**JVM, JRE AND JDK**

JVM- Java virtual machine

* it’s an abstract computing machine that enables a computer to run Java program.
* It is an interpreter that runs the byte codes and executes the program.
* Performs the tasks like loading byte code, code verification and code execution.
* VM is responsible for managing memory, garbage collection, and security.
* Java source code->java complier->java class file->JVM->output in operating system.

JRE- Java runtime environment

* JRE (Java Runtime Environment) is a software package installed in our computer that allows java program to run but we cannot compile it.
* It includes the Java Virtual Machine (JVM) and class libraries necessary to run Java programs.
* In other words, if you only want to run Java programs on your computer, you only need JRE.
* JRE includes browser, JVM, applet supports and plugins.

JDK- Java Development kit

* It is a complete java development kit that includes everything necessary for developing Java applications.
* To execute a java program, we need to install JDK in our computer.
* It includes JRE, JVM, and development tools such as javac (Java compiler), jar (Java archiver), and javadoc (Java documentation generator).
* JDK is used by developers to create, compile, and run Java programs.
* JDK physically exist.
* JVM+CLASS LIBRARIES=JRE
* JRE+DEVELOPMENT TOOLS(JAVAC)=JDK
* In summary, JRE is for running Java programs, JVM is for executing Java code, and JDK is for developing Java applications.
* Top of Form
* Bottom of Form

**Naming conventions**

* Java naming convention is a rule to follow as you decide what to name your identifiers such as class, package, variable, constant, method, etc.
* By using standard Java naming conventions, you make your code easier to read for yourself and other programmers.
* Readability of Java program is very important. It indicates that less time is spent to figure out what the code does.

**1.Lower camel case**

Variables and methods

It should start with lowercase letter.  
It should be a verb such as main(), print(), println().  
If the name contains multiple words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().

**2.Upper camel case**

Classes, interfaces, annotations, enums, records

It should start with the uppercase letter.  
It should be a noun such as Color, Button, System, Thread, etc.  
Use appropriate words, instead of acronyms.

**3.Streaming snake case**

Constants

It should be in uppercase letters such as RED, YELLOW.  
If the name contains multiple words, it should be separated by an underscore (\_) such as MAX\_PRIORITY.  
It may contain digits but not as the first letter.

**4.Lower dot case**

**Packages and property files**

It should be a lowercase letter such as java, lang.  
If the name contains multiple words, it should be separated by dots (.) such as java. util, java.lang.

**DATA TYPES**

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1. **Primitive data types:**

* The primitive data types include Boolean, char, byte, short, int, long, float and double.

1. **Non-primitive data types:**

* The non-primitive data types include [Classes](https://www.javatpoint.com/object-and-class-in-java), [Interfaces](https://www.javatpoint.com/interface-in-java), and [Arrays](https://www.javatpoint.com/array-in-java).

**PRIMITIVE DATA TYPES**

In Java, primitive data types are the basic building blocks of data. There are eight primitive data types in Java:

1. byte: a 1-byte integer, can hold values from -128 to 127.
2. short: a 2-byte integer, can hold values from -32,768 to 32,767.
3. int: a 4-byte integer, can hold values from -2,147,483,648 to 2,147,483,647.
4. long: an 8-byte integer, can hold values from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
5. float: a 4-byte floating-point number, can hold decimal values with up to 7 digits of precision.
6. double: an 8-byte floating-point number, can hold decimal values with up to 15 digits of precision.
7. boolean: a true/false value, can only hold the values true or false.
8. char: a 2-byte character, can hold a single character or a Unicode value.

**NON PRIMITIVE DATA TYPE.**

In Java, non-primitive data types are also known as reference data types. They are not basic building blocks like primitive data types, but they are created using classes, interfaces, and arrays. Non-primitive data types can store a large amount of data and are more complex than primitive data types. Some examples of non-primitive data types in Java include:

* String: a sequence of characters
* Object: the root class of all Java classes
* Array: a collection of variables of the same type
* Class: a template for creating objects
* Interface: a collection of abstract methods that can be implemented by a class
* Enum: a special type of class used to represent a fixed set of constants

Variables of non-primitive data types store the memory address of the object they refer to, rather than the object itself. Non-primitive data types are stored in the heap memory.