Development Tools

javac

- javac is used to compile a single or a group of Java Programs i.e., Java Source Files.
- Possible version of javac commands is:

  javac [Options] Test.java
  javac [Options] A.java B.java C.java
  javac [Options] *.java

- The Various Possible Options are:

  -version
  -d
  -source
  -verbose
  -cp or -classpath

java

- java command is used to run a single class file.
- Possible Version of java commands is:

  java [Options] <ClassName> <Argument-1> <Argument-2> … <Argument-N>

- The various Possible Options are:

  -version
  -D
  -cp or -classpath
  -ea | -esa | -dsa | -da

Note: We can compile multiple Java Source files at a time but we can run only one Java class file at a time.

Classpath

- Classpath is a path where required .class files are available.
- Java Compiler and JVM will use classpath to locate required .class files.
- By default, JVM will search for .class file in Current working Directory.
- If we want to make JVM to search in specified path, then we are required to set the classpath explicitly. Now JVM won't search in current working directory.
- If we try to run the java class file from any other location other than current working directory without setting classpath, then we are going to get run time exception saying **NoCLassDefFoundError**.

We can set the Classpath in three different ways.

1. By Using Environment Variable Classpath. This is a permanent approach to set classpath. This will persist if we shut down our system also.
2. By Using set classpath=<path> on Command Prompt. This is a temporary approach and will last until the CMD is open. Once CMD is closed, set path will be unset.

3. While compiling or Running Java program we can give the classpath options. This method is also temporary and will exist until the program gets executed completely.

   Example of how to use this method:
   javac -cp <classpath> Test.java
   java -cp <classpath> Test

   Where <classpath> is the path which developer want to specify.

Once we set the classpath, we can run our Java program from any location. Out of these three approaches, setting classpath at program execution level is recommended.

If we have set the classpath, now onwards JVM will not look for the class files in current working directory. Rather, it will look into specified classpath.

How can we make our JVM to first search in current working directory for .class files and if that file is not there search in specified classpath?

java -cp .;E: Test

First searches in current working directory which is specified by (.) and if it is not found in current working directory, then search in E: Drive.

If any location is created because of package creation that location should be resolved by using import or static import statement and base package location we need to update in classpath.

Compiler will check only one level of dependency while JVM will check for all level of dependency.

If we have multiple classpath defined using -cp the order of classpath declaration is important and searching precedence is from left to right.

JAR File

- If there are many dependent class files then instead of adding classpath for all, we can zip them together into a single JAR File and that jar file we can place in classpath.
- All third-party software plugins of Java are by default available in Jar file format.
- For example – To write a simple JDBC Program, we need OJDBC14.jar file in Classpath, then only we can write our program. This file contains many compiled version of Classes like Connection, Statement, ResultSet etc.

Various Commands for JAR Files

- To Create a Jar File (Zip File)
  1. Jar -cvf <FileName>.jar <ClassName>.class
  2. Jar -cvf <FileName>.jar <ClassN1>.class <ClassN2>.class … <ClassNN>.class
  3. Jar -cvf <FileName>.jar *.class
  4. Jar -cvf <FileName>.jar *.*

  Here command 1 is used if we want to create a Jar file with a single class. Command 2 is used if we want to create a Jar file for multiple Class Files specified by class Names.

Command 3 is used if we want to consider all the .class files present in a current working directory or specified classpath and Command 4 is used to consider all the files present in a current working directory or specified classpath.

- To extract a Jar File (Unzip File)
  1. Jar -xvf <FileName>.jar

- To Display table of contents of Jar File
  1. Jar -tvf <FileName>.jar

- While compiling any program which is using JAR File, we need to set the Jar file name also along with setting class path.
- We can place JAR file at JDK/JRE/lib/ext path. If we do so, the classes and Interface present in JAR File will be by default available to Java Compiler and JVM. In this case we are not required to set classpath explicitly.

How to create an Executable Jar file?

Let's create a small sample Program:

```
1  import java.awt.*;
2  import java.awt.event.*;
3
4  class JarDemo
5  {
6      public static void main(String[] args) {
7          Frame f = new Frame();
8          f.addWindowListener(new WindowAdapter(){
9              public void windowClosing(WindowEvent e){
10                 for(int i=1; i <= 10; i++)
11                 {
12                     System.out.println("I am closing Window : " + i);
13                 }
14                 System.exit(0);
15             }
16         });
17         f.add(new Label("Executable JAR File"));
18         f.setSize(500,500);
19         f.setVisible(true);
20     }
21 }
```

When we compile this program by using javac command as:

javac JarDemo.java

This Compilation will create two .class files. One is JarDemo.class and other one is JarDemo$1.class which is a .class file for Anonymous Class.

Now to create an executable Jar file we have to run the command:

jar -cvfm <Jar FileName> <Manifest File> <Java Class 1> <Java Class 2> … <Java Class N>

For Above code Command will be:

jar -cvfm Demo.jar SampleManifest.txt JarDemo.class JarDemo$1.class

In Manifest file we have to define the Main Class Name followed by Enter Key as:

Main-Class : JarDemo

Now we can run this Jar file either from Command Prompt or by directly double clicking Jar File.

In CMD –> java -jar <Jar File Name>

Ways to Run Java Program

- By Running Java Class from CMD – java <class Name>
- By Running JAR File from CMD - java -jar <Jar File Name>
- By Double clicking on JAR File.
- By Double clicking on BATCH File. Batch File is nothing but a file with .bat extension where we can write all our required Java commands and when we double click that batch file all the written commands will be executed in sequence on the CMD.

System Properties

- For every System, some persisitant information will be maintained in the form of System Properties.
- This includes OS Name, Java Version, JVM Vendor, User Country etc.
- The main advantage of setting System Properties is we can customize the behaviour of Java Program.

Demo Program to Print System Properties

Main.java

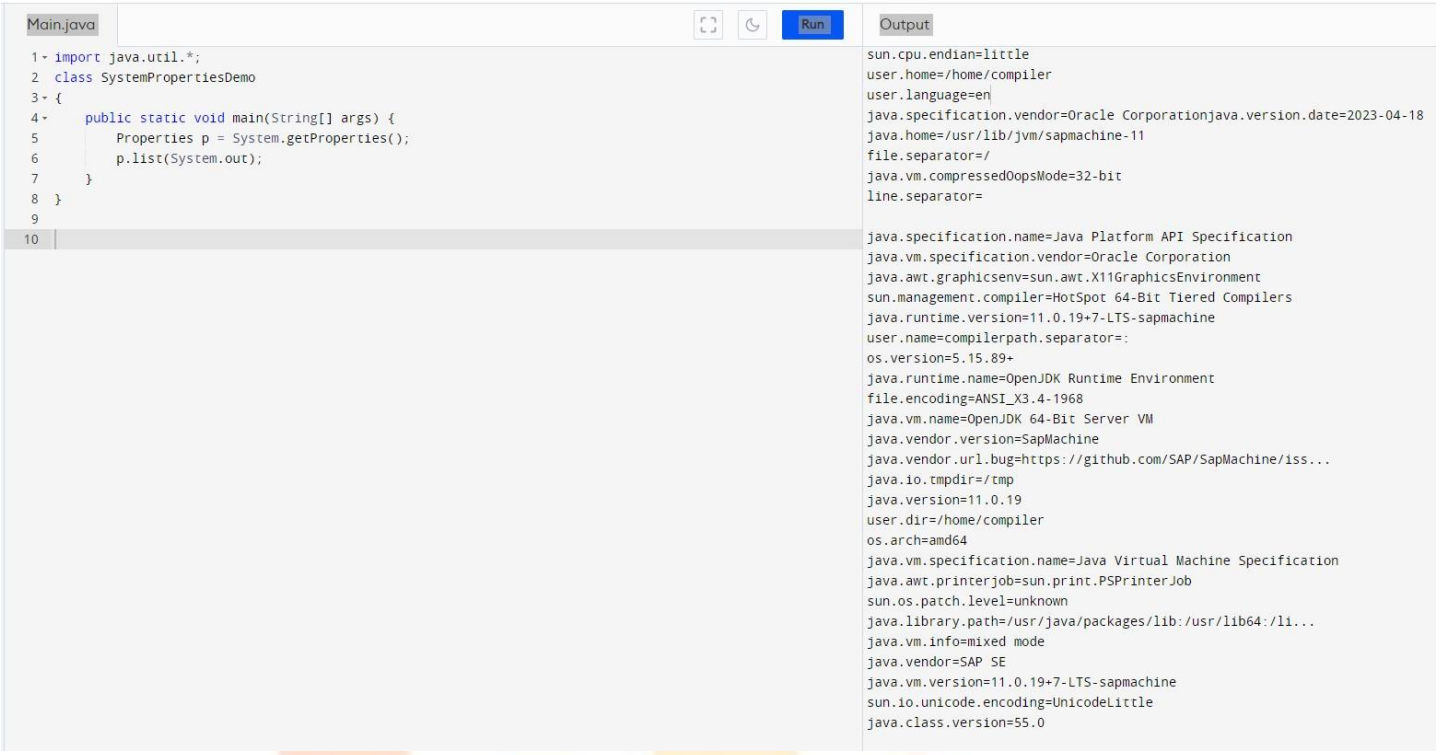
Run

```
import java.util.*;
class SystemPropertiesDemo
{
    public static void main(String[] args) {
        Properties p = System.getProperties();
        p.list(System.out);
    }
}
```

Output

```
java -cp /tmp/v7BB3OkbUR SystemPropertiesDemo
-- listing properties --awt.toolkit=sun.awt.X11.XToolkit
java.specification.version=11
sun.cpu.isalist=
sun.jnu.encoding=ANSI_X3.4-1968
java.class.path=/tmp/v7BB3OkbUR
java.vm.vendor=SAP SE
sun.arch.data.model=64
java.vendor.url=https://sapmachine.io/
user.timezone=
os.name=Linux
java.vm.specification.version=11
sun.java.launcher=SUN_STANDARD
user.country=US
sun.boot.library.path=/usr/lib/jvm/sapmachine-11/lib
sun.java.command=SystemPropertiesDemo
jdk.debug=release
sun.cpu.endian=little
user.home=/home/compiler
user.language=en
java.specification.vendor=Oracle Corporationjava.version.date=2023-04-18
java.home=/usr/lib/jvm/sapmachine-11
file.separator=/
java.vm.compressedOopsMode=32-bit
line.separator=

java.specification.name=Java Platform API Specification
java.vm.specification.vendor=Oracle Corporation
java.awt.graphicsenv=sun.awt.X11GraphicsEnvironment
sun.management.compiler=HotSpot 64-Bit Tiered Compilers
java.runtime.version=11.0.19+7-LTS-sapmachine
user.name=compilerpath.separator=:
os.version=5.15.89+
java.runtime.name=OpenJDK Runtime Environment
file.encoding=ANSI_X3.4-1968
```



We can set System Property Through Command Line as:

java -D<PropertyName>=<PropertyValue> Test

-D is use here to set System Properties.

Test is a class file to run.

We can set System Properties in Java Program by using

System.setProperties("PropertyName", "PropertyValue");

JAR vs WAR vs EAR

JAR	WAR	EAR
JAR Stands for Java Archive. Hence, JAR File is Java Archive File.	WAR Stands for Web Archive. Hence, WAR File is Web Archive File.	EAR stands for Enterprise Archive. Hence, EAR file is an Enterprise Archive File.
A JAR File represents one Java Program which Contains Various .class Files.	A WAR File represents one web application which contains servlets, JSPs, HTML Pages, CSS, JS Files etc.	A EAR File represents one enterprise application which can contain Servlets, JSPs, EJBs, JMS Components etc.
The Only advantage of Maintaining a JAR file for a Java application, it is easy to deploy and host.	The Only advantage of Maintaining a WAR file for a web application is Project Deployment, Delivery and Transportation becomes easy.	The Only advantage of Maintaining an EAR file for an enterprise application is Project Deployment, Delivery and Transportation becomes easy.

In General, EAR file is group of JAR files and WAR Files.

Web vs Enterprise Application

A Web application can be developed by using Web Related Technologies like Servlets, JSPs, HTML, CSS, JS etc. Example of Web Application includes Online Library Management, Online Shopping cart etc.

An Enterprise Application can be developed by using any technology from Java J2EE which includes all web related Technologies and in addition to that EJBs, JMS components etc. Example – Banking Applications, Telecom Based Project etc.

J2EE or JEE compatible applications are Enterprise applications.

Web Server vs Application Server

Web Servers provides an environment to run Web Application. Web Servers provides support for Web Related Technologies like Servlets, JSPs, HTML, CSS, JS etc. Example of Web Server is Tomcat Web Server.

An application server provides an environment to run Enterprise Application. Application servers can provide support for any technologies from Java J2EE which includes all web related Technologies and in addition to that EJBs, JMS components etc. Example of Application Server is Web Logic, JBoss etc. Every application server contains an inbuilt web server to provide support for Web based technologies. If any programmer doesn't want an inbuilt web server then in that case he can go for any other web servers.

J2EE or JEE compatible server is Application Server.

Path vs Classpath

Classpath describes a path at where all the required .class files are present. Java compiler and JVM will use this classpath to locate all the required .class files. If we are not setting classpath then our program may not compile and may not run.

Path describes a location at which binary executables are available. For example – To compile a Java Program we need javac.exe (Java Compiler) which is a binary executable file present in folder JDK/bin. So, we need to configure this JDK/bin in the path. If correct path is not set were javac and java are present then while compiling and running java program we will get "javac is not recognized as an internal or external command, operable program or batch file".

When correct path is not set.

```
C:\Users\vikas>java --version
'java' is not recognized as an internal or external command,
operable program or batch file.
```

When correct path is set.

```
C:\Users\vikas>java --version
java 20 2023-03-21
Java(TM) SE Runtime Environment (build 20+36-2344)
Java HotSpot(TM) 64-Bit Server VM (build 20+36-2344, mixed mode, sharing)
```

Difference between JDK, JRE and JVM

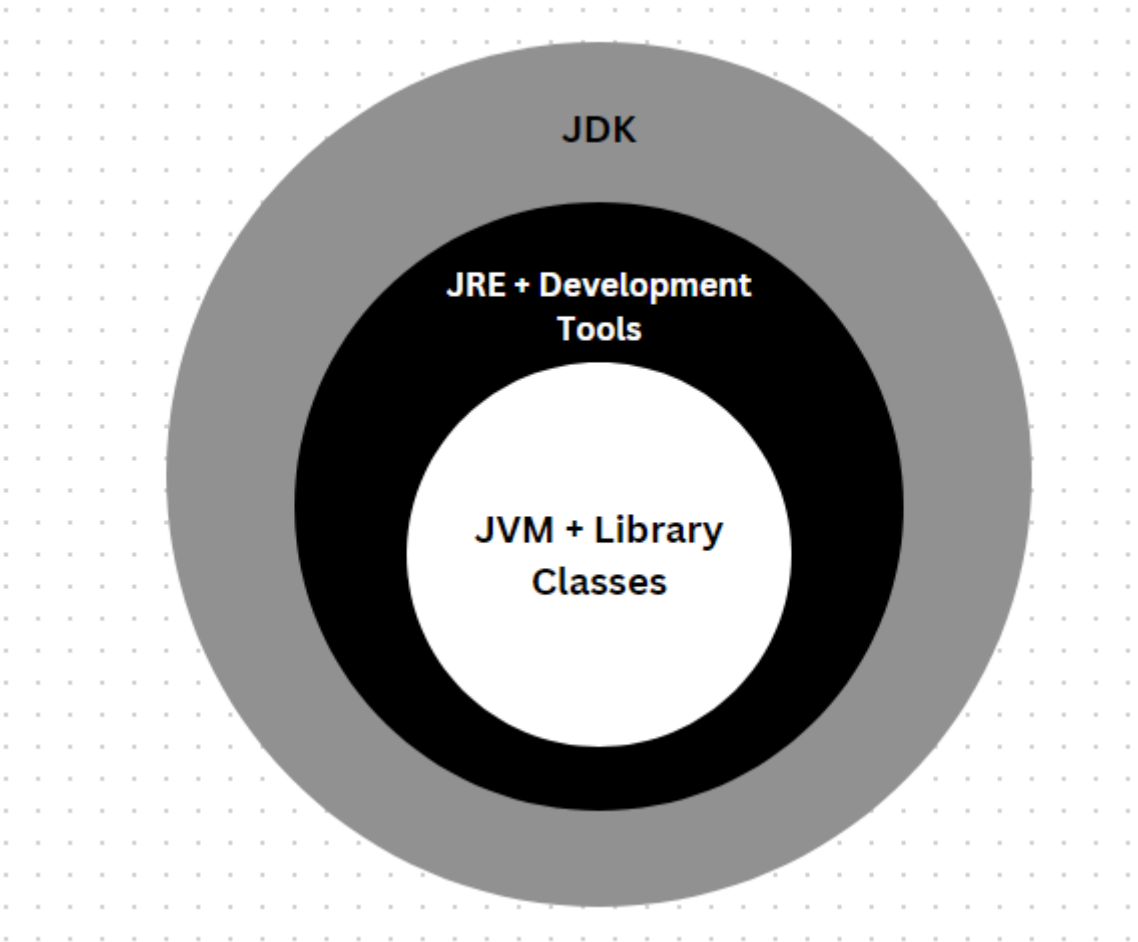
JDK (Java Development Kit) provides an environment to develop, compile and run the java application.

JRE (Java Runtime Environment) provides an environment only to run the Java Application.

JVM (Java Virtual Machine) is an interpreter which is responsible to execute Java program line by line.

JDK = JRE + Development Tools

JRE = JVM + Library Classes.



Note: On Client Machine, it is compulsory to have JRE but on Development Machine, it is compulsory to have JDK.

Java vs Javaw vs Javaws

We can use Java command to run Java class file where output of System.out.println() will be printed on the Console.

We can use Javaw command to run Java class file where output of System.out.println() will not be printed on the Console. In general, we can use Javaw command to run GUI based applications.

We can use Javaws (Java Web Start Utility) command to download java application from the web and start its execution. We can use Javaws command as follows:

Javaws Jnlp_url

It downloads an application from the specified URL and start executions.

The main advantage in this approach is every end user will get updated version and enhancement will become easy because of centralized control.