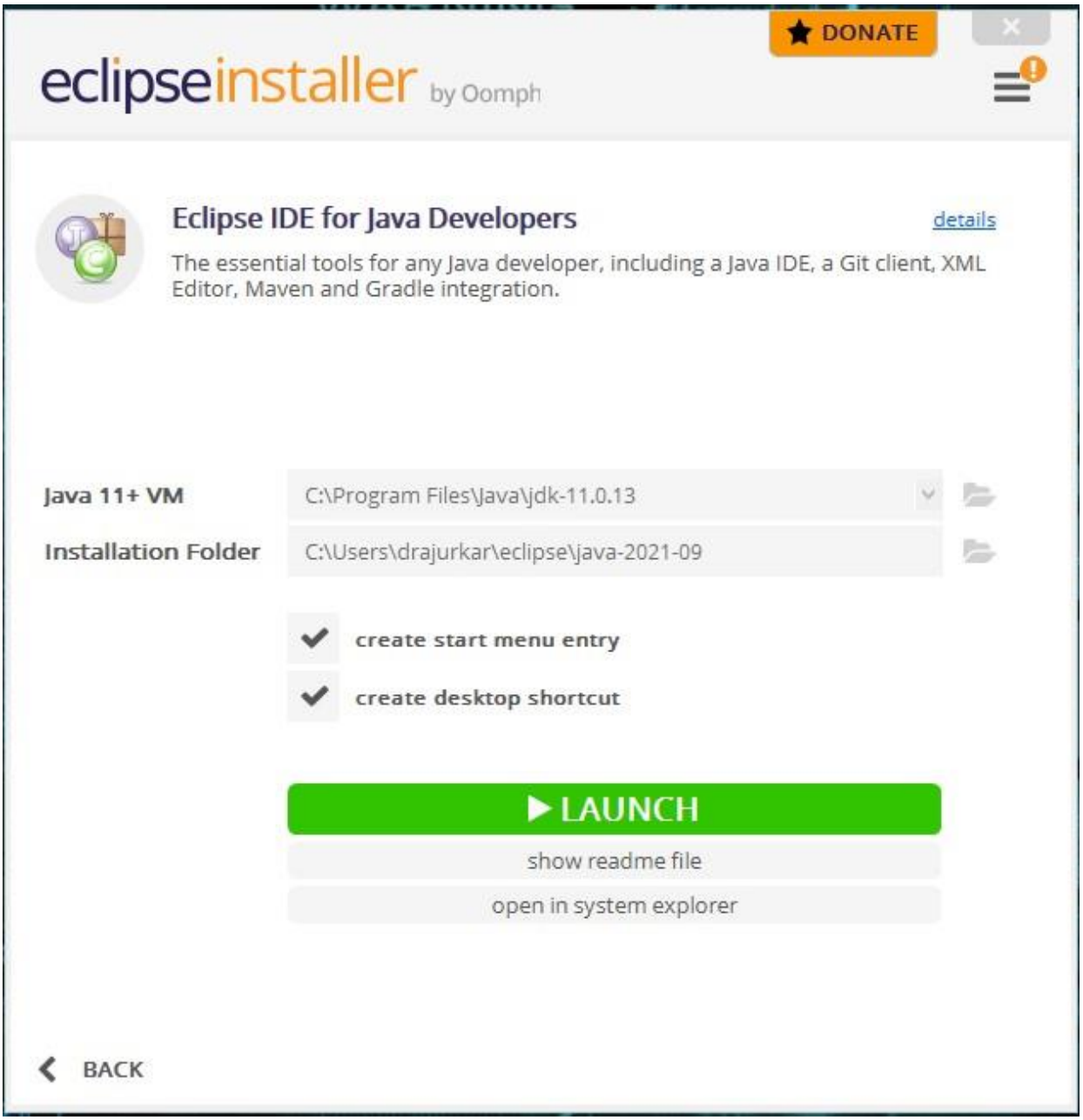


Step 8 : Now you will see below the installation started. Wait till installtion completed. It will take some time.



Step 9: Once Installation completed you will see the Launch button on same window.

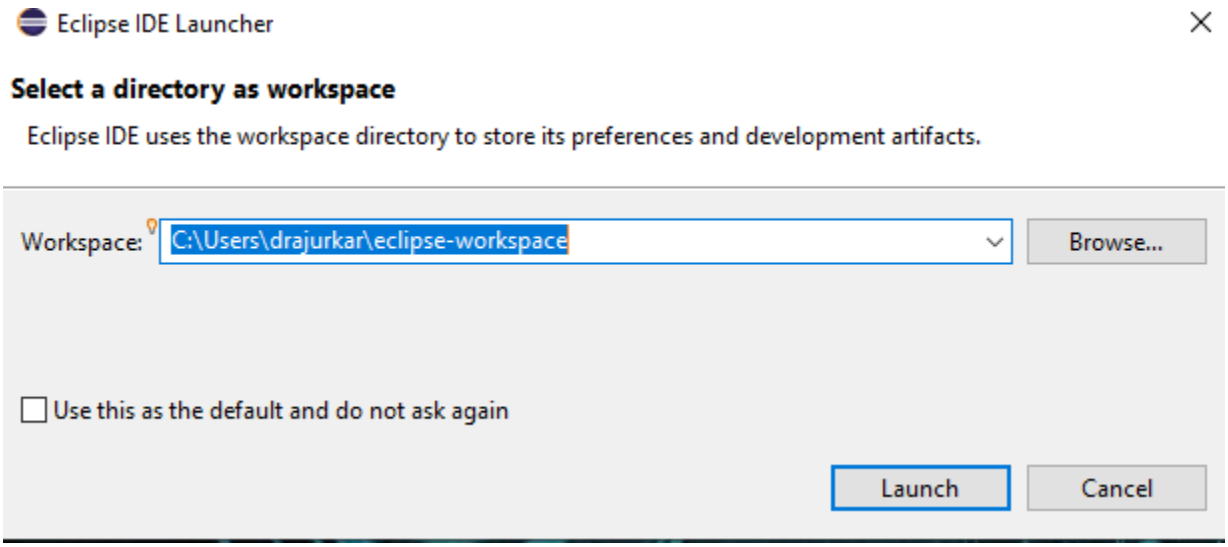
Just Click on Launch



Step 10: Once Launch one popup window showing. Here you can select the workspace where you want to store all project data.

Create your own folder and just select here by clicking browse button below.

Now, Just click on Launch



Step 11: The Eclipse installer has done its work.

Bonus Points:

In Eclipse IDE, We need to follow below 3 steps, while writing a program:

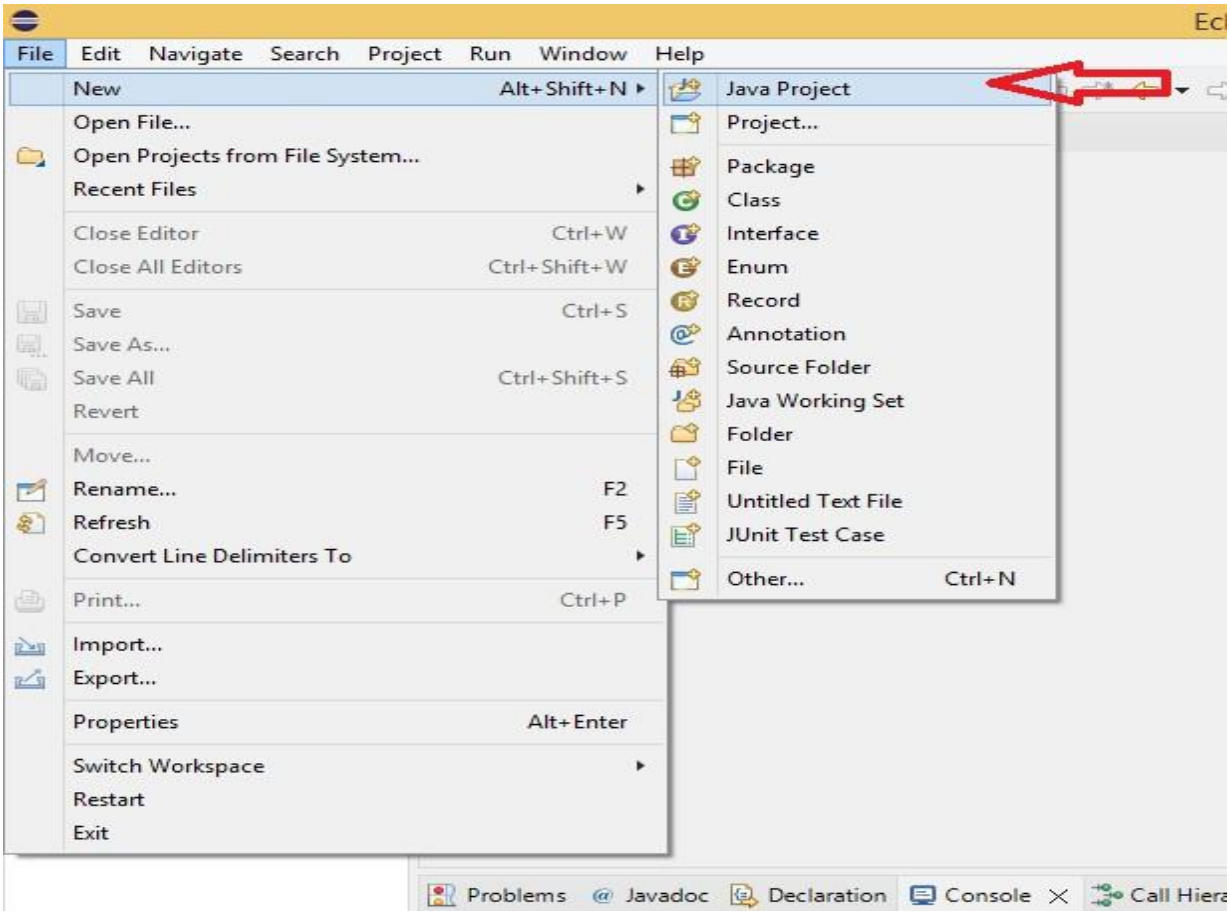
1. **Create a project**
2. **Create a package**
3. **Create class**

Project:

- Project is a collection of multiple packages, multiple classes, source folder, JRE system library, .jar files and many more…
- You can create many packages and classes inside the project.

How to create project:

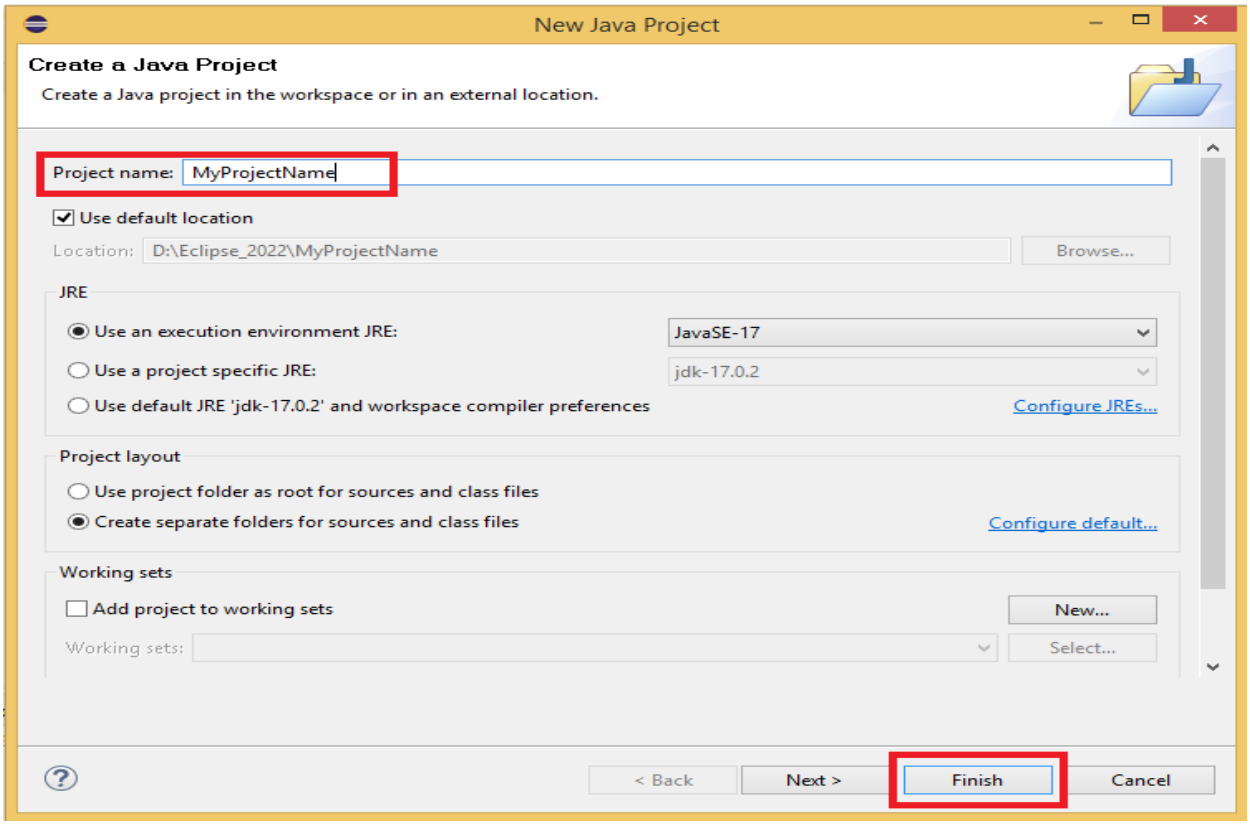
Step 1: Click on File Menu > New > Java Project


File
Edit
Navigate
Search
Project
Run
Window
Help
New Alt+Shift+N
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Import...
Export...
Properties Alt+Enter
Switch Workspace
Restart
Exit
Java Project
Project...
Package
Class
Interface
Enum
Record
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Other... Ctrl+N
Problems
Javadoc
Declaration
Console
Call Hiera

Step 2: We get new window Create Java Project

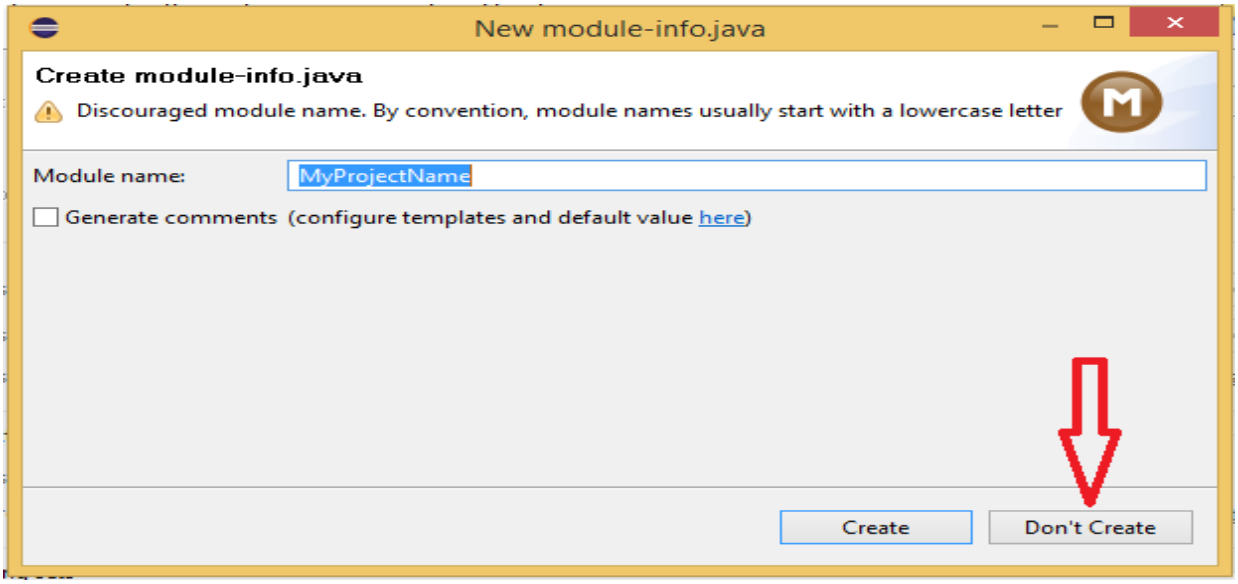
Enter project name (Name should start with uppercase letter)

Just click on Finish

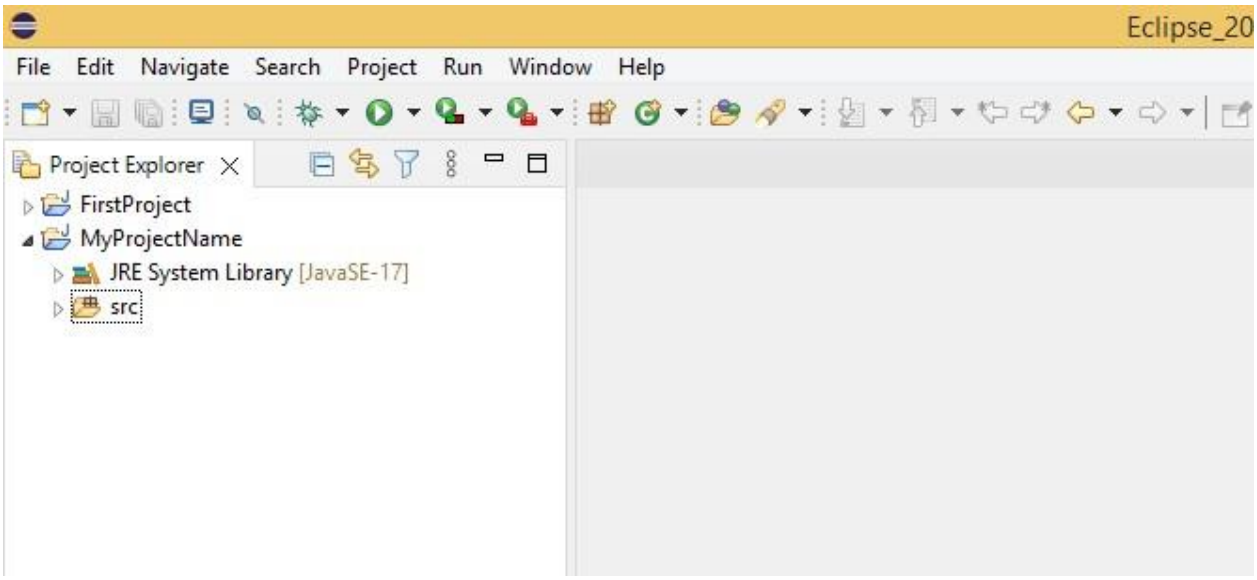


Step 3: After click on Finish, Will see new module info window

Click on don't create

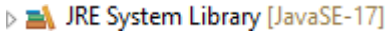

New module-info.java
Create module-info.java
Discouraged module name. By convention, module names usually start with a lowercase letter
Module name:
MyProjectName
Generate comments (configure templates and default value here)
Create
Don't Create

Step 4: All Done. Our project created.



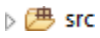
Note: If Project Explorer not visible just Go to Window > Show view > Project Explorer

JRE System Library:



- It is a library which contains predefine classes, module, keywords, syntax these are support to write a code.
- Any java program written in eclipse IDE can access predefined java functions because of a JRE system library.
- This particular JRE system library contains all the files with .jar extension.

Src Folder:



- It is a branch of project. All programming written in Src folder.
- Src folder is a collection of packages and classes.

- We can create multiple package and classes in Src folder.

Package:

- Package is a collection of multiple classes.
- There are two types of packages-

  User defined package – If we create package

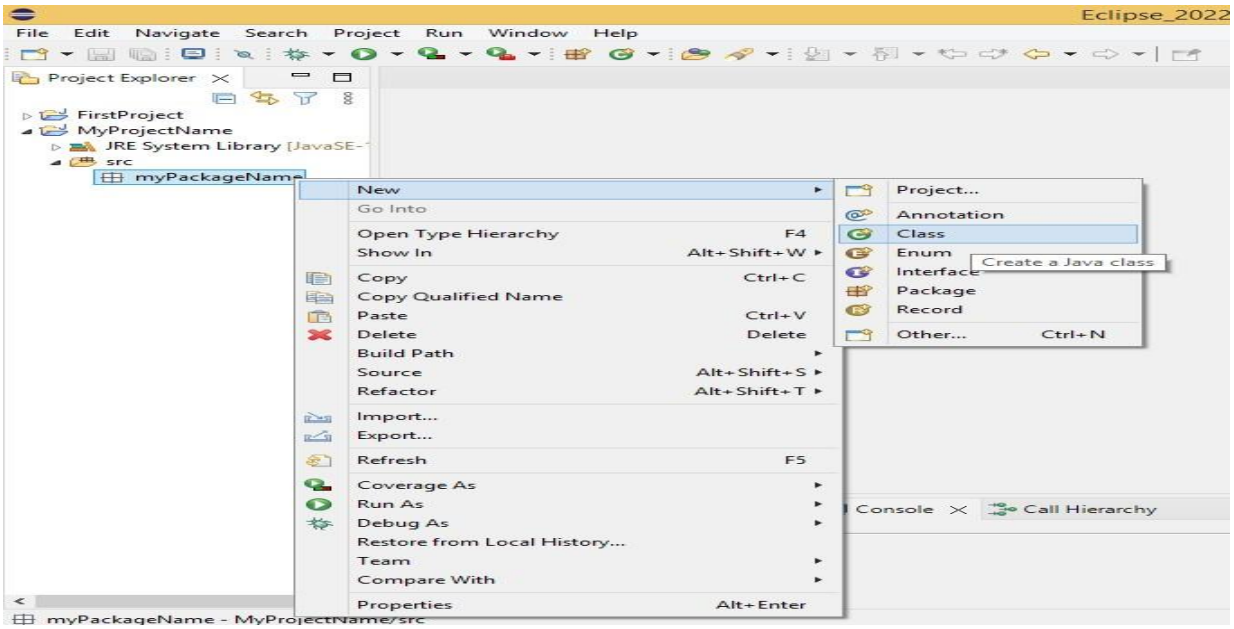
  Default package – If we are created number of classes without generating package and shows under default package.
- Syntax – package package_name ;
- If package does not contain class then symbol should be colorless 
- If class created inside package the symbol should be colorful 

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

  How to create class:

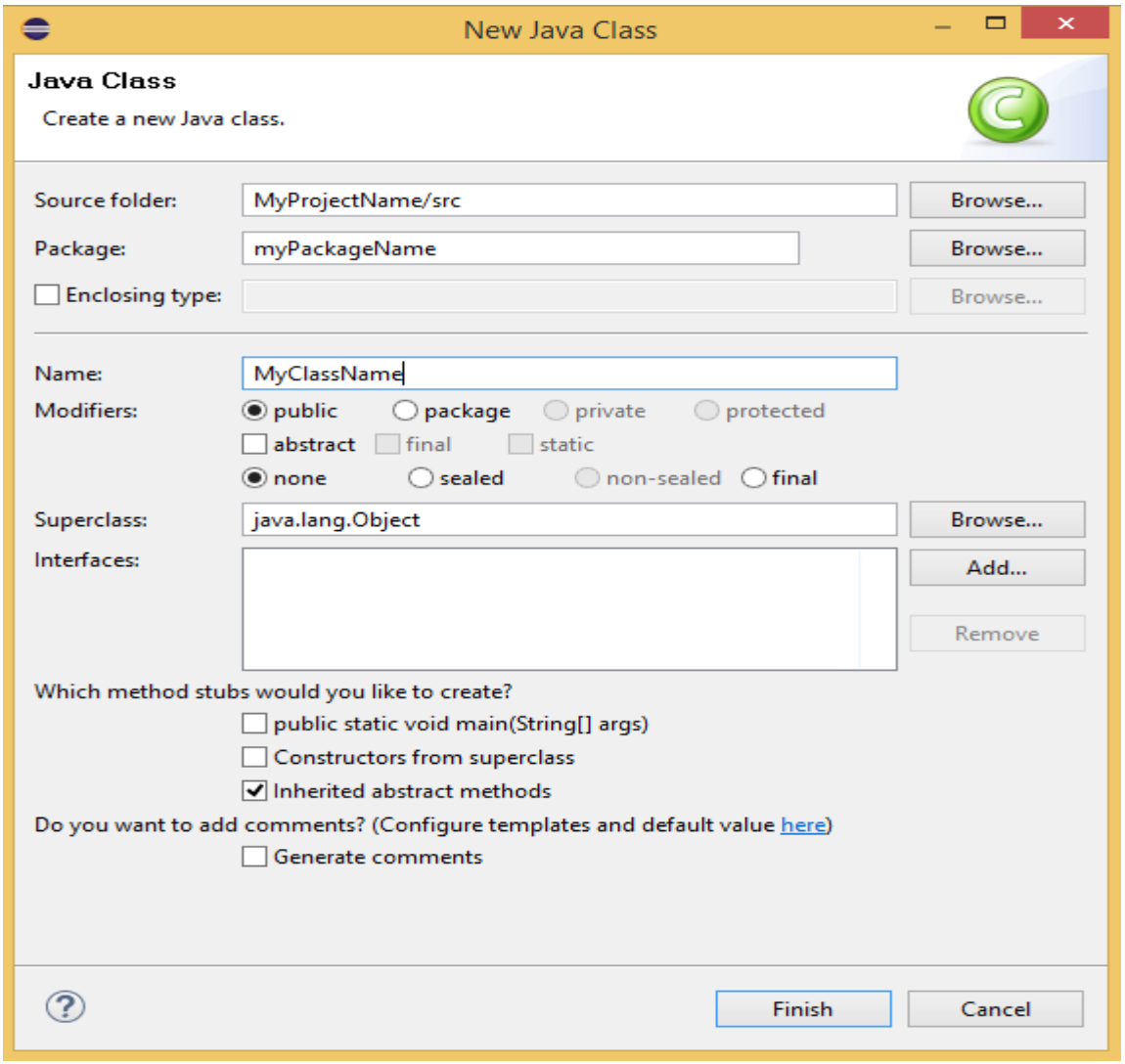
  Step 1: Right click on Package > New > Class


Eclipse_2022
File Edit Navigate Search Project Run Window Help
Project Explorer
FirstProject
MyProjectName
JRE System Library [JavaSE-
src
myPackageName
New
Go Into
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Coverage As
Run As
Debug As
Restore from Local History...
Team
Compare With
Properties Alt+Enter
Project...
Annotation
Class
Enum
Interface
Package
Record
Other... Ctrl+N
Create a Java class
Console
Call Hierarchy
myPackageName - MyProjectName/src

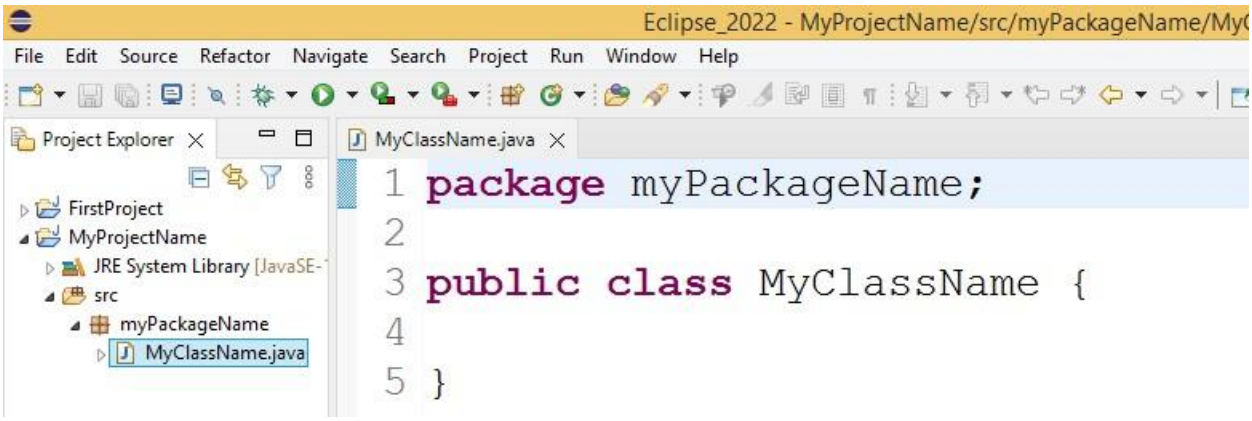
Step 2: Once click on Class, New Java Class window open as below

Insert Class Name in Name field

Click on Finish



Step 3: All Done. Class is created.


Eclipse_2022 - MyProjectName/src/myPackageName/MyC
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
MyClassName.java
FirstProject
MyProjectName
JRE System Library [JavaSE-
src
myPackageName
MyClassName.java
1 package myPackageName;
2
3 public class MyClassName {
4
5 }

Basis terms and Explanation

Case sensitivity

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

1. ClassName

- For all classname start with uppercase letter
- If have multi name class, then uppercase of each first letter of word
- Eg. public class MyFirstJavaProgram

2. 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()

3. ProjectName

- For all project name start with uppercase letter
- If have multi name project, then uppercase of each first letter of word
- Eg. MyProject

4. 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. myPackage

5. Comments (//)

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

6. Signature Bracket ()

- Also called argument 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 brakes
- 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

We can easily modify code using method

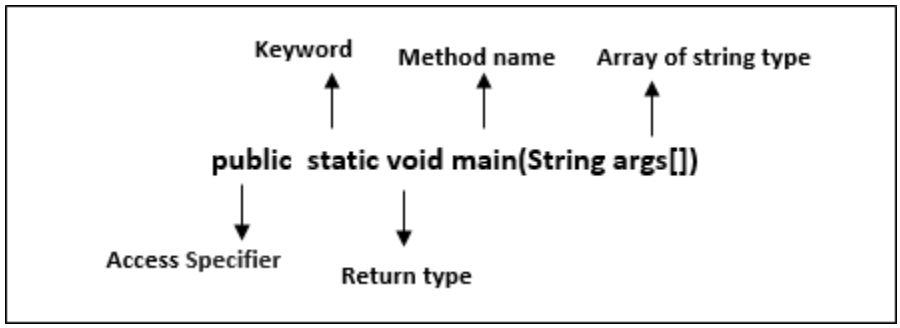
Method is always declared inside class body.

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:



1. **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.

2. **static:** You can make a method static by using the keyword static. We

should call the main() method without creating an object.

3. **void:** In Java, every method has the return type. Void keyword in main() method does not return any value.

4. **main():** It is a default method name which is predefined in the JVM.

5. **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

Static Method

- The method which contains static keyword and used to perform particular task
- It is static in nature
- To call static method there is no need to create object.
- Main method is 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();

Non-Static Method

- A method who does not contain static keyword is known as non-static method
- It is dynamic in nature, we can modify it.
- To call non-static method into main method, it is mandatory to create object.

What is Object?

- It is instance of class / example of class / Blueprint of class and used to call non-static method
- The object is an entity which has state and behavior like bike, chair, table, pen. It can be physical or logical.

How to Create Object in Java using new keyword

Syntax: ClassName ObjectReferenceVariable = **new** Constructor();

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.

Sample Program for Non Static method:

SecondClass.java

```
1  package firstPackageJava;
2
3  public class SecondClass {
4  //Non-Static Method
5  //Syntax : Classname ObjectRefVariable = new Classname();
6          public void googlepay()
7          {
8              System.out.println("Im Non Static method");
9          }
10 //Main Method
11     public static void main(String[]args)
12     {
13         SecondClass g = new SecondClass(); //Obj. Creation
14         g.googlepay();  //Non-static Method Calling
15     }
16 }
```

Problems | Javadoc | Declaration | Console | Call Hierarchy

<terminated> SecondClass [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Mar 7, 2022, 8:37:26 PM – 8:37:27 PM)

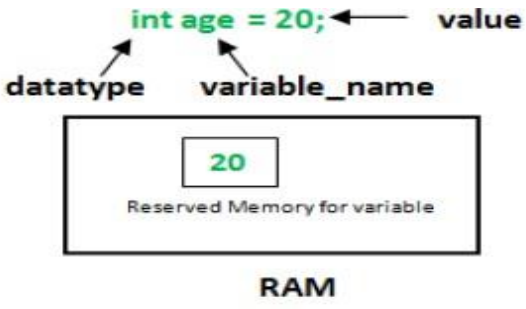
```
Im Non Static method
```

Variables

- Variable is a piece of memory used to store the information.
- Variable always assigned with datatype.
- A variable is a name given to a memory location.
- According to all programming language we cannot declare information directly, so variables are introduced.

- Reusable – It help us the info will use again and again.

How to Declare Variables?



Datatype variable = information;

There are three types of variable

1. Local variable.
2. Static variable
3. Instance/Global variable

Local variable

- A variable declared inside the body of the method is called local variable.
- You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
- A local variable cannot be defined with "static" keyword.

Static variable

- A variable that is declared as static is called a static variable. It cannot be local.
- 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

Instance/Global variable

- A variable declared inside the class but outside the body of the method is called as Instance variable
- It is also known as Global Variable.

We have default values:

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

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.

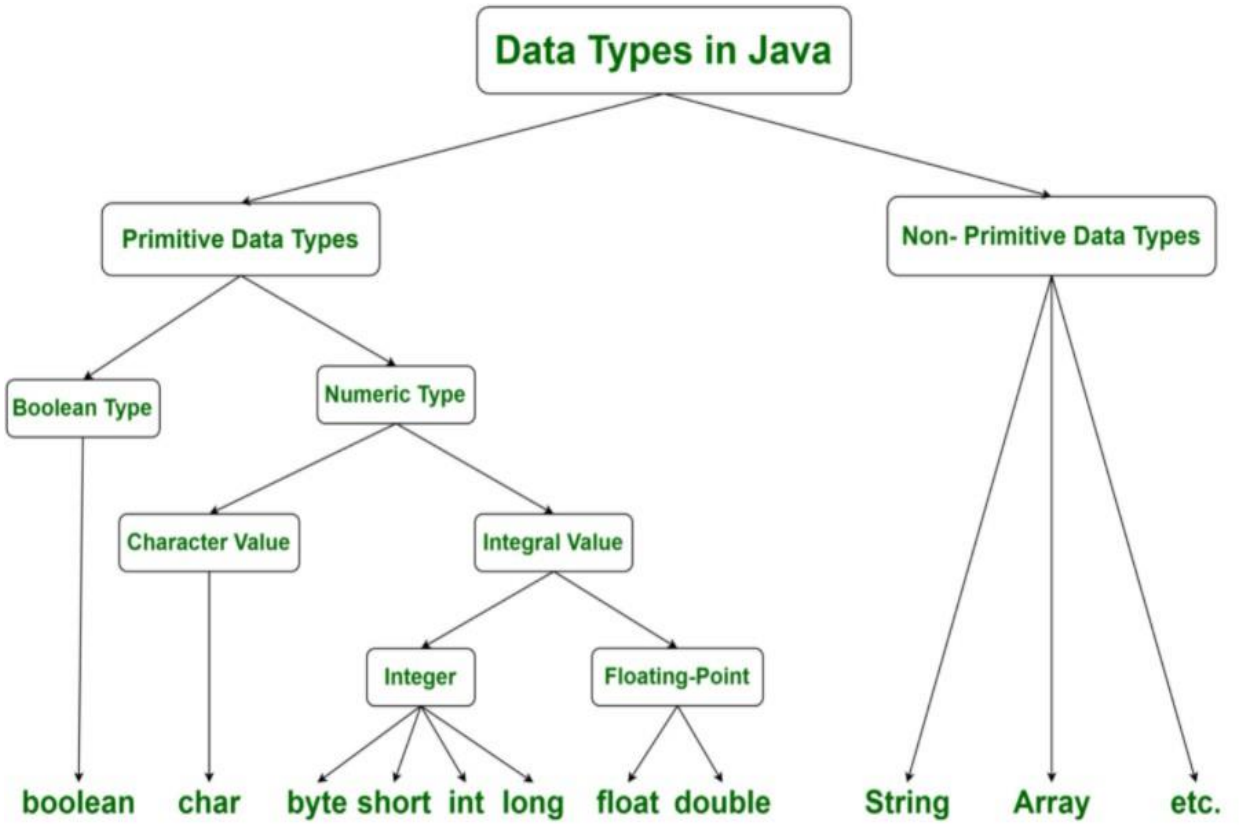
Types of data types in Java:

Primitive data types:

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

Non-primitive data types:

- The non-primitive data types include Classes, Interfaces, and Arrays.



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

Type	Size	Range	Default
boolean	1 bit	true or false	false
byte	8 bits	[-128, 127]	0
short	16 bits	[-32,768, 32,767]	0
char	16 bits	['\u0000', '\uffff'] or [0, 65535]	'\u0000'
int	32 bits	[-2,147,483,648 to 2,147,483,647]	0
long	64 bits	$[-2^{63}, 2^{63}-1]$	0
float	32 bits	32-bit IEEE 754 floating-point	0.0
double	64 bits	64-bit IEEE 754 floating-point	0.0

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';

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 similar to method.
- Constructors are special member of class
- According to java each class must have a constructor, if no constructor is present then at time of compilation compiler will create a default constructor for the class and this default constructor will have a blank body.

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:

- By creating object of a class.
  - Constructor executes automatically when we create an object.
- By using '**new**' keyword with constructor name with method signature followed by semicolon.
  - i.e **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 programmer declared constructor then compiler will not provide any default constructor
- If constructor provided by compiler at the time of compilation is known as default constructors
- The best example of default constructor is main method

```
package constructorExamples;

public class Example1 {

	public void method1()
	{
		String name ="Shivlila";
		System.out.println(name);
	}

	public static void main(String args[])
	{
		Example1 obj=new Example1();
		obj.method1();
	}

}
```

2. User defined constructor

- A programmer is declaring 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
2. parameterized constructor

1. Zero argument constructor

- We can initialize data member of a class
- constructor with no argument constructor is known as zero argument constructor or non-argument constructor
- the signature is same as a default constructor

```
package constructorExamples;

public class Example2 {

//declaration
static int a;
static String name;

//constructor
Example2()
{
        //initlition
        a=20;
        name="Ganesh";
        //usage
        System.out.println(a);
        System.out.println(name);
}
public static void main(String args[])
{
        Example2 xyz=new Example2();
}
}
```

2. Parameterize constructor

- Parameterize constructor a constructor is called as parameterized constructor
- when it accepts a specific number of a parameter or argument in a signature bracket to initialize data members of a class with a distinct value
- this constructor is used to allow multiple constructor by providing different types of arguments in simple words constructors with argument is known as parameters constructors
- access scope of this constructor will be same as a class access code it means if class is a public then constructor is a public hip classes private then constructor is private or vice versa
- constructors are going to invoke/use at the time of object creation at the time of object creation

```
package constructorExamples;

public class ArgumentCons {

String city;
ArgumentCons()
{
        city="Pune";
        System.out.println(city);
}

ArgumentCons(int a)
{
        System.out.println(a);
}

ArgumentCons(String name)
```

```
{
        System.out.println(name);
}

public static void main(String args[])
{
        ArgumentCons obj1=new ArgumentCons();
        ArgumentCons obj2=new ArgumentCons(852681256);
        ArgumentCons obj3=new ArgumentCons("Shivlila");
}
}
```

Java Keywords

Java keywords are also known as reserved words. Keywords are particular words that act as a key to a code. These are predefined words by Java so they cannot be used as a variable or object name or class name.

List of Java Keywords

1. abstract: Java abstract keyword is used to declare an abstract class. An abstract class can provide the implementation of the interface. It can have abstract and non-abstract methods.

2. boolean: Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.

3. break: Java break keyword is used to break the loop or switch statement. It breaks the current flow of the program at specified conditions.

4. byte: Java byte keyword is used to declare a variable that can hold 8-bit data values.

5. case: Java case keyword is used with the switch statements to mark blocks of text.

6. catch: Java catch keyword is used to catch the exceptions generated by try statements. It must be used after the try block only.

7. char: Java char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters

8. class: Java class keyword is used to declare a class.

9. continue: Java continue keyword is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.

10. default: Java default keyword is used to specify the default block of code in a switch statement.

11. do: Java do keyword is used in the control statement to declare a loop. It can iterate a part of the program several times.

12. double: Java double keyword is used to declare a variable that can hold 64-bit floating-point number.

13. else: Java else keyword is used to indicate the alternative branches in an if statement.

14. enum: Java enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.

15. extends: Java extends keyword is used to indicate that a class is derived from another class or interface.

16. final: Java final keyword is used to indicate that a variable holds a constant value. It is used with a variable. It is used to restrict the user from

updating the value of the variable.

17. finally: Java finally keyword indicates a block of code in a try-catch structure. This block is always executed whether an exception is handled or not.

18. float: Java float keyword is used to declare a variable that can hold a 32-bit floating-point number.

19. for: Java for keyword is used to start a for loop. It is used to execute a set of instructions/functions repeatedly when some condition becomes true. If the number of iteration is fixed, it is recommended to use a for loop.

20. if: Java if keyword tests the condition. It executes the if block if the condition is true. 21. implements: Java implements keyword is used to implement an interface.

22. import: Java import keyword makes classes and interfaces available and accessible to the current source code.

23. instance of: Java instance of keyword is used to test whether the object is an instance of the specified class or implements an interface.

24. int: Java int keyword is used to declare a variable that can hold a 32-bit signed integer.

25. interface: Java interface keyword is used to declare an interface. It

can have only abstract methods.

26. long: Java long keyword is used to declare a variable that can hold a 64-bit integer.

27. native: Java native keyword is used to specify that a method is implemented in native code using JNI (Java Native Interface).

28. new: Java new keyword is used to create new objects.

29. null: Java null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.

30. package: Java package keyword is used to declare a Java package that includes the classes.

31. private: Java private keyword is an access modifier. It is used to indicate that a method or variable may be accessed only in the class in which it is declared.

32. protected: Java protected keyword is an access modifier. It can be accessible within the package and outside the package but through inheritance only. It can't be applied with the class.

33. public: Java public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.

34. return: Java return keyword is used to return from a method when its execution is complete.

35. short: Java short keyword is used to declare a variable that can hold a 16-bit integer.

36. static: Java static keyword is used to indicate that a variable or method is a class method. The static keyword in Java is mainly used for memory management.

37. strictfp: Java strictfp is used to restrict the floating-point calculations to ensure portability.

38. super: Java super keyword is a reference variable that is used to refer to parent class objects. It can be used to invoke the immediate parent class method.

39. switch: The Java switch keyword contains a switch statement that executes code based on test value. The switch statement tests the equality of a variable against multiple values.

40. synchronized: Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.

41. this: Java this keyword can be used to refer the current object in a method or constructor.

42. throw: The Java throw keyword is used to explicitly throw an exception. The throw keyword is mainly used to throw custom exceptions. It is followed by an instance.

43. throws: The Java throws keyword is used to declare an exception. Checked exceptions can be propagated with throws.

44. transient: Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.

45. try: Java try keyword is used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.

46. void: Java void keyword is used to specify that a method does not have a return value.

47. volatile: Java volatile keyword is used to indicate that a variable may change asynchronously.

48. while: Java while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iteration is not fixed, it is recommended to use the while loop.

This Keyword:

- This is a keyword refers to current class instance variable in a method or constructor.
- This keyword is used to solve the naming conflict.
- This keyword is used to invoke current class method.
- If we don't add this keyword in the program, compiler will automatically add this keyword while invoking the method.
- This keyword is used to invoke (Initiate) current class constructor.

Let's understand the problem if we don't use this keyword by the example given below:

ThisKeyword.java

```
1  package thisKeyword;
2  public class ThisKeyword {
3      int rollno;
4      String name;
5      float fee;
6      ThisKeyword(int rollno,String name,float fee)
7      {
8          rollno=rollno;
9          name=name;
10         fee=fee;
11     }
12     void display()
16     public static void main(String args[])
17     {
18         ThisKeyword s1=new ThisKeyword(150,"xyz",160f);
19         ThisKeyword s2=new ThisKeyword(200,"abc",170f);
20         s1.display();
21         s2.display();
22     }}
```

Problems | Console | Javadoc | Declaration | Call Hierarchy

<terminated> ThisKeyword (1) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Mar 24, 2022, 2:05:42 PM – 2:05:42 PM)

```
0 null 0.0
0 null 0.0
```

In the above example, parameters and instance variables are same. So, we are using this keyword to distinguish local variable and instance variable.

Solution of the above problem by this keyword

ThisKeyword.java

```
1  package thisKeyword;
2  public class ThisKeyword {
3      int rollno;
4      String name;
5      float fee;
6      ThisKeyword(int rollno,String name,float fee)
7      {
8          this.rollno=rollno;
9          this.name=name;
10         this.fee=fee;
11     }
12     void display()
16     public static void main(String args[])
17     {
18         ThisKeyword s1=new ThisKeyword(150,"xyz",160f);
19         ThisKeyword s2=new ThisKeyword(200,"abc",170f);
20         s1.display();
21         s2.display();
22     }}
```

Problems | Console | Javadoc | Declaration | Call Hierarchy

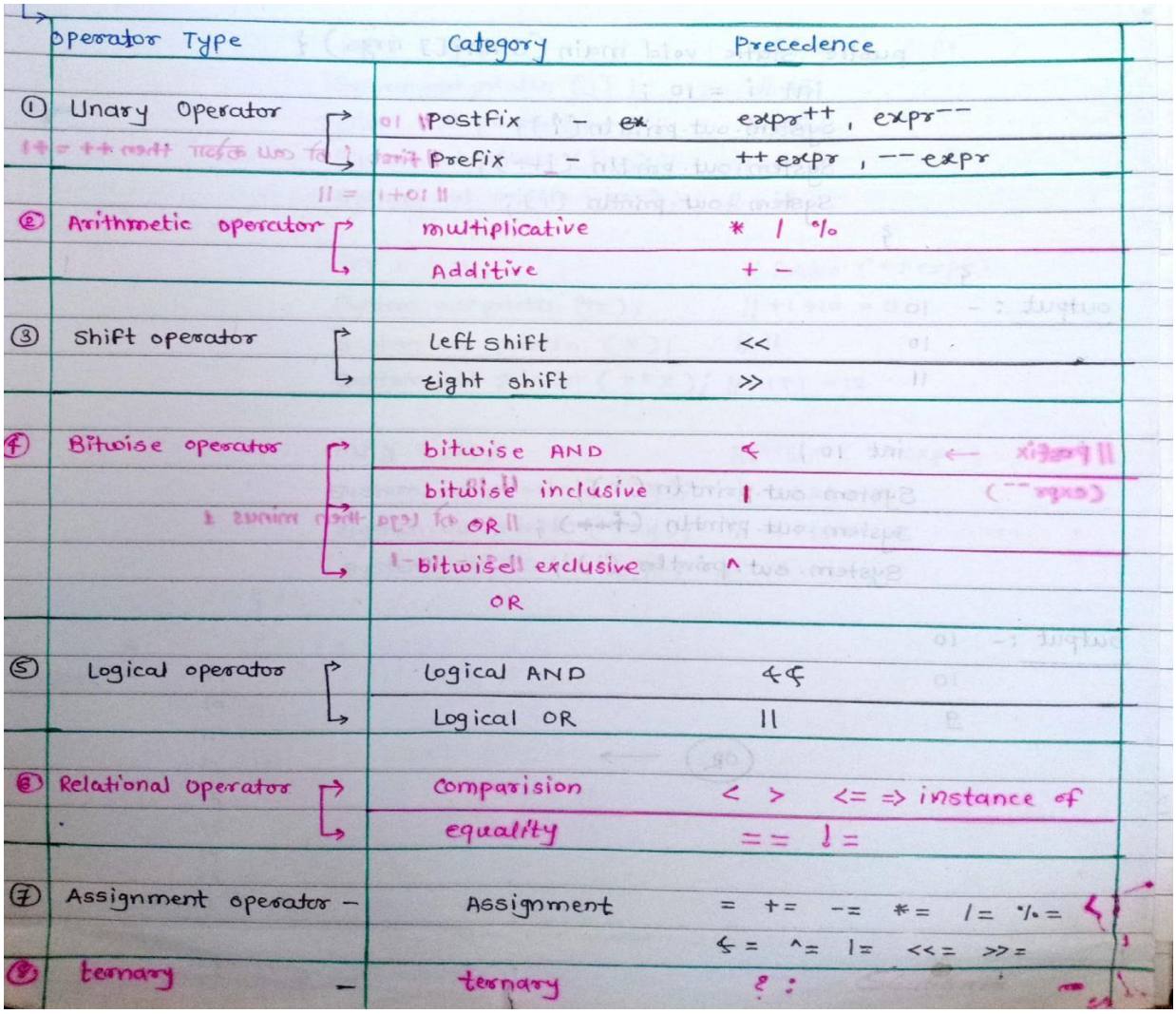
```
<terminated> ThisKeyword (1) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe  (Mar 24, 2022, 2:08:11 PM – 2:08:11 PM)
150 xyz 160.0
200 abc 170.0
```

Operators:

- Operator in java is symbol which is used to perform operations.
- Operators are used to manipulate variables.

There are many types of operators are given below:

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



	Operator Type	Category	Precedence
①	Unary Operator	Postfix – ex	expr++ , expr--
		Prefix –	++expr , --expr
②	Arithmetic operator	multiplicative	* / %
		Additive	+ –
③	Shift operator	Left shift	<<
		Right shift	>>
④	Bitwise operator	bitwise AND	&
		bitwise inclusive OR	|
		bitwise exclusive OR	^
⑤	Logical operator	Logical AND	&&
		Logical OR	||
⑥	Relational Operator	Comparision	< > <= => instance of
		equality	== !=
⑦	Assignment operator –	Assignment	= += -= *= /= %= &= ^= |= <<= >>=
⑧	ternary –	ternary	? :

Sumago Infotech Pvt. Ltd.

Strive With Technology...!

Control Statements

- As the name suggests control statements means which controls the flow of the program.
- A java program executes from top to bottom but if we want to control the order of execution of our program based on the logic applied, we use control statements.

Various types of Control Statements

1. Conditional Statements

- If Statement
- If else
- If else-if
- Nested if
- Nested if-else
- Switch

2. Looping Statements

- for loop
- while loop
- do while loop

3. Jump Statements

- break
- continue

If Statement

If statement is used to test the condition. It will execute the if block only when condition is true.

Syntax

```
if (condition) //will execute when condition is true
{
    //Code
}
```

Program

```
public class If_Statement {

public static void main(String[] args) {

int num = 5;

if (num > 0)
{
System.out.println("Number is a greater than zero");
}

System.out.println("End of the code");

}

}
```

Sumago Infotech Pvt. Ltd.

Strive With Technology...!

If Else Statement

- In If-else statement , if the specified condition in the if statement is true it will execute if block otherwise else block will be executed.

Syntax

```
if (condition)  //when condition is true it will execute
{
    //code
}
else            // when condition is false it will execute
{
    //code
}
```

Program

```
public class IfElse {

public static void main(String[] args) {

int num = 10;

if (num>0)
{
System.out.println("Number is greater than Zero");
}
else
{
System.out.println("Number is less than Zero ");
}

}

}
```

Sumago Infotech Pvt. Ltd.

Strive With Technology...!

If Else-If Statement

- The if else-if statement contains if statement followed by multiple else-if statements.
- In if else-if statement, suppose if condition is true, then if statement will be executed and the remaining code will be skipped.
- But suppose no conditions are true then else block will be executed.

Syntax

```
if (condition 1)|
{
     //code
}
else if (condition 2)
{
     //code
}

else

{
     //code
}
```

Program

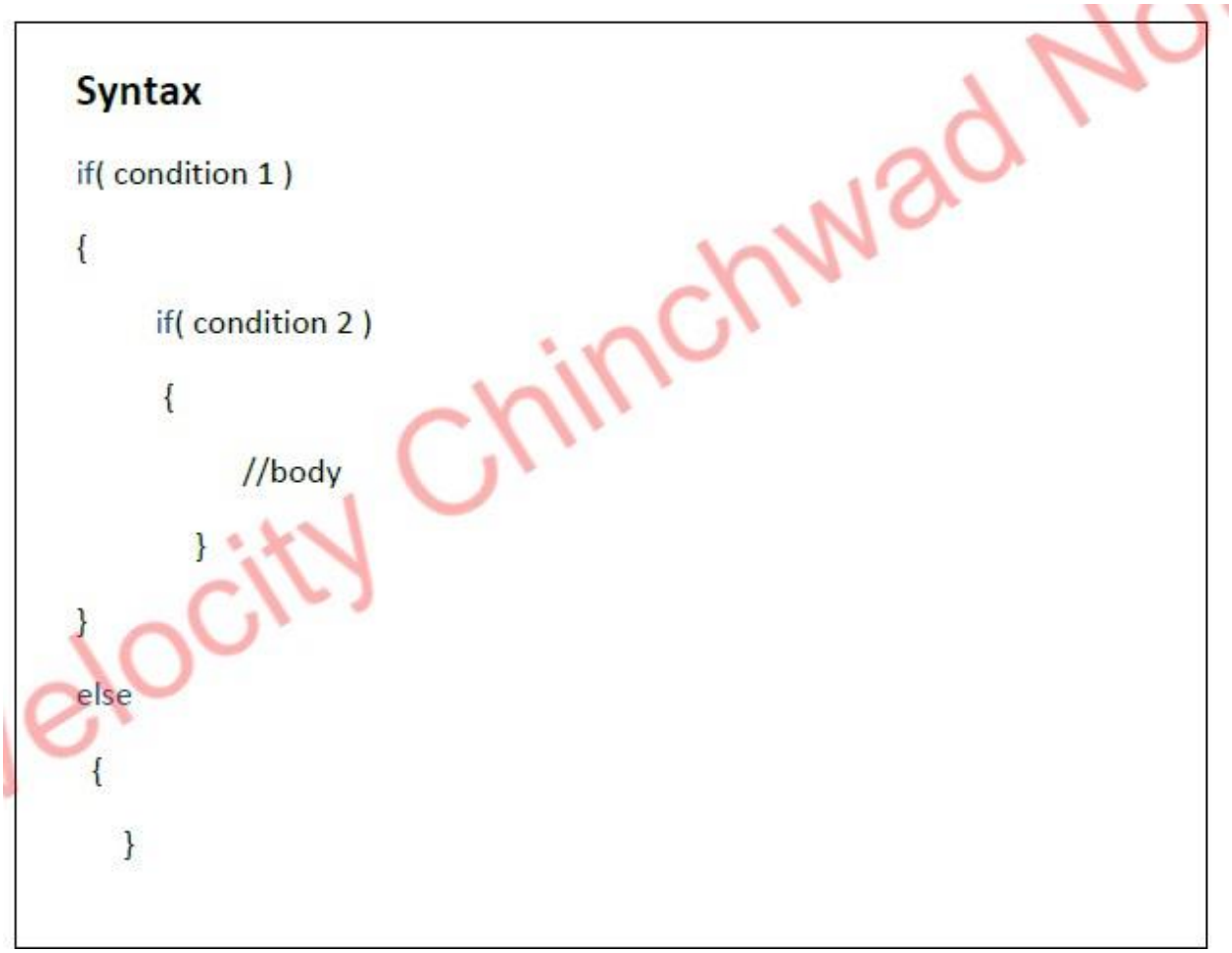
```
public class If_ElseIF {

     public static void main(String[] args) {

          int money = 101;

          if(money > 3000 && money <= 10000)
          {
               System.out.println("I will buy a smart phone");
          }
          else if (money > 10000 && money <= 30000)
          {
               System.out.println("I will buy a Bicycle");
          }
          else if(money > 30000 && money <= 100000)
          {
               System.out.println("I will buy a laptop");
          }

          else
          {
               System.out.println("Insufficient Money");
          }
     }
}
```

Sumago Infotech Pvt. Ltd.

Strive With Technology...!

Nested If Statement

- In nested if-statement, the if statement can contain a if statement inside it.



Program

```
public class NestedIF {

public static void main(String[] args) {

System.out.println("Welcome Blood Donation Camp");
int age = 14;
int weight = 35;

if(age >= 18 )
{
System.out.println("COndition 1 satisfied");

if(weight > 45)
{
System.out.println("COndition 2 satisfied");
System.out.println("You are Eligible to donate blood");
}
}
else
{
System.out.println("Your age is not valid to donate the
blood");
}

}

}
```


Nested If-else Statement

- In nested if statement , the if statement contains a if-else statement inside it.

Syntax

```
if( condition 1 )
{
    if( condition 2 )
    {
        //code

    }
    else
     {

     }

}
else
 {

  }
```


Program

```
public class NestedIFElse {

public static void main(String[] args) {
System.out.println("Even, odd or Negative");

int a = -7;

if (a > 0) {
System.out.println("Yes number is positive");
if (a % 2 == 0) {
System.out.println("Yes number is an Even number");
} else {
System.out.println("Number is Odd");
}
} else {

System.out.println("Its a Negative number");
}
}
}
```

Switch case Statement

- In Java, Switch statement contains multiple block of code called as cases and a single case is executed based on the variable which is being switched.
- In switch statement we are going to use only one input.
- Switch statement works with int, String, char , etc.
- The case value must be unique, it should not be duplicate.
- Each case statement can have a break statement to terminate the code ones condition is satisfied. And it is optional , if not used next case will be executed.
- While using switch statements, we must see that the case expression will be of the same type as of variable.

Program

```
public class SwitchCase {

	public static void main(String[] args) {

		char c = 'g';
		switch(c)
		{
		case 'a':
			System.out.println(c+" is a vowel");
			break;
		case 'e':
			System.out.println(c+" is a vowel");
			break;
		case 'i':
			System.out.println(c+" is a vowel");
			break;
		case 'o':
			System.out.println(c+" is a vowel");
			break;
		case 'u':
			System.out.println(c+" is a vowel");
			break;
			default:
				System.out.println(c+" Is not a Vowel");
		}
	}
}
```

Java User Input (Scanner)

Scanner Class

- The Scanner class is mainly used to get the user input, and it belongs to the java.util package.
- In order to use the Scanner class, you can create an object of the class and use any of the Scanner class methods.

Input Types

Method	Description
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads a int value from the user
nextLine()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user
nextLine().chatAt(i)	Reads a char value from the user

Object creation

```
Scanner input = new Scanner(System.in)
```

Examples:-

```
import java.util.Scanner;  // Import the Scanner class
 public class Example1 {
            public static void main(String[] args) {
                        // Create a Scanner object
                        Scanner obj = new Scanner(System.in);
                        System.out.println("Enter username");
                        // Read user input
                        String userName =obj.nextLine();
                        // Output user input
                        System.out.println("Username is: " + userName);
              }
 }
```

Output :-

```
Enter username
My name is Khan
Username is: My name is Khan
```

```
import java.util.Scanner;
public class Example2 {
              public static void main(String[] args) {
                        Scanner myObj = new Scanner(System.in);
                        System.out.println("Enter name, age and salary:");
                        // String input
                        String name = myObj.nextLine();
                       // Numerical input
                       int age = myObj.nextInt();
                       double salary = myObj.nextDouble();
                       // Output input by user
                       System.out.println("Name: " + name);
                       System.out.println("Age: " + age);
                       System.out.println("Salary: " + salary);
                  }
}
```

Output :-

```
Enter name, age and salary:
Mayur
25
100000
Name: Mayur
Age: 25
Salary: 100000
```

Sumago Infotech Pvt. Ltd.

Strive With Technology...!

Close Scanner

- Java Scanner class uses the "Close ()" method to close the Scanner.
- Input.close();

Do you need to close a Scanner class?

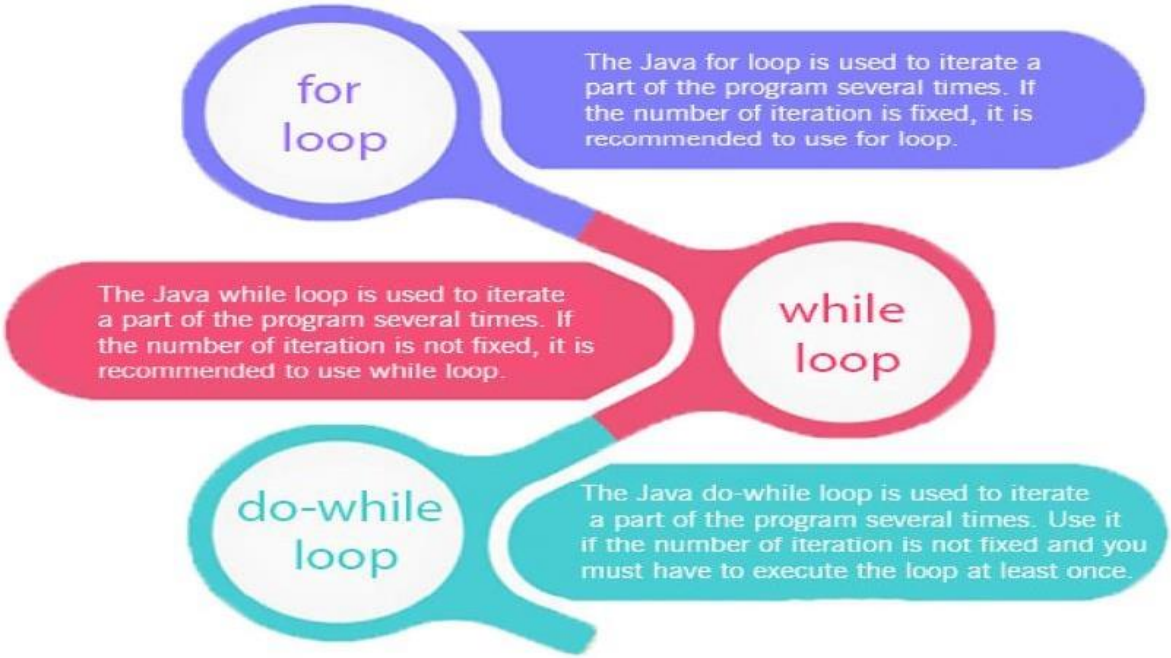
- It is better but not mandatory to close the Scanner class as if it is not closed, the underlying Readable interface of the Scanner class does the job for you. The compiler might flash some warning though if it is not closed.
- It is a good programming practice to explicitly close the Scanner using the Close () method once you are done using it.

Loops in JAVA

- Loop statements are used to execute the set of instructions in a repeated order.
- The set of instructions execute depends upon a particular condition and In programming, sometimes we need to execute the block of code repeatedly while some condition evaluates to true.

In Java, we have three types of loops that execute similarly.

1. for loop
2. while loop
3. do-while loop



Java for loop

- The Java for loop is used to iterate a part of the program several

times. If the number of iterations is fixed, it is recommended to use for loop.

- It enables us to initialize the loop variable, check the condition, and increment/decrement in a single line of code. We use the for loop only when we exactly know the number of times, we want to execute the block of code.

Syntax:

```
for (initialization, condition, increment/decrement)
{
        statements;
}
```

1. **Initialization:**
   - It is the initial condition which is executed once when the loop starts. Here, we can initialize the variable, or we can use an already initialized variable.

2. **Condition:**
   - It is the second condition which is executed each time to test the condition of the loop. It continues execution until the condition is false. It must return boolean value either true or false.

3. **Increment/Decrement:**
   - It increments or decrements the variable value.

4. **Statements:**
   - The statement of the loop is executed each time until the second condition is false.

Q. Print 1 to 10

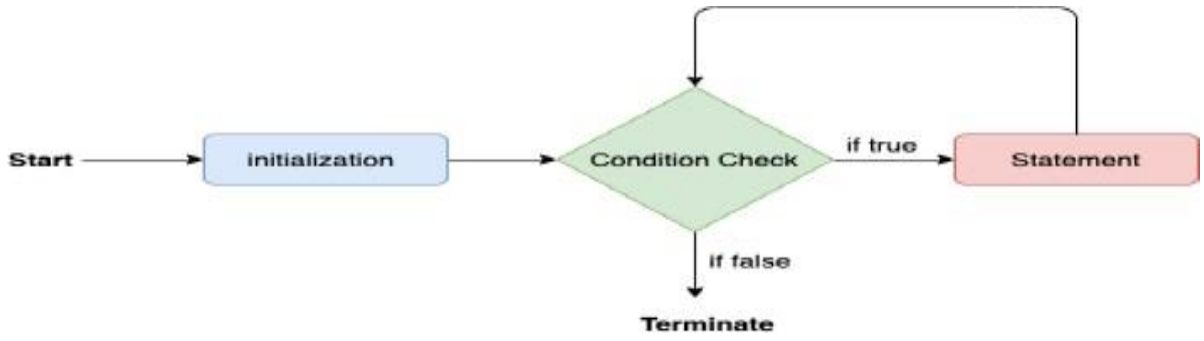
```
public class ForExample {
    public static void main(String[] args) {
        //Code of Java for loop
        for(int i=1;i<=10;i++)
        {
            System.out.println(i);
        }
    }
}
```

Output – 1

....

10

FlowChart:



Java Nested for Loop

- If we have a for loop inside the another loop, it is known as nested for loop.
- The inner loop executes completely whenever outer loop executes.

Syntax:

```
for (initialization, condition, increment/decrement)
{
        for (initialization, condition, increment/decrement)
        {
                statements;
        }

}
```

Example:

```
public class NestedForExample {
    public static void main(String[] args) {
        //loop of i
        for(int i=1;i<=3;i++)
        {
            //loop of j
            for(int j=1;j<=3;j++)
            {
                System.out.println(i+" "+j);
            }//end of i
        }//end of j
    }
}
```

Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

Java while loop

The while loop is also used to iterate over the number of statements multiple times. However, if we don't know the number of iterations in advance, it is recommended to use a while loop.

- The Java while loop is used to iterate a part of the programs repeatedly until the specified Boolean condition is true. As soon as the Boolean condition becomes false, the loop automatically stops.
- The initialization and increment/decrement doesn't take place inside the loop statement in while loop.
- It is also known as the entry-controlled loop since the condition is checked at the start of the loop.

Syntax:

```
1.    while(condition)
      {
            statements;
            Increment / decrement statement;
      }
```

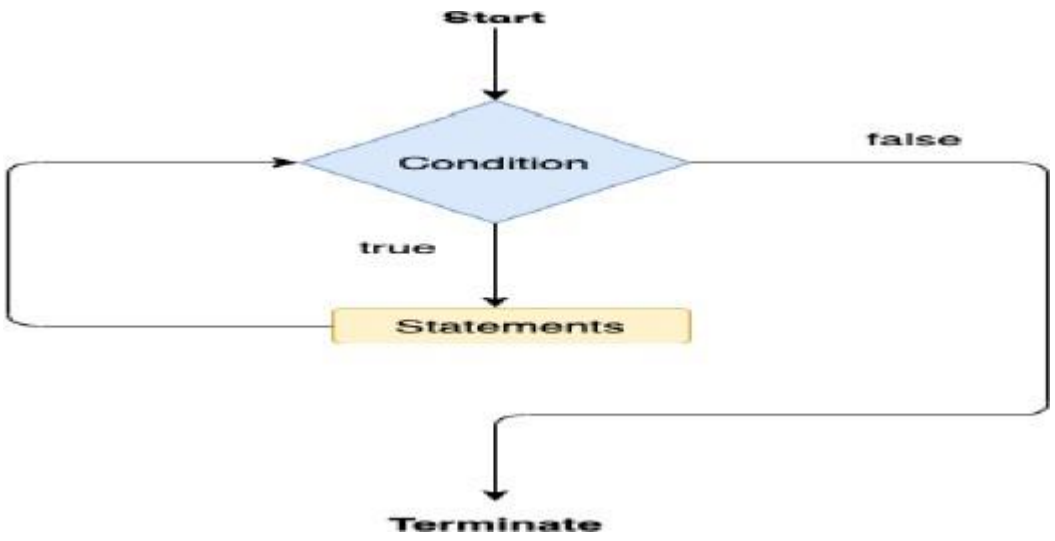
Example:

we print integer values from 1 to 10.

```
public class WhileExample {
    public static void main(String[] args) {
        int i=1;
        while(i<=10)
        {
            System.out.println(i);
            i++;
        }
    }
}
```

Output – 1
.
.
10

Flowchart:



Java do-while loop

- The Java do-while loop is used to iterate a part of the program repeatedly, until the specified condition is true.
- If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use a do-while loop.

1. Java do-while loop is called an **exit control loop**.
2. Therefore, unlike while loop and for loop, the do-while check the condition at the end of loop body. The Java do-while loop is executed at least once because condition is checked after loop body.

Syntax:

```
do
{
        //statements;
} while (condition);
```

Example:

```
we print integer values from 1 to 10.
        public class DoWhileExample {
                public static void main(String[] args) {
                        int i=1;
                        do
                        {
                                System.out.println(i);
                                i++;
                        }while(i<=10);
                }
        }
```

Output – 1
.
.
10

Java for Loop vs while Loop vs do-while Loop

Comparison	for loop	while loop	do-while loop
Introduction	for loop is a control flow statement that iterates a part of the programs multiple times.	while loop is a control flow statement that executes a part of the programs repeatedly on the basis of given boolean condition.	do while loop is a control flow statement that executes a part of the programs at least once and the further execution depends upon the given boolean condition.
When to use	If the number of iteration is fixed, it is recommended to use for loop.	If the number of iteration is not fixed, it is recommended to use while loop.	If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use the do-while loop.

Syntax	for(init;condition;incr/decr){
	// code to be executed
	}
		while(condition){
		//code to be executed
		}
			do{
			//code to be executed
			}while(condition);

Syntax for infinitive loop	for(;;){
	//code to be executed
	}
		while(true){
		//code to be executed
		}
			do{
			//code to be executed
			}while(true);

OOPs (Object-Oriented Programming System)

Object means a real-world entity such as a pen, chair, table, computer, watch, etc. Object-Oriented Programming is a methodology to design a program using classes and objects.

Pillars/feature of OOPs:-

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

Object:-

Any entity that has state and behaviour is known as an object. For example a chair, pen, table, keyboard, bike, etc.

1. It can be physical or logical.
2. An Object can be defined as an instance of a class.
3. An object contains an address and takes up some space in memory.
4. Objects can communicate without knowing the details of each other's data or code.

Example:

A table/bike is an object because it has states like color, name etc. as well as behaviors like running etc.

1. **State**: represents the data (value) of an object.
2. **Behaviour**: represents the behaviour (functionality) of an object such as deposit, withdrawal, etc.

Class

- Collection of objects is called class. It is a logical entity.
- A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance

- Inheritance in Java is a mechanism in which one class acquires all the properties and behaviours of another class with the help of extends keyword.
- Inheritance is one of the most important Object-oriented programming language system.
- Also, we can say that subclass can acquire the properties of superclass with extends keyword
- Inheritance represents the IS-A relationship which is also known as a parent child relationship.

Why use inheritance in java

1. For Code Reusability.

2. For code optimization

Terms used in Inheritance

- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent

class.

Types of Inheritance

1. Single Level Inheritance
2. Multi-Level Inheritance
3. Multiple Inheritance
4. Hierarchical Inheritance

1. Single Level Inheritance.

- For single level inheritance two classes are mandatory.
- It is an operation where inheritance takes place between two classes to perform Single Level Inheritance.
- In short we can say that when a class inherits another class it is known as single level Inheritance.

```
package SingleLevelInh;

public class Bank {

        public void account()
        {
                System.out.println("This is the process for accout creation");
        }

        public void withdraw()
        {
                System.out.println("This is the process for withdraw");
        }
    }
```

```
package SingleLevelInh;
public class HDFCClass extends Bank{

        public void bankName()
        {
                System.out.println("The name of bank is HDFC");
        }
}
```

Sumago Infotech Pvt. Ltd.

Strive With Technology...!

```
package SingleLevelInh;
public class MasterClass {
	public static void main(String[] args) {
		HDFCClass obj=new HDFCClass();
		obj.bankName();
		obj.account();
		obj.withdraw();

		Bank obj1=new Bank();
		obj1.account();
		obj1.withdraw();
	}
}
```

Output:-

```
The name of bank is HDFC
This is the process for accout creation
This is the process for withdraw
This is the process for accout creation
This is the process for withdraw
```

MultiLevel Inheritance

- Multilevel inheritance takes place between three or more classes.
- In multilevel inheritance one subclass can acquire the property of another superclass and this phenomenon continues so that we known as "Multi level Inheritance"
- In short when there is a chain of inheritance it is also known as multilevel inheritance.

```java
package MUltiLevelIn;
public class ClassA {
    public void add()
    {
            int a=10;
            int b=20;
            System.out.println("I am from Class A");
            System.out.println("Addition result is:- "+(a+b));
    }
}
```

```java
package MUltiLevelIn;
public class ClassB extends ClassA{
        //ClassB have properties ClassB+ClassA
        public void sub()
        {
                int a=10;
                int b=20;
                System.out.println("I am from Class B");
                System.out.println("sub result is:- "+(b-a));
        }
}
```

```java
package MUltiLevelIn;
public class ClassC extends ClassB{
        //preperties
        public void multi()
        {
                int a=10;
                int b=20;
                System.out.println("I am from Class C");
                System.out.println("multi result is:- "+(a*b));
        }
}
```

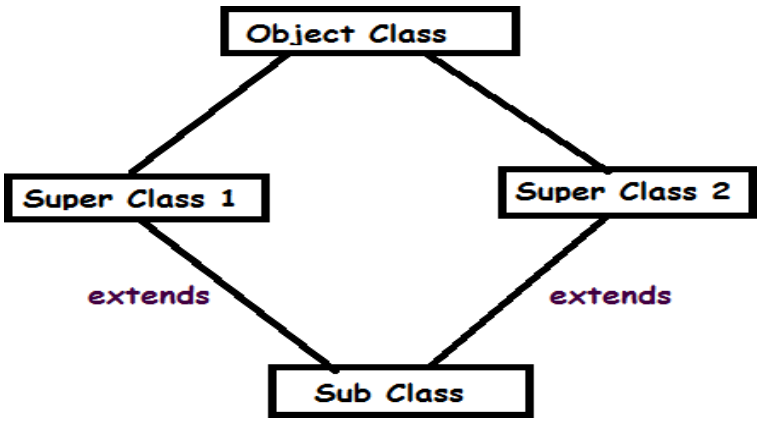
```
package MUltiLevelIn;
public class MasterClass {
	public static void main(String[] args) {

		ClassC c=new ClassC();
		c.add();
		c.sub();
		c.multi();

		ClassB b=new ClassB();
		b.add();
		b.sub();

		ClassA a=new ClassA();
		a.add();
	}
}
```

```
Output:-
I am from Class B
sub result is:- 10
I am from Class C
multi result is:- 200
I am from Class D
div result is:- 2
I am from Class B
sub result is:- 10
I am from Class C
multi result is:- 200
I am from Class B
sub result is:- 10
I am from Class A
Addition result is:- 30
```

Multiple Inheritance

- In multiple inheritance one subclass acquiring properties of two superclasses at a same time so this known as multiple inheritance
- Java doesn't support multiple inheritance because it results diamond ambiguity problem
- It means the diamond like structure get created in jVM and object variable get confused for who need to inherited the property of superclass Ito subclass
- SuperClass of all the classes is object class so there diamond ambiguity forms
- To reduce the complexity and simplify the language multiple inheritance not supported in java

Diagram:



Hierarchical Inheritance

- Hierarchical Inheritance takes place between one superclass and multiple subclass
- So the property of superclass can be acquired by multiple subclass this is kowm as Hierarchical Inheritance
- The two or more classes inherits a single class it is known as Hierarchical Inheritance
- This inheritance takes place between one superclass and multiple subclass

```java
package Hierachicaln;
public class Bank{
    public void mainMethod()
    {
            System.out.println("This is main method from bank");
    }
}
```

```java
package Hierachicaln;
public class HDFC extends Bank {
        public void hdfcMethod()
        {
                System.out.println("This is hdfc method from HDFC class");
        }
}
```

```java
package Hierachicaln;
public class Axis extends Bank{
        public void axisMethod()
        {
                System.out.println("This is Axis method from axis class");
        }
}
```

```java
package Hierachicaln;
public class SBI extends Bank{
```

```
        public void sbiMethod()
        {
                System.out.println("This is SBI method from SBI CLASS");
        }
}
```

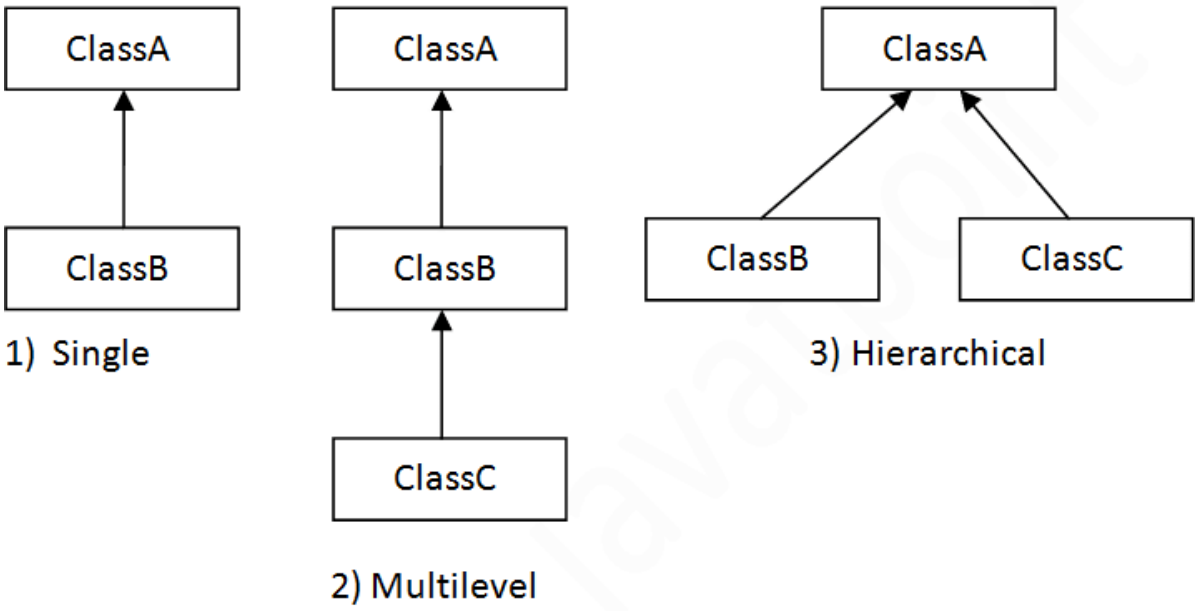
```
package HierachicalIn;
public class MasterClass {
        public static void main(String[] args) {
                Bank b=new Bank();
                b.mainMethod();

                HDFC h=new HDFC();
                h.mainMethod();
                h.hdfcMethod();

                SBI s=new SBI();
                s.mainMethod();
                s.sbiMethod();

                Axis a=new Axis();
                a.mainMethod();
                a.axisMethod();
        }
}
```

Output:-

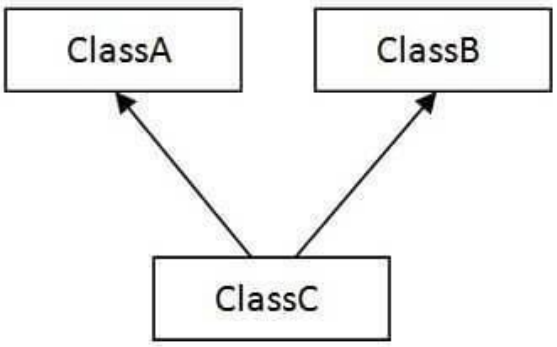
```
This is main method from bank
This is main method from bank
This is hdfc method from HDFC class
This is main method from bank
This is SBI method from SBI CLASS
This is main method from bank
This is Axis method from axis class
```

Hybrid Inheritance

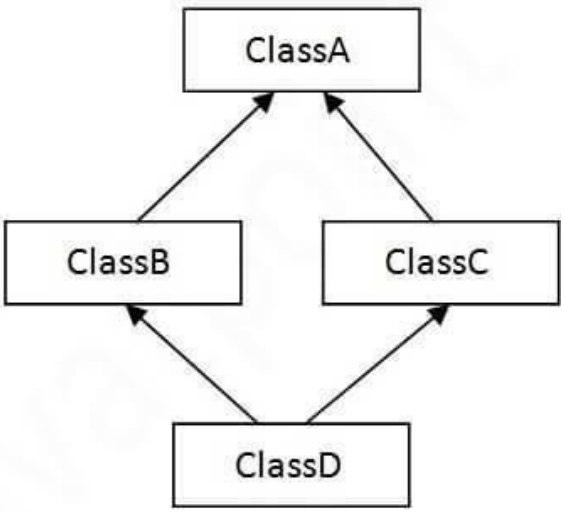
Hybrid Inheritance is a combination of Inheritances. Since in Java Multiple Inheritance is not supported directly we can achieve Hybrid inheritance also through Interfaces only.



Note: Multiple inheritance is not supported in Java through class.



4) Multiple



5) Hybrid

Polymorphism

- Polymorphism in Java is a concept by which we can perform a single action in different way.
- Polymorphism is derived form 2 Greek works: Ploy and morphs
  - "Ploy" means many
  - "morphs" mean forms
- One object shows different behaviour at different stages of life cycle as called as "Polymorphism"

There are two types of polymorphism in JAVA

1. Compile-Time polymorphism
2. Run-Time polymorphism

1. Compile Time Polymorphism

In compile time polymorphism method declaration is going to get bonded to it's definition at compile time based on arguments so that we called it as "Compile time polymorphism"

- At compile-time, Java knows which method to call by checking method signature so this is called compile time polymorphism.
- Also called as Static binding or Early binding.
- As binding takes place during compilation time so it is known as "Early binding"
- Method overloading is An example of compile time polymorphism.

```
package Ex1poly;

public class Example1MethodOverload {

	public void test() //1
	{
		System.out.println("This is test method with no arg");
	}
	public void test(String a) //2
	{
		System.out.println("This is test method with String arg");
	}
	public void test(String b, int c) //3
	{
		System.out.println("This is test method with no arg");
	}
```

```
	public void test(int d, String e) //4
	{
		System.out.println("This is test method with no arg");
	}

	public static void main(String args [])
	{
		Example1MethodOverload obj=new Example1MethodOverload();
		obj.test("hs", 10);
	}
}
```

2. Run time polymorphism

In Run time polymorphism method declaration is going to get bonded to its definition during run time or execution time based on object creation so that we called it as "runtime polymorphism".

- Runtime polymorphism also called as "Dynamic biding" or late binding
- Method overriding is an example of run-time polymorphism

```
package Ex1poly;
public class Test1 {
    public void test(int a)
    {
            System.out.println("Test method from test1");
    }
    public void display()
    {
            System.out.println("display method from test1");
    }
}
```

```
package Ex1poly;
public class Test2 extends Test1 {
public void test(int b)
{
System.out.println("Test method from test2");
}
public static void main(String args[])
{
```

```
t2.display();

Test1 t1=new Test1();
t1.test(20);
}
}
```

```
Test method from test2
display method from test1
Test method from test1
```

Sumago Infotech Pvt. Ltd.

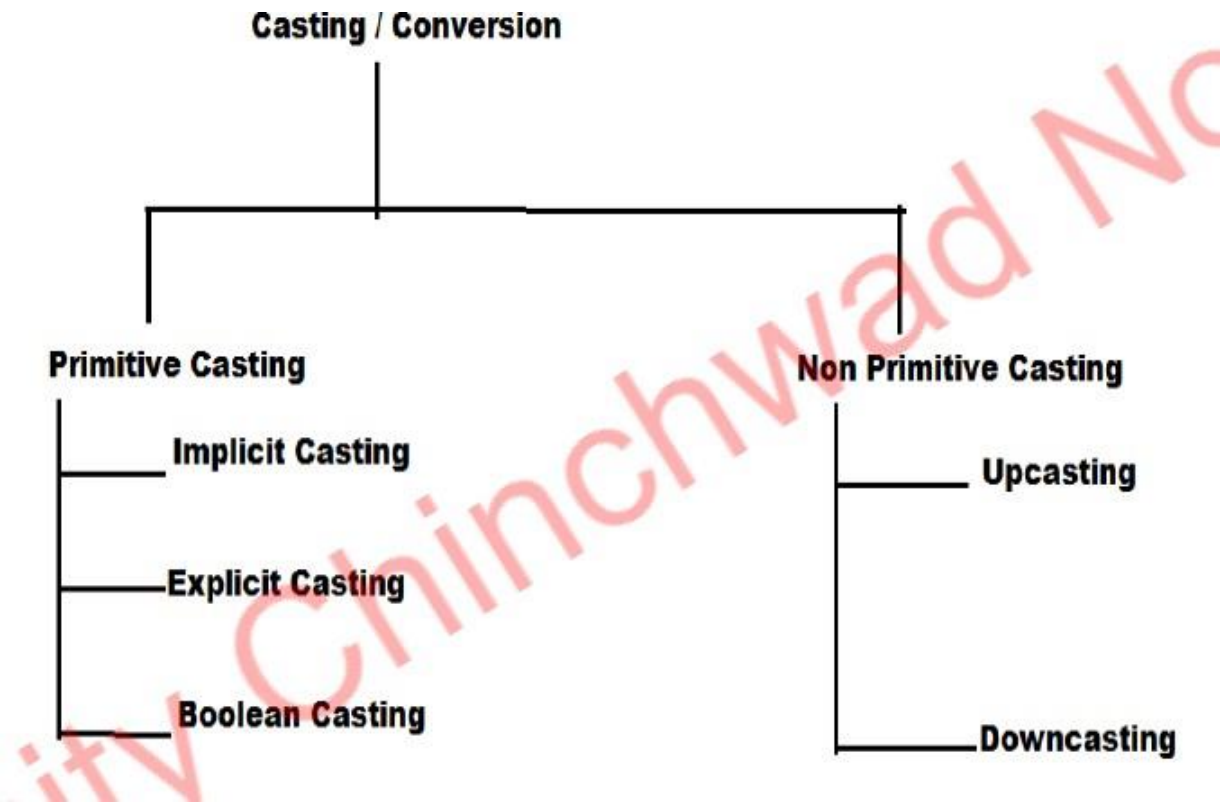
Strive With Technology...!

Difference between method overloading and method overriding in java

No.	Method Overloading	Method Overriding
1	Method overloading is used *to increase the readability* of the program.	Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class.
2	Method overloading is performed *within class*.	Method overriding occurs *in two classes* that have IS-A (inheritance) relationship.
3	In case of method overloading, *parameter must be different*.	In case of method overriding, *parameter must be same*.
4	Method overloading is the example of *compile time polymorphism*.	Method overriding is the example of *run time polymorphism*.
5	In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter.	*Return type must be same or covariant* in method overriding.

Conversion / Casting

Converting one type of information into another type of information is called Conversion or Casting.



Primitive Casting

- Converting one data type of information into another data type information is known as Primitive Casting.
- Primitive casting happens between 2 data types.
- There are 3 types of Primitive Casting.
    1. Implicit Casting
    2. Explicit Casting

3. Boolean Casting

Implicit Casting

- Converting lower data type of information into higher data type is called implicit casting.
- Also known as Automatic casting or Widening conversion.
- There is no loss of data.

Program

```
public class ImplicitCasting {

    public static void main(String[] args) {

        byte b = 10;
        short h = b;
        System.out.println(h);

        short s = 100;
        int a = s;
        System.out.println(a);


        int n = 1010;
        long l = n;
        System.out.println(l);

    }
}
```

Explicit Casting

- Converting of higher data type information into lower data type information is called explicit casting.
- Also known as Forceful casting or Narrowing conversion.
- In this casting there may be loss of data.

Program

```
public class ExplicitCasting {

    public static void main(String[] args) {


        int i = 100000;
        short b = (short)i;
        System.out.println(b);

        float a = 57.99f;
        short s = (short)a;
        System.out.println(s);

        double d = 454545454;
        float f = (float)  d;
        System.out.println(f);

    }

}
```

OUTPUT—

```
-31072
57
4.5454544E8
```

Boolean Casting

- Java does not support Boolean Casting. It is considered as a incompatible type of casting , because we cannot convert true into false or vice versa.

Non Primitive Casting

- Non primitive casting means converting one type of class into another type of class is known as non primitive casting.
- It is classified into 2 types
    1. Upcasting
    2. Downcasting

Upcasting

- Assigning subclass property into superclass is known as Upcasting.
- Before performing upcasting inheritance operation takes place. Where properties of superclass are inherited into subclass.
- Programmers can declare new properties in subclass.
- At the time of upcasting the properties which are inherited from superclass are only eligible for upcasting. New properties declare are not eligible.
- Syntax : Superclass refVar = new Subclass();

Programs:

Condition 1 – With Overriding methods

```
public class Animal {

     String name = "AnimalVar";

     public void nature()
     {
          System.out.println("Animal");
     }

}
```

```
public class Shark extends Animal{

     String name = "SharkVar";

     public void nature()
     {
          System.out.println("Aquatic Animal");
     }

}
```

```
public class UpcastingConcept1 {

    public static void main(String[] args) {
        Animal a = new Shark();
        a.nature();
        System.out.println(a.name);

    }

}

OUTPUT—
Aquatic Animal
AnimalVar
```

Condition 2 – Without Overriding methods

```
public class Father {

    int age = 60;

    public void home()
    {
        System.out.println("Father's house ");
    }

    public void farm()
    {
        System.out.println("Father's farm ");
    }

}
```

```
public class Son extends Father{

	int age = 30;

	public void bike()
	{
		System.out.println("Son's bike");
	}


	public void job()
	{
		System.out.println("Son's job");
	}
}
```

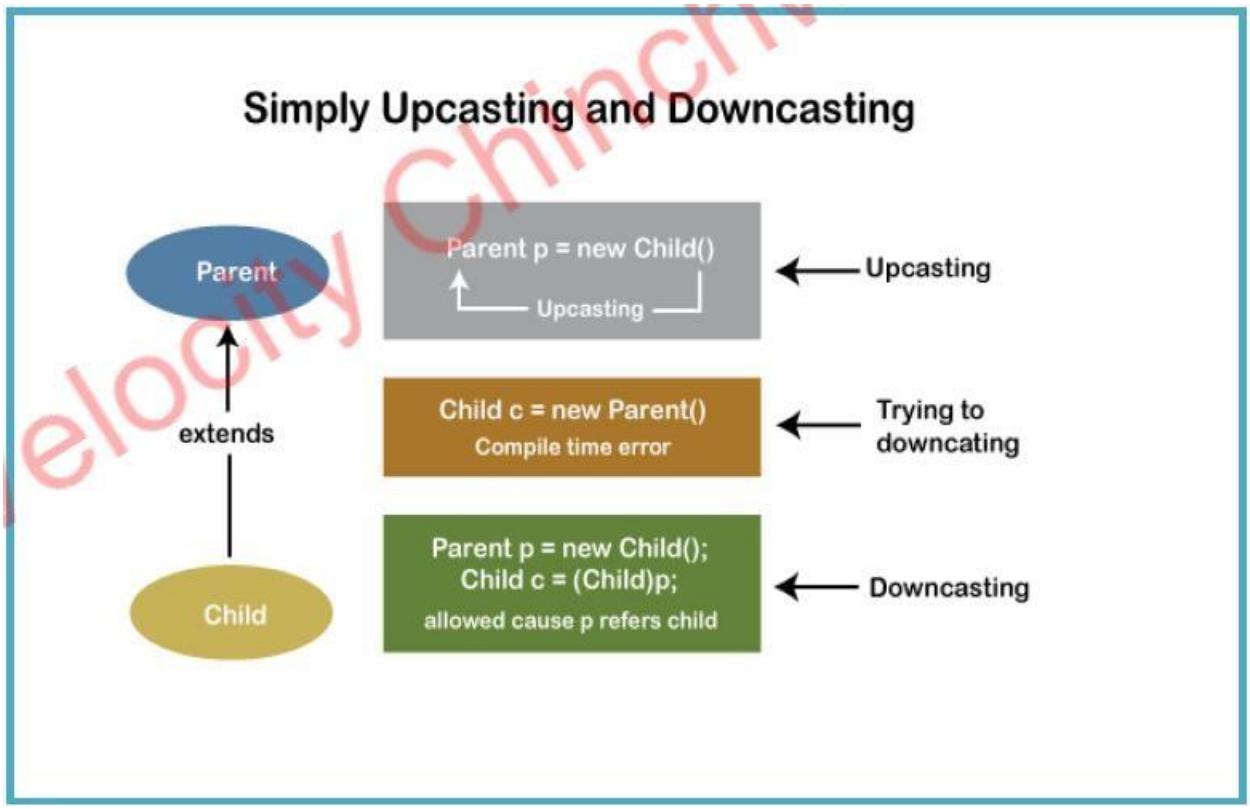
```
public class UpcastingConcept2 {

	public static void main(String[] args) {

		Father f = new Son();
		f.home();
		f.farm();
		System.out.println(f.age);

	}

}
```

OUTPUT–

```
Father's house
Father's farm
60
```

Downcasting

- Assigning super class properties to sub class after Upcasting is known as Downcasting.
- In java downcasting is rarely used because java doesn't support downcasting directly.
- Before performing downcasting compulsory upcasting should be performed.
- We can perform downcasting in explicit way i.e forcefully.
- So there may be loss of data or information.



Access Modifiers in Java

The Access Modifiers

- The access modifiers in Java defines the accessibility or scope of a field, method, constructor, or class.
- We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

1. **Private:**
   The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

2. **Default:**
   The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

3. **Protected:**
   The access level of a protected modifier is within the package and outside the package through child class (Inheritance). If you do not make the child class, it cannot be accessed from outside the package.

4. **Public:**
   The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

1. Private

- The access level of a private modifier is only within the class.
- It cannot be accessed from outside the class.

In this example, we have created two classes A and Simple. A class contains private data members and private methods. We are accessing these private members from outside the class, so there is a compile-time error.

```
public class A {
        private int data=40;
        private void msg()
        {
              System.out.println("Hello java");}
         }

 public class Simple{
        public static void main(String args[])
        {
            A obj=new A();
            System.out.println(obj.data);//Compile Time Error
            obj.msg();//Compile Time Error
        }
 }
```

Role of Private Constructor

- If you make any class constructor private, you cannot create the instance of that class from outside the class.

  For example:

```
class A{
        private A()         //private constructor
        {
        void msg(){System.out.println("Hello java");}
        }
public class Simple{
         public static void main(String args[])
         {
             A obj=new A();//Compile Time Error
         }
}
```

2. Default

- If you don't use any modifier, it is treated as default by default.
- The default modifier is accessible only within package.
- It cannot be accessed from outside the package.
- It provides more accessibility than private. But it is more restrictive than protected, and public.

In this example, we have created two packages pack and mypack. We are accessing the A class from outside its package, since A class is not public, so it cannot be accessed from outside the package.

```
//save by B.java
package pack;
class A{
        void msg(){System.out.println("Hello");}
}
//save by B.java
package mypack;
import pack.*;
class B{
        public static void main(String args[])
        {
             A obj = new A();//Compile Time Error
             obj.msg();//Compile Time Error
        }
 }
```

In the above example, the scope of class A and its method msg() is default so it cannot be accessed from outside the package.

3. Protected

- The protected access modifier is accessible within package and outside the package but through inheritance only.
- The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.
- It provides more accessibility than the default modifier.

In this example, we have created the two packages pack and mypack. The A class of pack package is public, so can be accessed from outside the package. But msg method of this package is declared as protected, so it can be accessed from outside the class only through inheritance.

```
//save by A.java
package pack;
public class A{
      protected void msg()
      {
          System.out.println("Hello");}
      }
//save by B.java
 package mypack;
 import pack.*;
 class B extends A{
          public static void main(String args[])
          {
          B obj = new B();
          obj.msg();
          }
 }
```

4. Public

- The public access modifier is accessible everywhere.
- It has the widest scope among all other modifiers.

```
//save by A.java
package pack;
public class A{
       public void msg()
      {
        System.out.println("Hello");}
      }
//save by B.java
 package mypack;
 import pack.*;
 class B{
          public static void main(String args[])
          {
              A obj = new A();
              obj.msg();
          }
 }
```

Abstraction in Java

- Abstraction is a process of hiding the implementation details / code / information and showing only functionality to the end user.
- Hiding internal functionality and showing external functionality to users.
- Another way, it shows only essential things to the user and hides the internal details

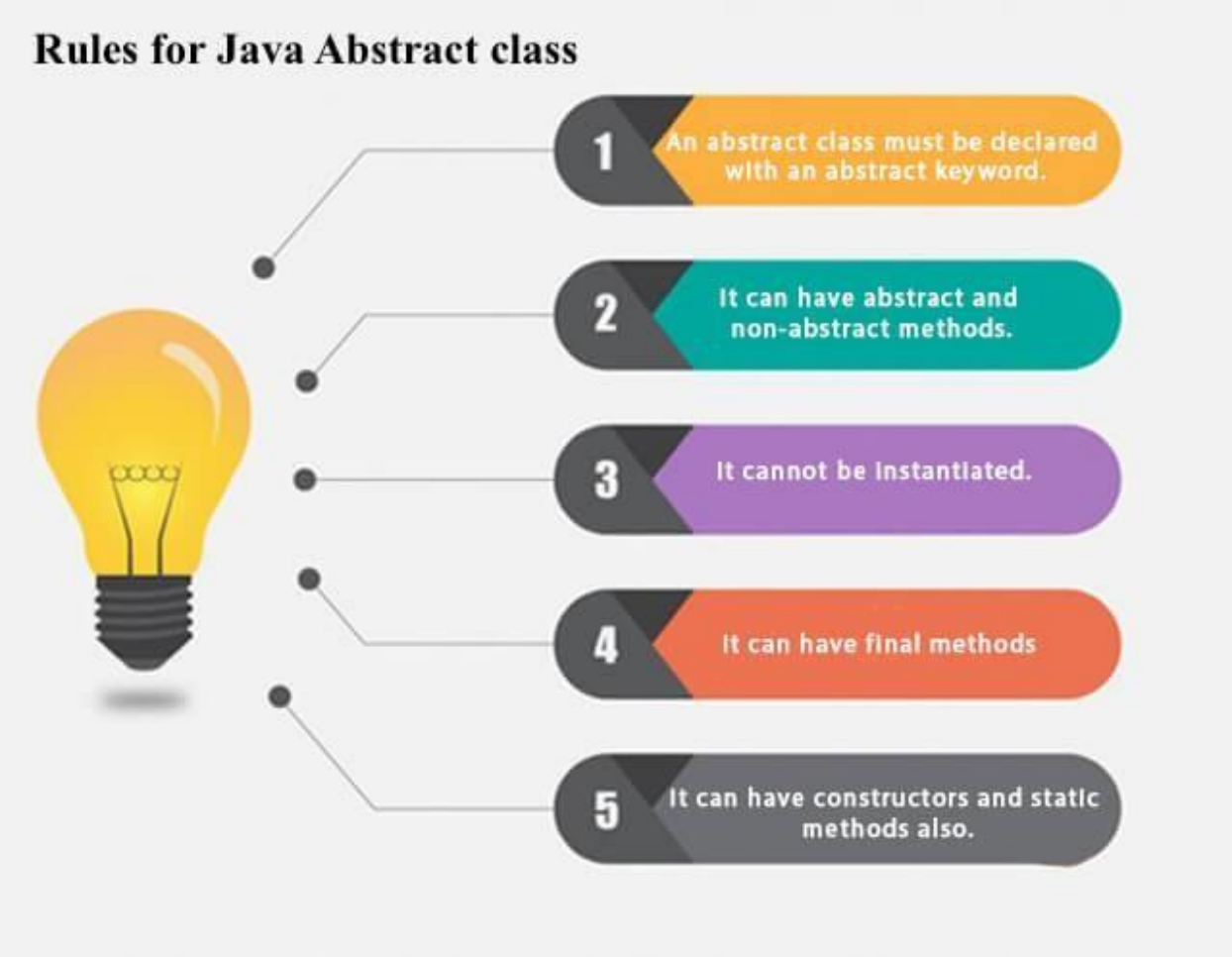
Example : Sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

Abstract class in Java

- A class which is declared with an abstract keyword is known as an abstract class.
- Abstract class has abstract and non-abstract methods (i.e. incomplete and complete methods).
- Abstract Method - Method declaration is present but don't have method body(implementation).
- It needs to be extended and its method implemented. It cannot be instantiated.



Points to Remember

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

Abstract Method in Java

- A method which is declared with an abstract keyword and does not have implementation is known as an abstract method.
- A method does not have a method body.

Example of abstract method

```
abstract void printStatus(); //no method body and abstract
```

Concrete Class

- The class which is responsible to implement all the abstract methods from the abstract class.
- We can't create object of abstract class to create object of abstract class their is approach called as "concrete class".
- An abstract class can have a data member, abstract method, method body (non-abstract method), constructor, and even main() method.

Note:

- If there is an abstract method in a class, that class must be abstract.
- If you are extending an abstract class that has an abstract method, you must either provide the implementation of the method or make this class abstract.

Interface in Java

Interface :

- The interface in Java is a mechanism to achieve abstraction
- It is used to achieve abstraction and multiple inheritance in Java.
- An interface in Java is a blueprint of a class.
- There will be only abstract methods in the Java interface, not method body.
- It cannot be instantiated just like the abstract class.
- When an interface inherits another interface extends keyword is used whereas class use implements keyword to inherit an interface.

Why use the Java interface?

There are mainly three reasons to use interfaces. They are given below.

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling


It is used to achieve abstraction.
1
2
By interface, we can support the functionality of multiple inheritance.
It can be used to achieve loose coupling.
3

Feature of Interface/Point to remember

- Data member declared Inside interface by default static and public
- Methods declared inside interface are default public and abstract
- Constructor concept is not present inside interface
- Object of interface can not be created
- Interface supports multiple inheritance
- Since Java 8, we can have default and static methods in an interface.
- Since Java 9, we can have private methods in an interface.

How to declare an interface?

- An interface is declared by using the interface keyword.
- It provides total abstraction; means all the methods in an interface are declared with the empty body and public and abstract by default, and all the fields/variables are public, static and final by default.

The Java compiler adds **public and abstract** keywords before the interface method. And it adds **public, static and final** keywords before data members.

Syntax:

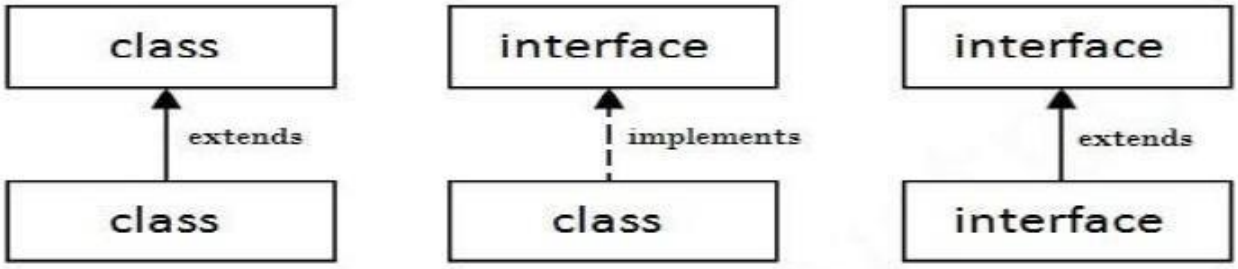
```
interface <interface_name>{
    // declare constant fields
    // declare methods that abstract
    // by default.
}
```

Implementation Class

- A class which provides implementation for all the incomplete methods of interface with the help of implement keyword is known as Implementation class
- At the time of Implementation declared method with public access modifiers compulsory

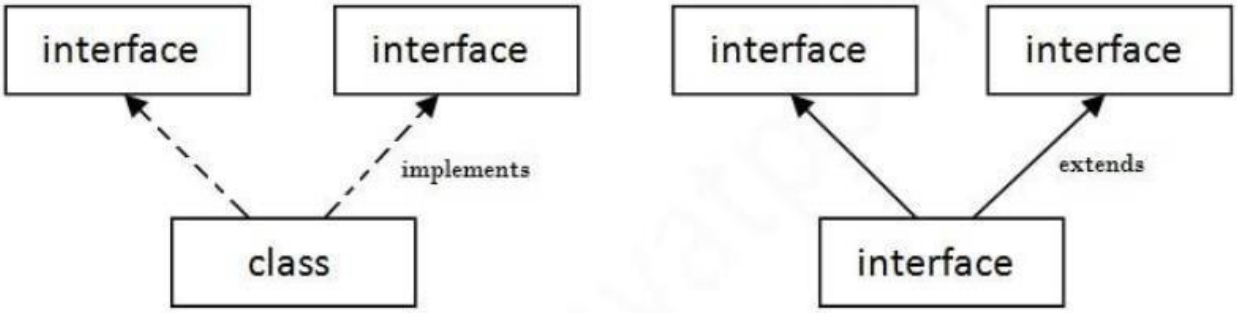
The relationship between classes and interfaces

- A class extends another class, an interface extends another interface, but a class implements an interface.



Multiple inheritance in Java by interface

- If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.



Multiple Inheritance in Java

Encapsulation

- The process of wrapping the data and corresponding methods into a single unit is called Encapsulation.
- Examples are Medicine capsule , School Bag.

- If any component follows data hiding and abstraction then it is called encapsulation.
  Encapsulation = Data Hiding + Abstraction

Steps to achieve encapsulation

1. We have to declare variables of class as private. (By declaring variable as private we have achieved data hiding means outside person cannot access this variable).
2. We have to use public getters and setters method to modify and view the variable values.

Few points about encapsulation

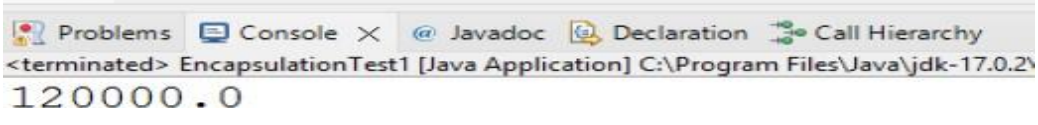
- User would never know of what is going on in the background, they would be only aware of update the field by set method and read a field by get method.
- Set and get words used in the method names are only for naming

purpose so that programmer should understand which is set and get method.

- **Advantages are :-**
    1. Encapsulated class is easy to test so it is better for doing unit testing.
    2. It provides security.

```
package encapsulationPkg;

public class EncapsulationTest1 {
    private double balance; //GlobalorInstance

    //Setter
    public void setBalance(double balance) //Local
    {
        this.balance = balance;
    }

    //getter
    public double getBalance()
    {
        return balance;
    }

    public static void main(String[] args) {
        EncapsulationTest1 et = new EncapsulationTest1();
        et.setBalance(120000);
        et.getBalance();

        double z = et.getBalance();
        System.out.println(z);
    }
}
```

Output:



Super Keyword

"**super**" keyword is used to access global variable from super class or different class

Example:

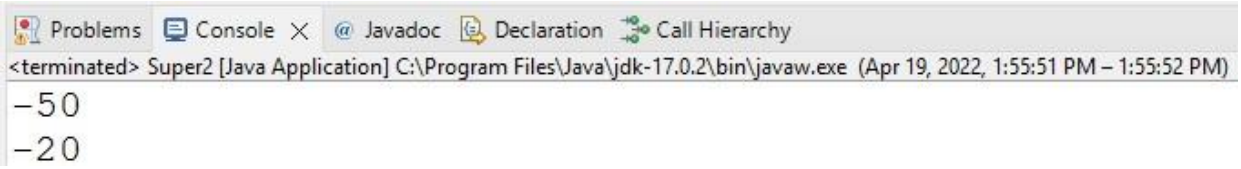
Super Class:

```
package superKeywordPckg;

public class Super1 {

    int a = 10;
    int b = 30;

    public void test()
    {
    }
}
```

Sub Class:

```
package superKeywordPckg;

public class Super2 extends Super1{
    int a = 50;
    int b = 100;

    public void test2()
    {
        System.out.println(a-b);
        System.out.println(super.a - super.b);
    }

    public static void main(String[] args) {
        Super2 s = new Super2();
        s.test2();
    }
}
```

Output:


Problems
Console
Javadoc
Declaration
Call Hierarchy
<terminated> Super2 [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Apr 19, 2022, 1:55:51 PM – 1:55:52 PM)
-50
-20