Introduction of Java -

1. Author :- James Gosling
2. Vendor - Sun Microsystem (Now merged in Oracle)
3. Initial Name - Oak
4. Programming Type - Object Oriented and Open Source
5. Present Name - Java
6. Project Name - Project Green
7. Slogan - WORA (Write Once Run Anywhere)
8. logo - Coffee cup with Saucer
9. Extension - .java, .class, .jar
10. First Version - JDK 1.0
11. Present Version - JDK 22

What is Java?

Java is a high-level, object oriented programming language developed by Sun microsystems in the mid-1990s, now owned by Oracle Corporation. It is designed to be platform independent, meaning that Java code can run on any device that has a Java Virtual Machine installed. Java is known for its "Write Once, run anywhere" capability, which is made possible through the JVM. It is widely used for building applications, from mobile apps to large scale enterprise systems, web Applications and more.

History = Java was developed by James Gosling at sun microsystems and launched in 1995. It gained popularity for its platform independence, and in 2010, Oracle took over its development after ~~acq~~ acquiring Sun Microsystem.

Features of Java -

① Platform Independent - Java is a platform independent language because it does not depend on OS or other things. Every system which have JVM installed are able to run java programs.

② Object Oriented - Java is a object-oriented programming language means it supports concept like inheritance, Polymorphism, Abstraction and Encapsulation. This make code modular, flexible and easier to manage.

③ Simple and Familiar - Java's syntax is straight forward and easy to learn. Especially for those who are familiar with C or C++. This makes the code modular, flexible and easier to manage.

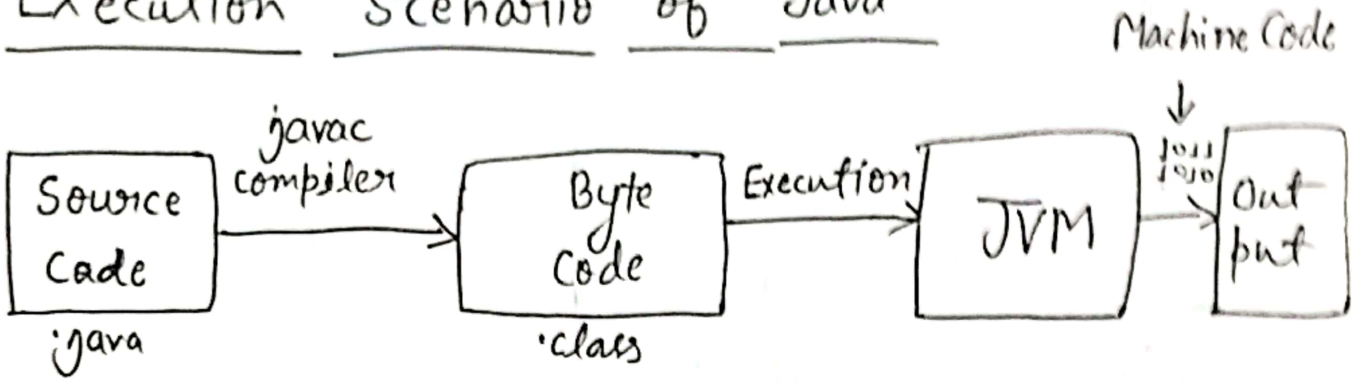
④ Secure - It provide a secure environment for developing application with its security manager.

⑤ Robust - Java emphasizes early error checking and runtime checking which makes it robust and less prone to crashes.

⑥ High Performance - Java Uses Just-In-Time (JIT) compilers to convert bytecode to native machine code at run-time enhancing performance.

⑦ Multi thread - Allowing developer to write program that can perform many tasks simultaneously improving the efficiency and performance of application.

Execution Scenario of Java -



1. Source Code (.java) Complied through Javac Complier into Byte code (.class).

2. Byte code (.class) executed through JVM.

3. JVM converts byte code into machine code and we get output.

Components of Java -

1. JDK - Java Development Kit is software package which contains Java Compiler, JRE and JVM.

2. Java Compiler - Java Compiler is a type of translator which converts source code into byte code.

3. JRE - Java Runtime Environment create a layer between java application and operating system. Java application is executed under supervision of JRE.

4. JVM - Java Virtual Machine is an agent of java, which converts byte code to ~~mah~~ machine code.

Keywords -

Keywords are reserved words predefined meanings in the compiler. There are 50 keywords in Java.

Eg:- int, float, char, double, long, short, byte, Boolean, if, else, switch, break, do, default, while, for, continue, void, return, new, class, interface, abstract etc.

Class - A class is a blueprint of objects. It is a container of variables and methods.

Objects - Object is an instance of a class that contains both data and methods to manipulate that data.

Package - A package is a collection of classes, interfaces and subpackages.

Variable - Variables are value containers that create space in memory.

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

String - String is a sequence of characters. In Java String is a class.

public static void main (String [] args) -

public - A keyword which works as an access modifier.

static - A keyword which works as modifier. If you create a method with static keyword modifier then that method can be call without object.

void - A keyword which works as return type. Indicates no return value.

main() - The pre-defined method name, starting point of the program.

String [] args - This array is used to create command line input.

System.out.println(" "); –

System - Class

out - Object

println() - Method

Scanner – Scanner is a class available in java.util package.. object of Scanner class is used to take input from user.

How to take Input ? –

```
import java.util.*;
Scanner sc = new Scanner( System.in);
```

① For integar input.

```
int a;
a = sc.nextInt();
```

② For Double Input

```
double b;
b = sc.nextDouble();
```

③ For Float input

```
float c = sc.nextFloat();
```

④ For String Input

```
String name = sc.nextLine
();
```

Operators

Operators are symbols which are used to perform operation on operands.

1. Arithmetic Operators -
   - Addition (+)
   - Substraction (-)
   - Multiplication (*)
   - Division (/)
   - Modulo (%)

2. Relation Operators - These operators specifies relation between two entities.
   - Greater Than (>)
   - Less Than (<)
   - Greater than or equal to (>=)
   - Less than or equal to (<=)
   - equality ([a == b])
   - Not equal to (!=)

3. Assignment Operators –

```
a = b   // Copies value of b to a
```

4. Increment Operator –

It improves/increase value of a variable by 1.

```
a++   // a = a+1;
```

5. Decrement Operator –

It decrease value of a variable by 1.

```
a--   // a = a-1;
```

6. Logical Operators–

Logical operators are used to combine two or more conditions.

- AND (&&)
- OR (||)
- NOT (!)

Decision Controls

Decision Controls are used for decision making. If you want to execute code based on some condition then you can use decision control. In Java there are following types of decision control:

1. **if Statement** - if is a keyword which works as decision control. We attach a condition with if statement. If given condition is true then code will be executed and if given condition is false then code will not be executed.

Syntax -

```
if (condition) {
    //code
}
```

2. **if-else Statement** - if else is variation of if statement. We attracted a condition with if statement. if given condition is true then if block code will executed and if given condition is false then else block code will be executed.

Syntax -

```
if (condition) {
// if block code
}
else {
// else block code
}
```

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

Syntax -

```
if (condition) {
//code 1
}
else if (condition 2) {
// code 2
}
else {
//code 3
}
```

4. Switch - Switch is a keyword which works as case control. It is used to create a menu based ~~on~~ program.

Syntax -

```
// int, char or String
switch (expression) {
case 1:
// code 1
break;
case 2;
// code 2
break;
default: // default code }
```

Loop Controls

If you have a block of code which you want to repeatedly execute up to given condition is true, then you can use a loop control. In Java there are four types of loop controls are available:

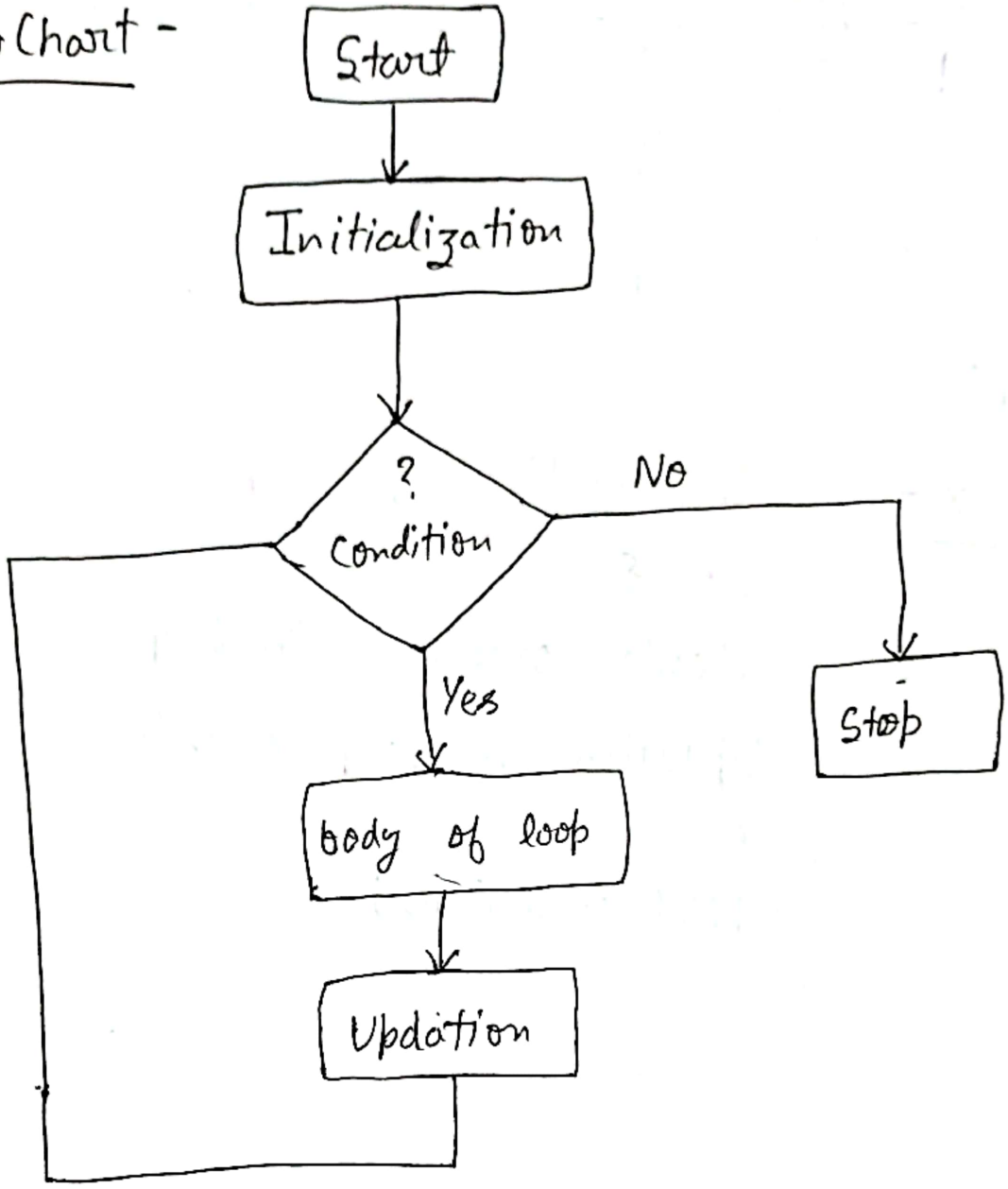
1. While Loop
2. for Loop
3. do-while Loop
4. for each loop

1. **While Loop** :- While is a keyword which works as loop control.

Syntax -

```
Initialization of loop counter;
While (condition)
{
// code
Updation of loop counter;
}
```

Flow Chart -



2. for Loop - For ~~loop~~ is a keyword which works as Loop control. The working of for loop is same as while loop but syntax is different.

Syntax - for(initialization of loop counter; condition; updation of Loop counter) {

// code

}

3. do-while Loop :- In this loop condition is tested after execution of code. So do while loop is known as exist control loop.

Syntax -

```
Initialization of Loop Counter;
do {
// code executed in Loop.
Updation of Loop counter;
}
while( condition);
```

Array

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

Declaration of Array -

datatype [] arrayname = new datatype[size];

eg:- int a[] arr = new int[10];

Initialization of Array -

```
int [] arr = {10, 20, 30, 40, 50};

arr[0] = 10;        arr[2] = 30;        arr[4] = 50;
arr[1] = 20;        arr[3] = 40;
```

Memory Representation of Array -

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
10	20	30	40	50

How to take input from user for an array ?

```
int [] arr = int [5];
int i;
Scanner sc = new Scanner (System.in);
System.out.println ("Enter five numbers : ");
for (i=0; i<5; i++) {
arr[i] = sc.nextInt();
}
```

Strings

Technically Strings is a sequence of characters In Java String is a class. The object of String class stores sequence of characters.

Syntax - String Stringname = "String value";

eg:- String name = "Hardik Srivastava";

Methods of String Class -

1. toUpperCase() - Converts String into UpperCase.
2. toLowerCase() - Converts String into lowercase.
3. length() - Find length of String.
4. equals() - Compare two Strings for equality. This method return boolean value.
5. equalsIgnoreCase() - This method also compare two Strings for equality, but this $ method avoid case sensitivity.

6. split() - This method is used to split String into sub-strings and return array of String.

eg:- String s = "He is smart.";

String[] words = s.split(" ");

words[0] = "He";

words[1] = "is";

words[2] = "smart";

7. replace() - This method is used to replace on string with another string in given string.

. charAt() - This method used to find haracters at specified location.

Method

Method is named block of code which perform specific task.

Why Method ? -

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

How to Create Method in Java ? -

```
modifier returntype methodname ( parameters) {
// code
}
```

E.g:-

```
public int sum ( int a, int b) {
return (a+b);
}
```

Types of Methods in Java -

1. **Static Method** - Static Methods are those methods which are created by using 'static' modifier. These methods are called class methods. There is no need of object to call static method.

e.g. Arrays.sort();

2. **Non-Static Method** - Non-Static methods are created without using static keyword. Non-Static methods can call by using object.

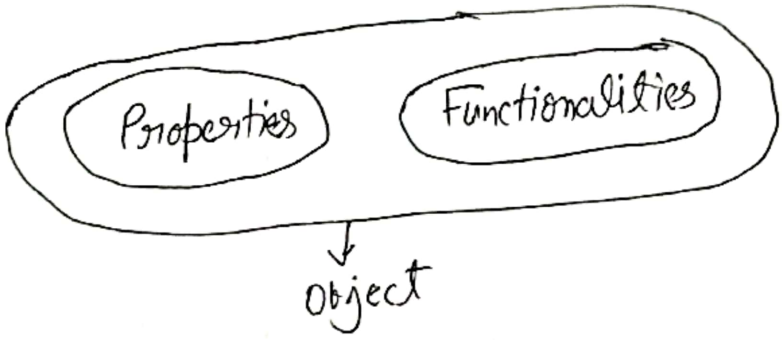
Note - You can create user defined method inside class and outside of main method.

OOPS

OOPS stands for Object Oriented Programming System. It is a mechanism of software development. The oops has four pillars -

1. Abstraction - Abstraction is a mechanism to show only essential functionalities and hide all other functionalities of an object.

2. Encapsulation - Encapsulation is a mechanism to wrap properties and functionalities in a single unit. That single unit is called object.



3. Inheritance - Inheritance is a feature of Object Oriented programming. In inheritance you can create a new product by using existing product.

4. Polymorphism - The term Polymorphism means one thing many forms.

Any programming language which follows these four concept is known as object oriented programming language.

Class - Class is a blueprint of object. Class is collection of variables and methods. Class is created by using class keyword followed by classname, and the body of class is enclosed within braces.

Syntax -

```
class classname {
    // variables and methods
}
```

Constructor - Constructor is a special method which is used to initialize variable. Constructor has following property -

1. Constructor's name is same as class name.
2. Constructor does not return any value.

3. Constructor call automatically as soon as object is created.

Note - If you not create any constructor in class, then java compiler create a default constructor in java.

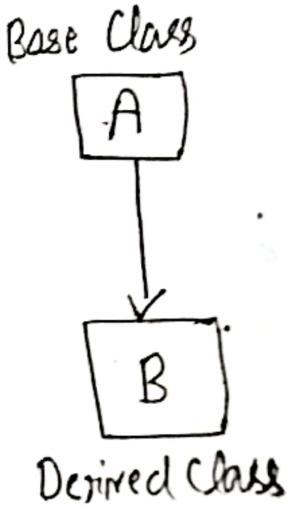
this - this is a keyword which denotes the object of current class.

Inheritance - Inheritance is a feature of OOP. In Inheritance you can create a new class by using existing class. The existing class is called base/super/parent class and new created class is called derived/sub/child class.

⇒ The concept of Inheritance is also called 'Reusability'.

Syntax -

```
class A {
// variable and methods
}
class B extends A {
// variable and methods
}
```

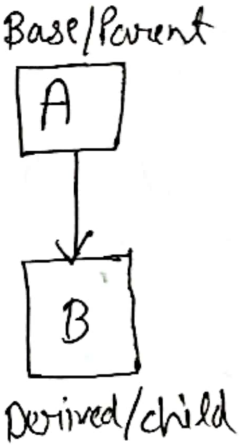


Types of Inheritance Supported in Java -

In Java Programming Language there are three types of Inheritance are supported -

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

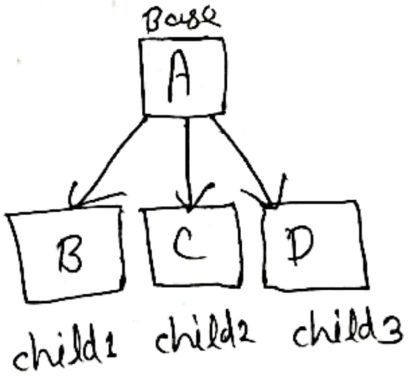
1. **Single Inheritance** - In Single Inheritance there is a base class and single derived class.



Syntax -

```
class A {
// code
}
class B extends A {
//code
}
```

2. **Hierarchical Inheritance** - In Hierarchical Inheritance there is a single base class and multiple derived classes.



Syntax -

```
class A {
//code
}
class B extends A {
// code
}
class C extends A {
//code }
```

3. Multi-Level Inheritance –

Syntax –

```
class A {
//code
}
class B extends A {
//code
}
class C extends B {
//code
}
```

Polymorphism – The term polymorphism means one thing many forms. In Java there are two types of polymorphism –

1. Compile Time Polymorphism (Method Overloading)
2. Run Time Polymorphism (Method Overriding)

1- Method Overloading – In java we can create multiple methods with same name in same class. But their parameters should be different. Based on method parameters it is decided at compilation time that which method call from where. This concept is called method overlodding.

Method Parameters can be different in two types –

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

1. eg. –

```
class Figure {
public int area(int s) {
return s*s;
}
public int area(int l, int b) {
return l*b;
}
}
```

2. Method Overriding – Re-writing of base class method into derived class is called Method Overriding.

eg=

```
class Connection {
public void connect() {
// code to connect oracle database
}
}
class NewConnection extends Connection {
public void connect() {
// code to connect MySQL database
}
}
```

Rules for Method Overriding -

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

Exception Handling in Java -

Exception - The term Exception means abnormal termination. When Exception is occured the program is terminated abnormally and rest of code is not executed. In Java Programming language there are three types of Exceptions -

1. Checked Exceptions
2. Unchecked Exceptions
3. Errors

1. Checked Exception - Checked Exceptions are those exceptions which are occured at compilation and intimated by compiler.

Eg:- InterruptedException, SQLException, ClassNotFoundException, FileNotFoundException, IOException etc.

2. Unchecked Exception - Unchecked Exceptions are those Exceptions which are occured at runtime. These exceptions are generated by either programmer mistake or user mistake.

E.g:- ArithmeticException, InputMismatchExcep., NullPointerExcep Exception, ArrayIndexOutOfBoundsException etc.

3. Errors - Errors are occured due to lack of system resources.

Eg:- IOError, AwtError, HeapMemoryFullError etc.

Exception Handling in Java

Exception Handling is a mechanism to handle exception to achieve normal execution of program. For exception handling you can use try, catch, finally, throw and throws keywords.

In Java we can handle exception in two ways -

1. By Using try - catch blocks.
2. By Using throws keyword.

Exception Handling by using try catch block -

```
try {
// code which you want to protect
}
catch( ExceptionType variable) {
// code which is used to handle Exception
}
finally // optional
{
// Code which you want to execute always
}
```

Note - For a single try block there can be multiple catch blocks.

Taking input from user by using object of BufferedReader class.

~~import~~

BufferedReader class is available in java.io package.

```
import java.io.*;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String name = br.readLine();
int a = Integer.parseInt(br.readLine());

float b = Float.parseFloat(br.readLine());

double c = Double.parseDouble(br.readLine());

long d = Long.parseLong(br.readLine());
```