JAVA

Topics covered:   
1. Introduction  
2. Installing of JAVA  
3. Java Architecture  
4. Java Programming Format  
5. Java Language Fundamentals  
1. Tokens  
1.1 Identifiers  
1.2. Literals  
1.3. Keywords  
1.4. Operators  
2.Data types  
3.Typecasting  
4. Java statement  
5.Arrays  
6. Variables  
7. Control statements   
7.1Select statements:  
if  
Nested ifs  
if else if ladder  
switch  
7.2Jump statements:  
break  
continue  
return  
7.3iterator statements:  
for  
while  
dowhile  
8.Methods  
9.OOPS   
Classes and objects  
super keyword  
Object class  
10. Constructor

11.Inheritance  
12.Abstraction   
Abstract class  
Java types  
13.Interfaces  
14.Encapsulation  
15.IIB  
16.SIB  
17. Exception handling  
18. Wrapper classes  
19. Collections  
List  
Queue  
Set  
Map

What is programming?  
- Programming is a way to “instruct the computer to perform various tasks”.

What is JAVA?  
\*JAVA is a high-level programming language. (Human understandable form - English)

\*JAVA is a object oriented programming language designed to develop all kinds of applications.

Programming paradigms available in market?  


Who invented java ?  
James Gosling – June 1991 - OAK  
Sun micro systems – 1996 – first version of java

Oracle – 2001 – 2025 (java is maintained)

What are the advantages of java?

1. Simple
2. Object oriented
3. Platform independent
4. Architectural neutral
5. Portable
6. Robust
7. Secure
8. Dynamic
9. Multi-threaded
10. Distributed
11. High performance
12. Interpretive
13. Simple

* Less memory and less execution time
* Removed all the confusion features like pointers, multiple inheritance, operator overloading
* Simplified syntax from c and c++

1. Object oriented

* Able to store data in the form of objects.

1. Platform independent

* Compile on one OS and to execute on another OS.

1. Architectural neutral

* One H/W arch and to execute on another H/W arch.

1. Portable

* Able to execute applications on all the os and all the H/W systems.

1. Robust

* Very good at Memory management system and having exception handling mechanisms
* Heap memory management system – allocates and deallocates memory of the objects at the run time.
* Very good predefined library

1. Secure

* JAAS [Java authentication and authorization service]
* Security manager inside the JVM – java virtual machine – implicit security
* JAAS to provide web security
* Network security algorithms

1. Dynamic

* Allow memory allocation for primitive data types at run time, not at compilation time.

1. Multi-threading
2. Distributed

* Standalone application
* Distributed application – web services, EJB’s, socket programming etc…

1. High performance
2. Interpretive

* Compilative and Interpretive

What are the dis advantages of java?

UI components

What is the full form of Java ?

* No full form
* Coffee symbol – coffee bean name

Diff technologies available in java ?

1. Java se – standard edition – desktop or standard applications
2. Java ee – enterprise edition – Websites for banking, health care etc..
3. Java me – micro edition – coffee machine, atm machine …

What is **Java Architecture** ?  
Principle – WORA (Write once and run anywhere)





JDK – Java development kit

JRE – Java Run time environment

JVM – Java virtual machine

JDK = JRE + Development tools

JRE = JVM + Library classes  
  
JDK – Java development kit – write, compile, debug, and run java prog.

JRE – Minimum run env +   
Development tools :   
javac – java compiler – java source code(.java) -> Java byte code(.class)  
java – java application launcher  
jar – tool for creating and managing java archive file which helps in distribution of code.  
Javadoc – a documentation generator.  
  
**JVM – Platform independent**



* JVM is the Interpreter byte code – machine code (instructions)
* Platform independent
* Loads the code
* Verifies the code
* Executes the code
* Provides run time environment

\*\*\*  
JVM performs following key steps :

1. Class Loader : loads the .class files into the memory  
   This involves 3 phases:

1.1 Loading : Locates and loads the binary data of a class.  
1.2 Linking : following 3 stages   
 1.2.1 Verification : structural correctness and security violations

1.2.2 Preparation : memory for static variables and initialize them with default values  
 1.2.3 Resolution : replaces symbolic references in the byte code with direct references.  
 1.3 Initialization : executes static initializers and static blocks in the class

2. Java Run time date area / JVM memory area   
 The JVM manages various memory areas during execution :  
 2.1 Method area : Stores class – local data , including meta data, static variables, and some method code.  
 2.2 Heap area : The run time data area where objects and arrays are allocated. This is the place where garbage collection primarily operates.   
 2.3 Stack area : Each thread in the JVM has its own private stack. Stack frames are created for each method call, storing local variable’s, operand stack and method return values.  
 2.4 PC Registers : Each thread has a PC (program counter) register that stores the address of the currently executing JVM instruction.  
 2.5 Native method stacks : Used to support native methods

3. Execution engine : Executing the byte code

3.1 Interpreter : Interprets byte code instructions line by line and executes them. This will be usually slow.  
 3.2 JIT compiler: Just in time – improve performance, jit compiler compiles frequently executed byte code. Executing the byte code much faster.   
 3.3 Garbage collector : Automatically reclaims memory occupied by the objects that are no longer referenced / unused.   
  
4. Java Native interface (JNI) : java code to interact with native applications and library written in other prog languages.

Why we need to install Java software into our system ?  
How do we download java software ?  
https://www.oracle.com/in/java/technologies/downloads/  
Which version of java software need to be installed ?  
 *Long-Term Support (LTS)*  - JDK21  
Short-Term support(STS)  
<https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe>   
[jdk-21\_windows-x64\_bin.exe](https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe)   
How do we set the java path in our system ?  
1. Go to this PC > Right click > Properties or  
Go to Control Panel\System and Security\System  
2. Click on Advanced system settings   
3. Click on Environment variables  
4. Click on New   
Variable name : JAVA\_HOME  
Variable value : C:\Program Files\Java\jdk-22  
5. Go to Path variable   
Click on New   
%JAVA\_HOME%\bin   
How do we cross check which version of java installed in out laptops ?  
Go to the cmd prompt and type java --version  
  
Where we need to write the java program ?  
Note pad or Note pad ++

Tools available for writing java programs ?  
1. Eclipse  
2. Intellij IDEA  
3. Netbeans  
4. Visual studio code  
  
Open the note pad and type   
  
class Test{

public static void main(String[] args)

{

System.out.println("Welcome to nammaqa bng");

}

}  
  
Save – “Test.java”  
open the cmd prmt > javac Test.java > java Test  
  
Download and install eclipse:  
https://www.eclipse.org/downloads/   
  
Steps to create java project   
\* Select the default work space and start the eclipse   
\* For creating the java project you can click on create a java project directly or click on File > New > Java project  
\*Provide “JavaPractice” in project name.  
\*Click on Next   
\* Click on Finish  
\*create a package by right click on src folder and > New > Package > “com.day1” – package name > save  
\*Create a class by right click on package name > New > Class > “Prog1” – class name > select public static void main > Finish  
\* Right click on Prog1.java > Run As > Java application > To run the java files and see the o/p in the console.  
  
Package section :  
Package is collection of related classes and interfaces as a single unit.  
Following advantages :  
1. Modularity

2.Abstraction  
3. Security  
4.Sharability  
5.Reusability  
There are 2types of packages in java :  
1. Pre defined packages -   
Eg : java.io, java.util, java.awt, java.sql….  
2. User defined packages  
Eg: org.app.login, org.app.signup  
syntax : package package\_name;  
2 conditions for packages :  
1. Package declaration statement must be the first statement in java .java files.  
Q: is it possible to provide more then one package declaration with the same name ?  
No  
Q: Can I place 2 package names in my java files ?  
No  
2.Package names must be unique, they must not be sharable and they must not be duplicated.  
  
import section :  
\* is to make available all the classes and interfaces of a particular package into present java file.  
syntax : import package\_name.\*;  
Eg :   
**import** java.io.\*;  
**import** java.lang.\*;  
**import** day2.\*;  
\* **import** java.io.\*; -> Able to import all the classes and interfaces from java.io package  
\* **import** java.io.BufferedInputStream; -> Able to import only the specified member from the specified package.  
  
Note : 1 package statement but we can have multiple import statements.  
  
Q: To use the classes and interfaces of a particular package in java files is it mandatory to import that package ?  
No  
  
Eg : With import statement :   
**import** java.io.\*;  
BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));

System.***out***.println("Please enter your name");

System.***out***.println("Your name is "+br.readLine());

Without import statement :   
java.io.BufferedReader br = **new** java.io.BufferedReader(**new** java.io.InputStreamReader(System.***in***));

System.***out***.println("Please enter your name");

System.***out***.println("Your name is "+br.readLine());

Java Programming Format :  
To design basic java application within single java file we have follow following structure:



Comment section :  
3 types :  
1. Single line comment :   
Syntax : //---------Description-------------  
2. Multi line comment :   
Syntax :   
/\*-----------  
--------------  
---Description--  
-------------  
--------------\*/  
3.Documentation comment:  
Syntax  
/\*  
\*------------------------  
\*------------------------  
\*--Description--------  
\*-----------------------  
\*/  
  
Class section :   
class - Blue print  
object – real world entity  
Syntax :  
**public** **class** className {

}  
  
Q: What are the diff members of a class ?  
Variables and methods  
  
Main class section :  
contains main() method  
1. To define application logic in java application.  
2. To define starting and ending point for the application execution.  
Syntax  
**public** **static** **void** main(String[] args) {

}  
  
Execution of java program will follows following steps :  
\* On right click run as java application : jdk will helps in converting .java to .class file.  
\*jvm will search for .class file and look for main() method present or not  
\***If the required .class file is not available at the specified location, then jvm will throw :  
java.lang.NoclassDefFoundError.   
  
java.lang.ClassNotFoundException -> after compiling the code we have passed the wrong class name to execute the prog.**  
\*When jvm identifies main .class file at the specified location then it will load byte code into the memory, this phase is called as “class loading”, this would be performed by “class loader” component existed inside JVM.

\*After loading main class byte code to the memory, jvm will search for main() method, if main() method does not exists then jvm will throw :  
Main method not found in class Test, please define the main method as:

public static void main(String[] args)  
\*If main() method is identified in main class bytecode then jvm will create a thread to access main() method called as “Main thread”.

\*When main thread reached to main() method ending point then main thread will get dead state, with this jvm will stop all of its internal process and jvm will go to shut down mode.  
  


Language Fundamentals:   
1. Tokens  
2.Data types  
3.Typecasting  
4.Java statement  
5.Arrays  
  
1.Tokens  
Smallest logical units – “Lexeme”.  
  
Eg: keyword – Token

|  |  |  |  |
| --- | --- | --- | --- |
| int | for | break | continue |
| lexeme | lexeme | lexeme | lexeme |

Operator – Token

|  |  |  |  |
| --- | --- | --- | --- |
| + | - | \* | / |
| lexeme | lexeme | lexeme | lexeme |

Types of tokens :  
1. Identifiers  
2. Literals  
3. Keywords  
4. Operators  
  
1. Identifiers   
name assigned – variable, methods, classes etc…  
Rules   
1. Must not start with a number  
2.May start with an alphabet, \_ symbol, $symbol, but subsequent symbol must start with an alphabet, a number, \_ symbol and $ symbol.  
3. Not allowing spaces in the middle.  
4. Should not be duplicated  
5. All predefined class names  
  
  
**int** empNo = 10;  
  
int = lexeme  
empNo = Identifier  
= = operator  
; = special symbol/terminator  
  
  
Suggestions :   
1. Must be meaningfull  
  
 String xxx = "abc123"; //not suggestible

String accNo = "123"; //Suggestible

1. There is no length restriction  
     
   String Empxxx\_temp\_adress\_xxxxxxxxxxxxxxxxxx = "abc123"; //not suggestible

String EmpTempAddress = "123"; //Suggestible

1. Use (\_)  
   String EmpTempAddress = "123"; //not Suggestible

String Emp\_Tem\_Address = "123"; //Suggestible

2. Literals  
Literal is a constant assigned in the variables

**int** i = 10;

System.***out***.println(i);

//int - lexeme, keyword, data type

//i - identifier

//= - operator

//10 - constant / literal

//; - terminator

1. Integral/integer literals
2. Floating point literals
3. Boolean literals
4. String literals

3.Keywords/Reserved words

Eg: goto, const..  
  
List of keywords:  
1. Data type and return types :  
byte, short, int, long, float, double, char, Boolean, void …  
2. Access modifiers :  
public, protected, private, static, final, abstract, native, synchronized, strictif ..  
3. Flow controllers :  
if, else, switch , case, default for, while, do, break, continue, return ….  
4. Class/object related :  
class, extends, interface, implements, enum, new, this, super, package, import …  
5. Exception handling:  
throw, throws, try, catch, finally…



4. Operators  
4.1 Arithmetic operators: +,-,/,\*,%,++,--  
4.2. Assignment operators: =, +=,-=,/=,%=  
4.3. Comparison operators: ==, != <, >, >=,<=  
4.4. Boolean logical operators: &, |, ^  
4.5. Bitwise logical operators: <<,>>  
4.6. short circuit operators: &&, ||



And - &

|  |  |  |
| --- | --- | --- |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

OR - |

|  |  |  |
| --- | --- | --- |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |