JVM, JRE and JDK

JVM → Java Virtual Machine

JVM is an **abstract** machine. It is a **specification** that provides runtime environment in which java **bytecode** can be executed.

The JVM performs following main tasks:

1. Loads code
2. Verifies code
3. Executes code
4. Provides runtime environment

The JVM doesn't understand Java source code, that's why we compile our ***.java** files to obtain ***.class** files that contain the **bytecodes** understandable by the JVM. It's also the entity that allows Java to be a "**portable** language" (write once, run anywhere). Indeed there are specific implementations of the JVM for different systems (Window, Linux…). The aim is that **with the same bytecodes they all give the same results.** JVM is the guy who actually makes a program run. The compiler just gives us a file.

JRE → Java Runtime Environment

The JRE is used to provide **runtime environment**. It is the **implementation** of JVM. It **physically** exists. It contains **set of libraries** + **other files** that JVM uses at runtime. The JRE **does not** contain tools and utilities such as compilers or debuggers for developing applets and applications.

JDK → Java Development Kit

The JDK is a superset of the **JRE**, and contains everything that is in the JRE, **plus tool** such as the compilers and debuggers necessary for developing applets and applications.

The way Java works:

Source Document -> Compiler -> Java bytecode -> Virtual Java Machine

Compiler vs JVM:

JVM is the guy who actually makes a program run. The compiler just gives us a file.