# **Java Notes**

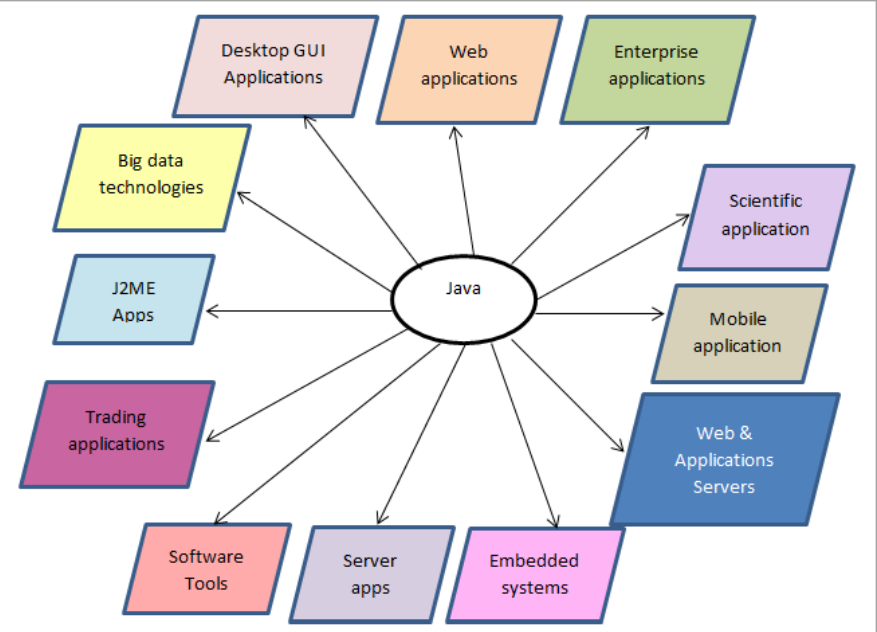
1. **Java Introduction** 
   1. **Java Introduction**

What is java?

* Java is a general purpose, high-level, object-oriented programming language.
* It was developed by James Gosling at Sun Microsystems in 1995.
* Later Oracle bought Sun MicroSystems.
* Java can be considered both a compiled and an interpreted language because its source code is first compiled into a binary bytecode.  This bytecode runs on the Java Virtual Machine (JVM), which is usually a software-based interpreter.

**Why Java is popular?**

**Applications of java**

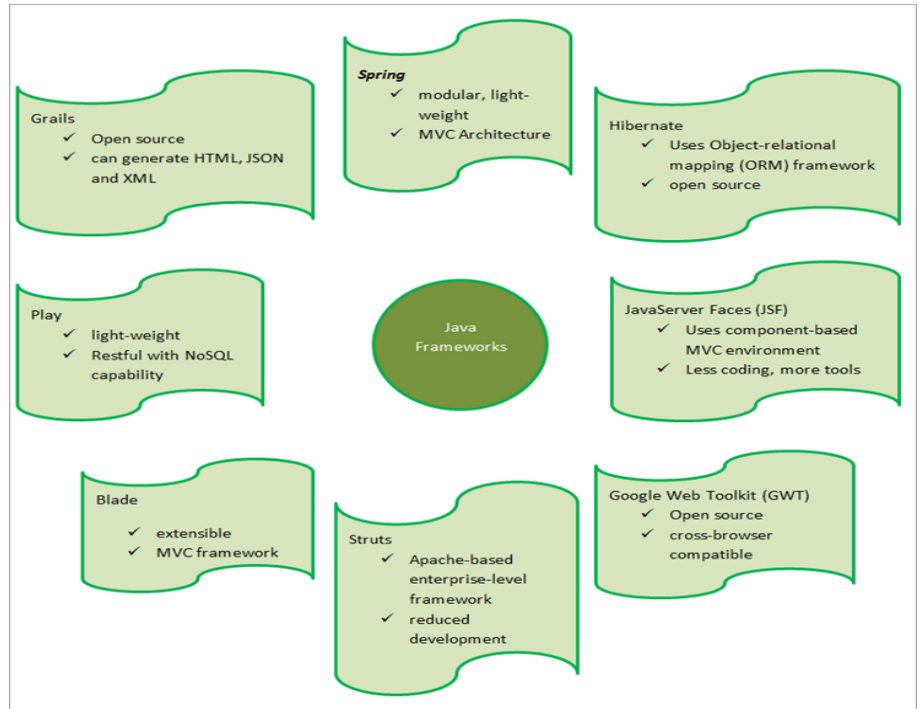


**Features of Java**

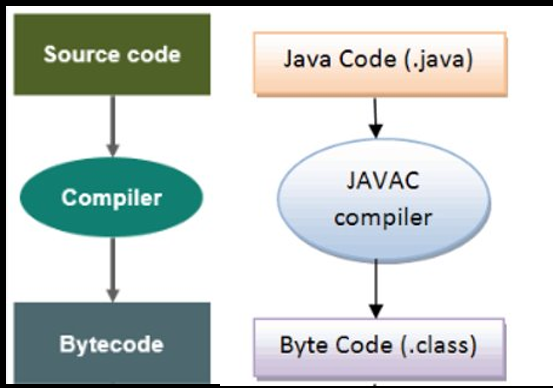
**Java is**

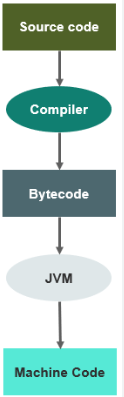
1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Multithreaded

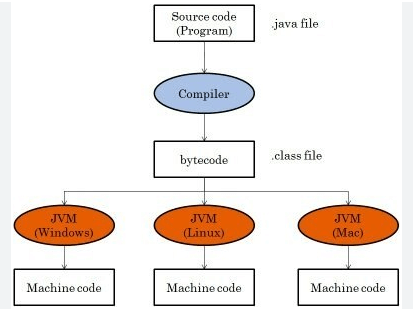
**Frameworks of java**

****

1. **Java Architecture**
   1. **Workflow of java**







A diagram of a computer code

Description automatically generated

**Working of JDK, JRE and JVM**

**At a high level, the Java architecture consists of four main components:**

1. **Java Development Kit (JDK)**
2. **Java Runtime Environment (JRE)**
3. **Java Virtual Machine (JVM)**
4. **Java Class Libraries**

JDK

**Java Development Kit (JDK):** The JDK is a software development kit that includes everything needed to develop and run Java applications, including the Java compiler, Java runtime environment, and a set of libraries and tools.

* **JDK = Java Development Kit**
* **JDK = JRE + Development tools**
* **The version of JDK is the same as version of Java**
* **JDK is a software development environment which is used to develop Java applications.**



JRE

**Java Runtime Environment (JRE):** The JRE is the subset of the JDK that is required to run Java applications. It includes the JVM and the class libraries needed for running Java programs. Please note that the JRE is enough to run any Java program if we are not interested in compilation of Java code. JRE doesn’t contain any development tools such as Java compiler, debugger. The Java architecture is designed to be highly modular and extensible, allowing developers to write custom libraries and tools that can be used alongside the standard Java libraries. This modular design also enables developers to build complex applications using a combination of pre-existing libraries and custom code, making Java a popular choice for enterprise-level software development.

* **JRE = Java Runtime Environment. It is also written as Java RTE**
* **JRE= JVM+ set of libraries + other files that JVM uses at runtime.**
* **The JRE is used for developing Java applications. It is used to provide the runtime environment.**



JVM

**Java Virtual Machine (JVM):** The JVM is an abstract machine that provides the runtime environment for Java programs. It is responsible for interpreting the compiled Java bytecode and executing it on the underlying hardware.

* **JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed.**
* **It is JVM which is responsible for converting Byte code to the machine specific code. It can also run those programs which are written in other languages and compiled to Java bytecode.**
* **JVMs are available for many hardware and software platforms.**

**The JVM performs the following main tasks:**

* **Loads code**
* **Verifies code**
* **Executes code**
* **Provides runtime environment**

**JVM contains**  
**1.Class loader**  
Class loader is used to load the byte code into the virtual machine.  
**2. Byte code verification**  
Byte code is verified that it is correct and according to norms.  
**3. Execution engine**  
Just in time (JIT) compiler is used by JVM to run the queue of runnable commands.

* **JRE and JDK are platform dependent. But JVM is platform independent. Hence Java is platform independent.**
* **There are three notions of JVM: specification, implementation and instance**

**Java Class Libraries:** The Java class libraries are a set of pre-written code modules that provide a wide range of functionality to Java programs, including input/output operations, networking, user interface development, and more.

1. **Downloading, installing and setting up path for JDK 17(LTS)** 
   1. **How to install java.**

**Download Java**

Step1: Click the below link or copy the link and paste it any browser, download java

<https://www.oracle.com/in/java/technologies/downloads/#jdk17-windows>

<https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe>

Step 2: Install java using jdk-17\_windows-x64\_bin.exe i.e open the exe file or double click it and proceed with next and close.

* 1. **How to setup environment variables.**

**Configure java path or Fixing of Environmental variable.**

Step 3: Copy java path → C:\Program Files\Java\jdk-17.0.2\bin

Step 4: Go to MyPC → Right Click → Properties → Advanced System Settings → Environment Variable → new 🡪 Enter variable name as javapath and enter variable value as C:\Program Files\Java\jdk-17.0.2\bin🡪 click ok

**To verify Java installation**

Step 5: Go to command Prompt → type java –version

 Output:

1. First Program
   1. Sample Java program in Note pad.

A diagram of a system

Description automatically generated

* 1. How to run java program through command prompt.

(Show how .class file will be created).

Syntax to compile Java program

>javac <program file name in .java extension>

Example:

>javac Sample.java

Note:

Once compiled Sample.class will be created.

Syntax to run or execute program

> java <program file name without extension>

Example:

>java Sample

Garbage Collection in Java

1. **Java Fundamental**

**Learning a Language**

**Steps in learning English language:**

**Steps in learning a Computer language:**

class Sample

{

public static void main(String args[])

{

System.out.println("welcome to java programming"); // output statemen

byte a=10, //declaration and initialization statement

int b=20; // declaration and initialization statement

int sum = a+b; // arithmetic statement

System.out.println("sum of a and b = "+sum); // output statement

}

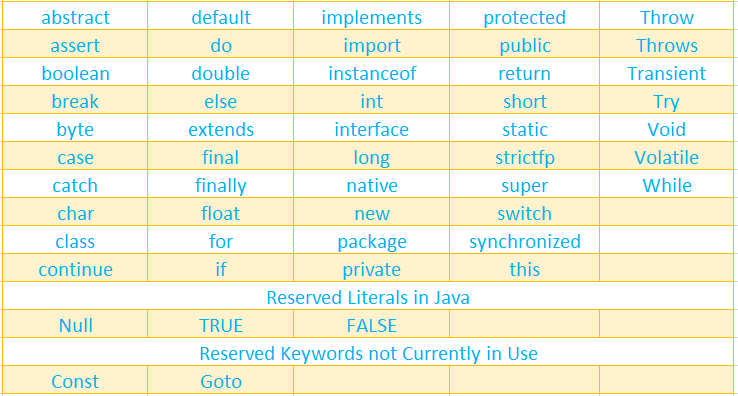
}

* 1. **Java Keywords**
* Set of words with some predefined meaning.
* Also called as Reserve words
* It’s having special meaning (predefined meaning) in each language.
* *Used to communicate with the compiler.*

Example

* printf - statement in c language. It is used to print the result in console i.e in output screen.
* break - it will break the current loop i.e execution will come out from the current loop.
* void means return null
* public means all can access

Example keywords in java



* 1. **Variables and Literals (Constants)**

**Variables**

* The values will be changed during execution

**Iiterals (Constants)**

* The value won't be changed during execution.

Example:

s=3,

b=2.5,

c='h'

name = "sam",

s, name, b, c--> variables

3, "sam", 2.5 ,'h'-->constants

* 1. **Java Identifiers**
* *Identifier is name, given to a variable, method, class, interface package etc.*
* *It can contain alphabets, digits and only 2 special symbols \_ (underscore), $(dollar).*
* *1st character should be an alphabet or \_ or $. But it cannot be a digit.*
* *You cannot use the keywords as identifiers*

Valid Identifiers

Name, x2, first\_name, $price, \_ Total 🡪 valid (correct) identifiers

Invalid Identifiers

First name 🡪 invalid identifier 🡪 space is not allowed

1customer 🡪 invalid identifier 🡪 first character should not be a number i.e digit

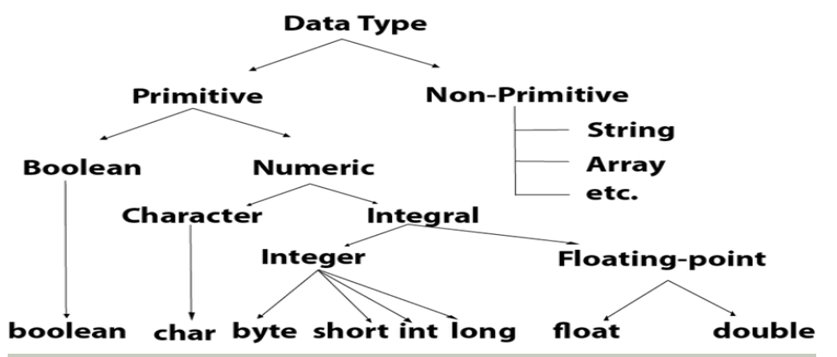
void🡪 invalid identifier🡪 keyword should not be a identifier

Give your answer in chat

Valid or invalid

1. Ab234 🡪
2. Last name
3. %mark
4. Total\_Amount
5. rateOfInterest
6. 12Customer

* 1. **Datatypes**
* A data type in Java represents the size and range of values that can be stored in a variable.
* **Java has two data types**

1. **Primitive – byte, short, int, long, float, double, char and boolean (true or false)​**
2. **Non-Primitive – Array,  String, Class, etc.​**
3. 

1. integer data type- byte, short, int, long

- only numbers without decimal point.

2. float, double

- number with decimal point

3. char

- Single character (may be alphabet or digits or symbols) expressed within single quotes

4. String – non-primitive data type

- combination or collection of characters expressed within double quotes

5. boolean

- true or false

**Memory size and Range of each data type with example**

|  |  |  |
| --- | --- | --- |
| **Data Type​**  **​** | **Size​**  **(byte)​**  **​** | **Range​**  **​** |
| byte​ | 1 byte​   (8 bits)​ | -128 to 127  ​  (Note: byte   a= 127;    is possible ​  byte a= 200; is not possible ​  short a=128;  is possible ​  int a=12345; is possible)​ |
| short​ | 2 bytes​  (16 bits)​ | -32768 to 32767​  short b=23500 🡪 possible  short s=33000 🡪 not possible |
| int​ | 4 bytes​  (32 bits)​ | -2147483648 to 2147483647(Note:  int mob = 7829266027 is not possible; Long mob = 78292660297 is possible)​ |
| long​ | 8 bytes​  (64 bits)​ | -9223372036854775808 to 9,223,372,036,854,775,807​ |
| float​ | 4 bytes​ | -3.4e38 to 3.4e38​ |
| double​ | 8 bytes​ | -1.7e308 to 1.7e308​ |
| char​ | 2 bytes​ | 0 to 65535​ |
| boolean​ | 1 bit​ | True (1) or false (0)  boolean check;​                                       check = 10 > 15;​ |

In Java, the key differences between primitive and non-primitive data types are:

1. Definition:
   * Primitive Data Types: Basic data types predefined by Java, used to store simple values.
   * Non-Primitive Data Types: Also known as reference types, these are objects created by programmers and include classes, arrays, and interfaces.
2. Memory Size:
   * Primitive Data Types: Fixed size (e.g., int is 4 bytes).
   * Non-Primitive Data Types: Size varies depending on the object and its properties.
3. Storage:
   * Primitive Data Types: Store actual values directly in the memory.
   * Non-Primitive Data Types: Store references (memory addresses) to the objects.
4. Default Values:
   * Primitive Data Types: Have default values (e.g., int is 0, boolean is false).
   * Non-Primitive Data Types: Default to null.
5. Operations:
   * Primitive Data Types: Direct operations are performed.
   * Non-Primitive Data Types: Operations are performed through methods (e.g., .length() for arrays).
6. Examples:
   * Primitive Data Types: int, char, float, boolean, etc.
   * Non-Primitive Data Types: String, Arrays, Classes, etc**.**
7. **Reading User Input**
   1. **- How to read user input using scanner class.**

* *The Scanner class is used to get user input*
* *It is found in the java.util package.*

*Built- in- package 🡪 util*

*public class Scanner*

*{*

*Int nextInt()*

*{…*

*….. return a;}*

*float nextFloat()*

*{……..}*

*…………*

*……*

*}*

*import java.util.Scanner;*

input methods

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

**Example program:**

Import java.io.\*;

import java.util.Scanner;

class Myclass

{

public static void main(string[] args)

{

Scanner obj = new Scanner(System.In);

    System.out.println(“Enter employee number, name and salary");

int empno = obj.nextInt();              // integer number input

  float emp\_score = obj.nextFloat();              // Float number input

double salary = obj.nextDouble();         // double number input

string name = obj.nextLine(); // string input

}

}

* 1. How to read user input using buffer reader class

User Input – using BUFFERED READER Class

* *Buffered Reader class is like Scanner class. It is used to read the sequence of characters.*
* *This class provides the following methods:*

1. *read () It is used to read a character.*
2. *readLine() It is used to read a line of characters. And we also need to use InputStreamReader.*