Java Notes - TheTestingAcademy (Pramod Sir)

✅ Why Learn Java?

- Java is a general-purpose, object-oriented programming language that was designed by James Gosling at Sun Microsystems in 1995
- Compilation of the Java applications results in the bytecode that can be run on any platform using the Java Virtual Machine. Because of this, Java is also known as a WORA (Write Once, Run Anywhere)
- Java Version - https://en.wikipedia.org/wiki/Java_version_history
- High Level, Platform independent & Architecture Neutral
- Secure, Robus, Multi Thread, Distributed and OOPs language.

⚠️ Why Java is not 100% Object-Oriented?

- Primitive data types.
- Use of Static.
- Wrapper class

⚠️ How is Java Secure?

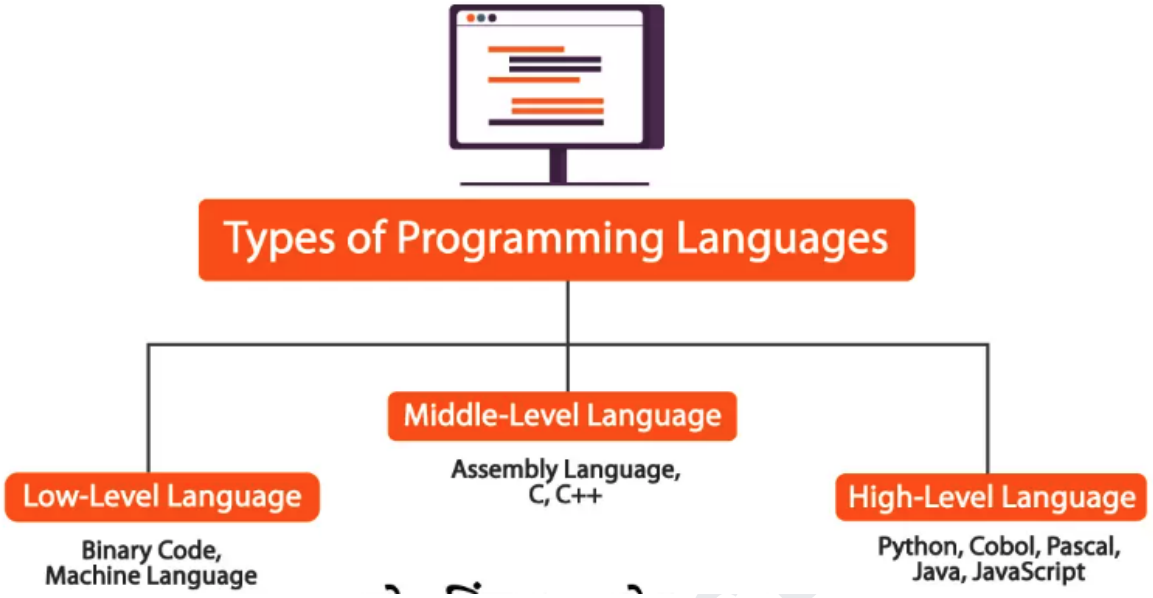
- JVM which protects from unauthorized or illegal access to system resources.
- Oops and inner classes

Ref - https://www.scaler.com/topics/why-java-is-not-100-object-oriented/

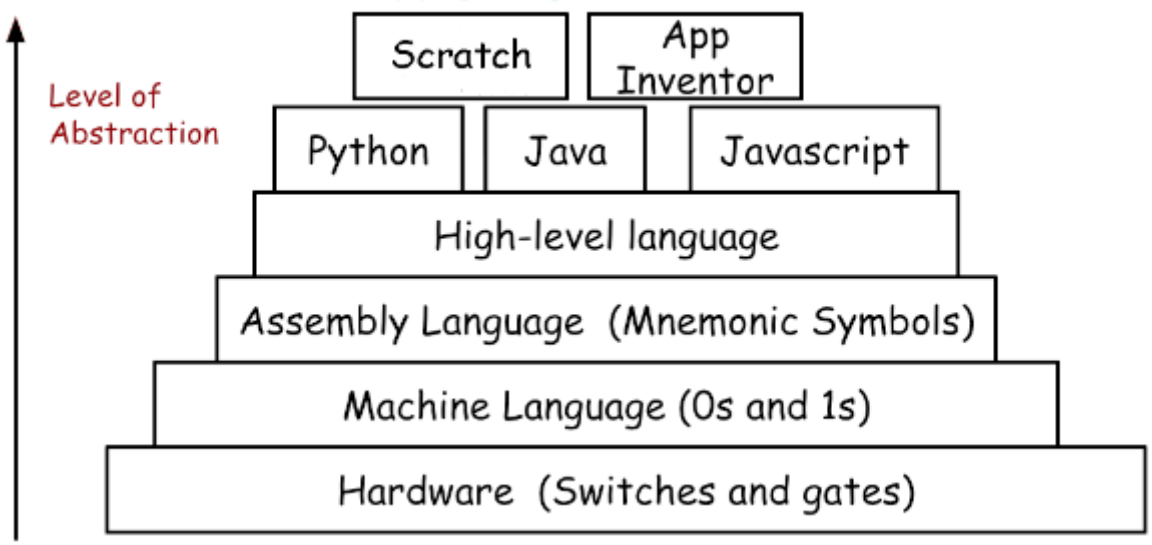
Important Points

- Since Java applications can run on any kind of CPU so it is architecture – neutral.
- Java program can be executed on any kind of machine containing any CPU or any
- operating system.

- Java is robust because of following: Strong memory management(Garbage Collector) No Pointers Exception handling Type checking mechanism Platform Independent.
- Using RMI and EJB we can create distributed applications in java



Check the Languages, with the Level of Abstraction



Major Features of Java Programming Language

- Simple.
- Object-Oriented.
- Platform Independent.
- Portable.
- Robust.
- Secure.
- Interpreted.
- Multi-Threaded

⚠️ **What is Source Code?**

Human understandable code written using High Level Programming language is called as Source Code. (NameOfFile.java)

⚠️ **What is Byte Code?**

JVM understandable code generated by Java Compiler is called as Byte Code. Byte code is also called as Magic Value.

⚠️ **Why C and C++ are Platform Dependent?**

When you compile C or C++ program on one Operating System then compiler generates that Operating System understandable native code.Native code generated on one OS will not run on other OS directly. Window -> exe, Mac -> dmg, Linux, deb, pkg

⚠️ **What is Java Compiler?**

Java Compiler is a program developed in C or C++ programming language with the name "javac". It will check syntactical or grammatical errors of the programs. It converts source code to byte code.

⚠️ **What is Java Interpreter?**

Java Interpreter is a program developed in C or C++ programming language with the name "Java". It will convert byte code to native code line by line. It will execute that native code.

⚠️ **What is JIT Compiler?**

JIT (Just-In-Time) compiler is a component of the Java Runtime Environment. JIT Compiler compiles or translates or converts the neccessary part of the bytecode into machine code instead of converting line by line. Because of this, performance of Java program has improved.

Run your First Java Program

1. Install Java and Set path to the Home in Windows / Mac
2. Install IDE (IntelliJ) and you can avoid the first step.
3. Create new command line project and Add the code and run the program.

How to Set JAVA_HOME path in MAC - Click here

1. Download the JDK latest -

```
echo export "JAVA_HOME=\$(/usr/libexec/java_home)" >> ~/.bash_profile
```

2. If you're using zsh (which probably means you're running macOS Catalina or newer), then it should instead be:

```
echo export "JAVA_HOME=\$(/usr/libexec/java_home)" >> ~/.zshrc
```

3. In either case, restart your shell.

How to Set JAVA_HOME path in Widnows

Set the JAVA_HOME Variable

To set the JRE_HOME or JAVA_HOME variable:

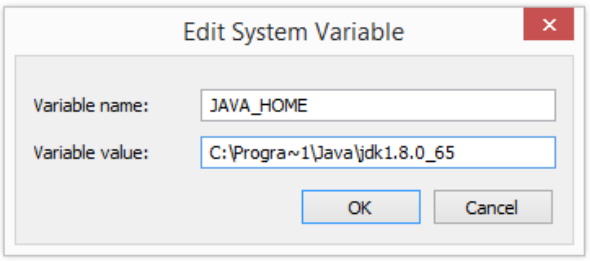
1. Locate your Java installation directory
   If you didn't change the path during installation, it'll be something like

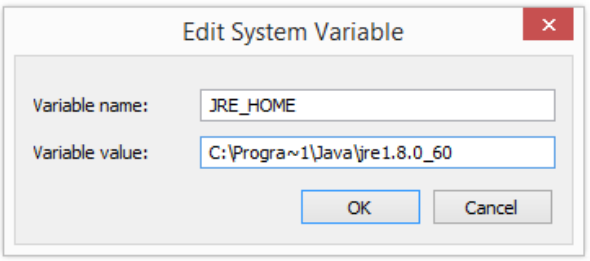
```
C:\Program Files\Java\jdk1.8.0_65
```

You can also type `where java` at the command prompt.

2. Do one of the following:
   Windows 7 – Right click **My Computer** and select **Properties** > **Advanced**
   Windows 8 – Go to **Control Panel** > **System** > **Advanced System Settings**
   Windows 10 – Search for **Environment Variables** then select **Edit the system environment variables**
3. Click the **Environment Variables** button.
4. Under **System Variables**, click **New**.
5. In the **Variable Name** field, enter either:
   - JAVA_HOME if you installed the JDK (Java Development Kit)
     or
   - JRE_HOME if you installed the JRE (Java Runtime Environment)

6. In the **Variable Value** field, enter your JDK or JRE installation path .
   If the path contains spaces, use the shortened path name. For example,
   `C:\Progra~1\Java\jdk1.8.0_65`





   Note for Windows users on 64-bit systems
   Progra~1 = 'Program Files'
   Progra~2 = 'Program Files(x86)'
7. Click **OK** and **Apply Changes** as prompted

⚠ What is JDK (Java Development Kit) / SDK (Software Development Kit)?

It is a set of various utility programs which are required for developing and executing the java programs.
It is Platform dependent. Various JDKs are provided for various Operating Systems.
Following are various utility programs provided under JDK:
1) Java Development Tools
   i. javac
   ii. java
   iii. javap
   iv. Jar
etc
2) Source Files
3) JRE
etc

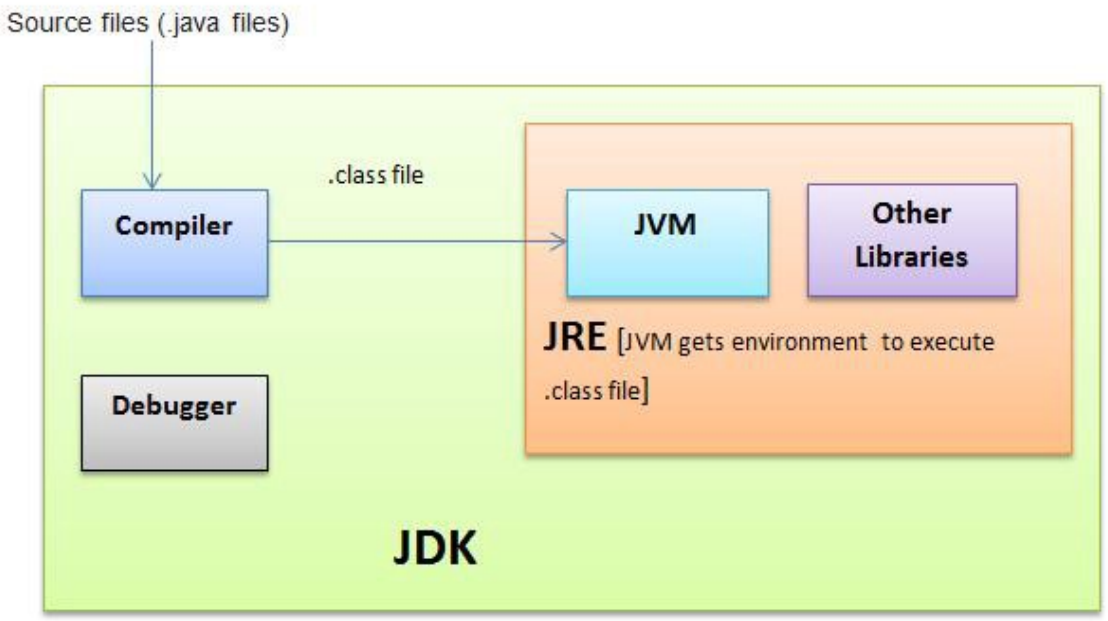
⚠ What is JRE?

Ans: JRE stands for Java Runtime Environment. It is an implementation of JVM. It contains class
libraries, Interpreter, JIT Compiler etc. Only JRE is enough to run the Java program

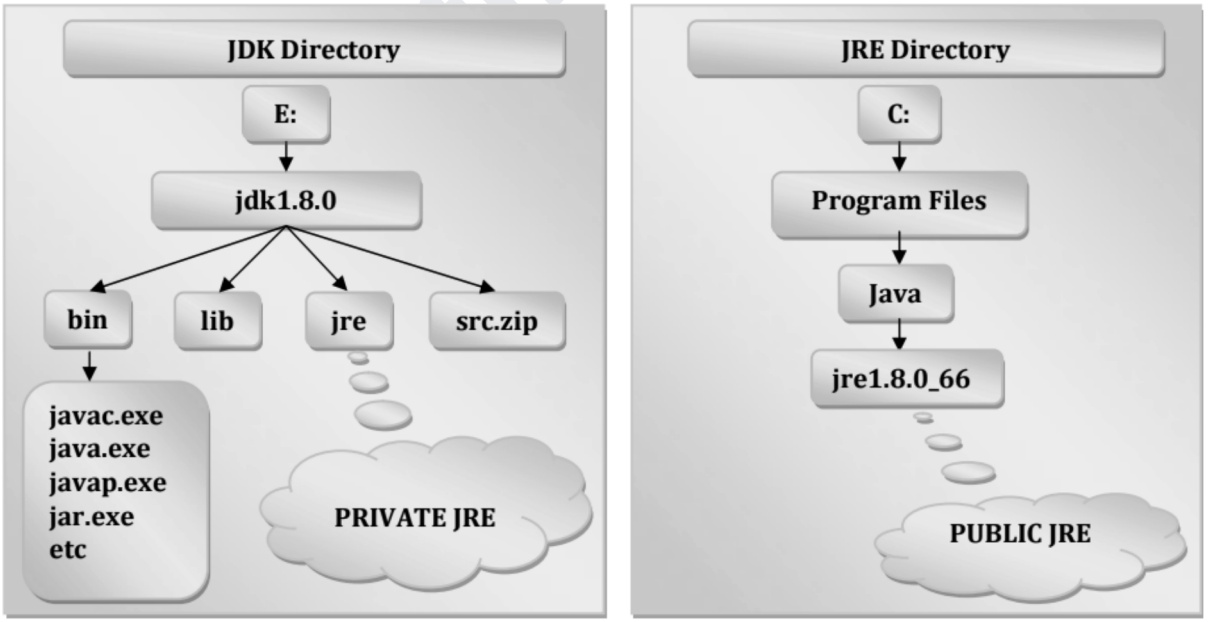
⚠ What is JVM?

Ans: JVM stands for Java Virtual Machine. It is a specification provided by SUN Microsystem whose
implementation provides an environment to run our Java applications. JVM becomes an instance of
JRE at run time.

Sun’s implementations of the Java Virtual Machine (JVM) is itself called as JRE. Sun’s JRE is availabe
as a part of JDK and also as a separate application.
Many vendors has implemented JVM. Some of them are SUN JRE, IBM JRE, Oracle JRE etc



JDK Vs JRE



Java Editions

1. Enterprise Edition - Servlets, JSP, JDBC etc.
2. Micro Edition - micro devices like Mobiles,Setup Box etc
3. Standard Edition - Used to develop standalone applications using applet and swing.

Edition	Description
Java SE (Standard Edition)	**The main edition of Java that is used for desktop and server applications. It includes the Java language, JVM, class libraries, and tools for developing, testing, and deploying Java applications.**
Java EE (Enterprise Edition)	**A set of specifications and APIs for developing enterprise applications, including web and mobile applications, distributed systems, and cloud computing. It includes components like servlets, JSPs, EJBs, JMS, JPA, and more.**
Java ME (Micro Edition)	**A platform for developing applications for mobile and embedded devices, like phones, PDAs, and set-top boxes. It includes a subset of the Java SE APIs and tools, as well as additional libraries for mobile-specific features like user interfaces and networking.**
Java FX	**A platform for developing rich client applications and user interfaces, with features like graphics and media support, animation, and web integration. It includes a set of APIs and tools for developing desktop, mobile, and web applications.**

Note that Java EE has been rebranded as Jakarta EE, as Oracle transferred ownership of the Java EE specification to the Eclipse Foundation.

Java Versions

Java Version	Key Features
Java 1.0	Garbage collection, multi-threading, network programming
Java 1.1	Inner classes, JavaBeans, JDBC
Java 1.2 (Java 2)	Collections framework, reflection, Swing GUI toolkit, JNDI
Java 1.3 (Java 2)	HotSpot JVM, Java Sound API, Bluetooth support
Java 1.4 (Java 2)	Assertions, regular expressions, Java Web Start, JNI
Java 5 (Java 1.5)	Generics, annotations, enhanced for loop, concurrent API
Java 6 (Java 1.6)	JAXB API, Java Compiler API
Java 7 (Java 1.7)	Try-with-resources, multi-catch exceptions, Fork/Join Framework, NIO.2
Java 8 (1.8)	Lambda expressions, Stream API, java.time package, CompletableFuture
Java 9	Module system, private interface methods, JShell
Java 10	Local variable type inference, garbage collector improvements
Java 11	HttpClient API, var keyword for lambda parameters, Unicode 10.0 support

Java 12	Switch expressions, garbage collector improvements
Java 13	Text blocks, Z Garbage Collector improvements
Java 14	Record classes
Java 15	Sealed classes and interfaces, garbage collector improvements
Java 16	Pattern matching for instanceof, garbage collector and JFR improvements
Java 17	Sealed Types Preview, garbage collector improvements

Java Comments

Comment Type	Syntax	Description
Single-line comment	// comment	A comment that extends from the // marker to the end of the line. It is ignored by the compiler and used for adding notes and explanations to the code.
Multi-line comment	/* comment */	A comment that can span multiple lines, starting with the /* marker and ending with the */ marker. It is also ignored by the compiler and used for adding longer explanations or temporarily disabling code.

Documentation comment	/** comment */	A comment that begins with /** and ends with */. It is similar to a multi-line comment, but it is specifically used for generating API documentation with tools like Javadoc. It can include tags like @param, @return, @throws, etc.
Javadoc comment	/**	A special type of documentation comment that starts with /** and includes additional Javadoc tags for generating documentation. It can be used to describe classes, interfaces, methods, fields, etc. and their functionality. It is a best practice to include Javadoc comments in your code for documentation purposes.

⚠️ **Why this Kolavari DI? (Reason for this error)**
Install JDK properly

```
'javac' is not recognized as an internal or external command,
operable program or batch file.
```

⚠️ **Difference between Java and C++**

Java	C++
1. Java doesn't support Pointers.	1. C++ supports Pointer.
2. Java does not support multiple inheritance with classes.	2. C++ supports multiple inheritance with classes.
3. Java doesn't support Global Variables (Variables declared outside the class).	3. C++ supports Global Variables.
4. Java doesn't support header files.	4. C++ supports header files.
5. const and goto keywords can't be used in Java.	5. const and goto keywords can be used in C++.
6. Java doesn't support Destructors.	6. C++ supports Destructors.
7. Java doesn't support Virtual Functions.	7. C++ supports Virtual Functions.
8. Java doesn't support Operator Overloading.	8. C++ supports Operator Overloading.
9. Java doesn't support Scope Resolution Operator.	9. C++ supports Scope Resolution Operator (::).
10. Java has Built-in API for Multithreading and Network Programming.	10. C++ doesn't have Built-in API for Multithreading and Network Programming.
11. Java has a Garbage Collector to clean the memory automatically.	11. No automatic memory cleaning process in C++.

⚠️ **Differences between Oracle JDK and OpenJDK**

Both OpenJDK and Oracle JDK are created and maintained currently by Oracle only.

Most of the vendors of JDK are written on top of OpenJDK by doing a few tweaks to [mostly to replace licensed proprietary parts / replace with more high-performance items that only work on specific OS] components without breaking the TCK compatibility.

https://stackoverflow.com/questions/22358071/differences-between-oracle-jdk-and-openjdk

⚠️ **What is the use of setting the PATH?**
Setting the PATH is important for the operating system to know where to look for executable files when a command is executed. In the case of Java, setting the PATH is essential for the system to locate the Java compiler and runtime environment (JRE) executable files.

⚠️ **What is the reason for the following error?**

- Javac : file not found Hello.java
- Error Could not find or load main class Hello
- java.lang.NoClassDefFoundError : hello(Wrong name Hello)
- java.lang.ClassFormatERROR : Incompatible magic value (change in .class)
- java.lang.UnSupporttedClassVersion : Hello (Compiled with Old)

⚠️ **What are the ways available to set the PATH?**
There are several ways to set the PATH, including using the command line, editing system environment variables, and using third-party tools or scripts. The specific method used may vary depending on the operating system being used.

⚠️ **What is the difference between PRIVATE JRE and PUBLIC JRE?**
A PRIVATE JRE is a Java runtime environment that is installed and used by a specific application or user, and is not shared with other applications or users on the same system. A PUBLIC JRE is a Java runtime environment that is installed and made available to all applications and users on the system.

⚠️ **Which JRE will be used while compiling the Java Source File?**
The JRE is not used for compiling Java source files, but rather the Java Development Kit (JDK) is used, specifically the Java compiler (javac) which is included in the JDK.

⚠️ **What will happen when I am compiling Java program without PRIVATE JRE?**
Compiling a Java program does not require a PRIVATE JRE. However, if the program requires a specific version of the JRE to execute, and that version is not installed on the system, the program may not run properly.

⚠️ **Which JRE will be used while executing Java application?**

The JRE that is used to execute a Java application depends on several factors, including the system's PATH settings, the application's CLASSPATH settings, and any environment variables that may be set.

⚠️ **What will happen when I am executing Java program without PUBLIC JRE?**
If a PUBLIC JRE is not installed on the system, the Java application will not be able to run. The user will need to install a JRE before the application can be executed.

⚠️ **Can we execute java program without setting PATH?**
It is possible to execute a Java program without setting the PATH, but it requires specifying the full path to the Java executable file every time the command is executed.

⚠️ **What is the difference between JVM and JRE?**
JVM (Java Virtual Machine) is a virtual machine that runs Java bytecode, while JRE (Java Runtime Environment) is a software package that includes the JVM as well as other libraries and components required to run Java applications.

⚠️ **What is the difference between JIT Compiler and Java Interpreter?**
A JIT (Just-In-Time) compiler is a component of the JVM that dynamically compiles bytecode into machine code at runtime, while a Java Interpreter executes bytecode directly without compiling it.

⚠️ **What is the difference between Byte code and Native code?**
Bytecode is a machine-independent code that is generated by the Java compiler and is designed to be executed by the JVM. Native code, on the other hand, is machine-specific code that is directly executable by the CPU.

⚠️ **Is it possible to execute Java program without having JDK in machine?**
It is possible to execute a Java program without having the JDK (Java Development Kit) installed on the machine, but a JRE (Java Runtime Environment) must be installed.

⚠️ **Can I install multiple JDK versions in my machine?**
Yes, it is possible to install multiple JDK (Java Development Kit) versions on the same machine. It is important to manage the PATH and other environment variables correctly to ensure that the correct version is used when compiling or executing Java programs.

⚠️ **Can I install multiple JRE versions in my machine?**
Yes, it is possible to install multiple JRE (Java Runtime Environment) versions on the same machine. It is important to manage the PATH and other environment variables correctly to ensure that the correct version is used

---

Explain the main method in Java

- The main method in Java is the entry point for any Java program.

- It is a predefined method that is executed when a Java program is run, and is required in every Java program.
- The main method has a specific signature, which includes the keyword "public", the keyword "static", the return type "void", and a parameter of type "String" array named "args". Here is an example of the main method signature:
- public static void main(String[] args) {
- // code to be executed
- }
- The main method takes an optional argument, "args", which is an array of strings that ca
- n be used to pass command-line arguments to the program.

Steps that the JVM takes to call the main method:

- Loads the necessary classes for the program.
- Locates the entry point class specified on the command line.
- Locates the main method in the entry point class.
- Sets up the environment and executes the code inside the main method.
- When the main method completes, the JVM terminates the program.

JDK 11

✅ Keywords & Identifiers

- Simple English words which are having predefined meaning in Java Programming Language.
- Keywords are also called as Reserved Words.
- All the keywords are defined in Lower Case.
- We can't use keywords as names for variables, methods, classes, or as any other identifiers.

List Of Keywords			
Data types (8)	boolean short float	byte int double	char long
Class & Object (9)	class extends super	interface implements new	enum this instanceof
Access Modifier (3)	private	protected	public
Modifier (9)	final synchronized static	native transient const *	abstract volatile strictfp
Package (2)	package	import	
Control Statements (12)	if case while continue	else default break return	switch do for goto *
Exception Handling (6)	try throw	catch throws	finally assert
Other Data type (1)	void		

Identifiers - names those will be used to identify the programming elements like classes, methods, variables, etc uniquely

Rules to follow when you define an Identifier:

1. Identifier can contain Alphabets, Digits, and two special symbol i.e. Dollar ($),
2. Underscore (_).
3. First character of an identifier must be an Alphabet or Dollar ($) or Underscore (_).
4. Keywords or Reserved words can't be used as Identifier.

Valid Identifiers	Invalid Identifiers
Hello Int getClassName studentEmail TOTAL_FFE	1stClass true student number student-email int

✅ Variables and Data Types in Java

Variables

- A variable is a container (storage area) used to hold data.
- Each variable should be given a unique name (identifier).
- Memory will be allocated for the variable while executing the program
- Value of the variable can be changed any number of times during the program execution