**JAVA-James Gosling -1955**

AWT and Swing are used in Java for creating standalone applications.

Java Applications-> standalone, webapp, enterprise,mobile

Java Platforms\_Editions ->SE,EE,ME,FX

**JVM**

A specification where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies

fn->Memory area,Class file format,Register set,Garbage-collected heap,Fatal error reporting etc.



Classloader->**Bootstrap ClassLoader(rt.jar loads java classes),Extension**

**ClassLoader(load installed java lib jars), System/Application ClassLoader(loads class path jars)**

**Rt jar contains core classes like java.lang**

**Java Native Interface->interacts with other languages and OS**

**Execution Engine-> virtual processor, JIT,interpretor**

Class file contains bytecode

Semicolon at the of class defination is optional

Main method cannot be abstract

Valid main class initializations

public static void main(String[] args)

public static void main(String []args)

public static void main(String args[])

public static void main(String... args)

static public void main(String[] args)

public static final void main(String[] args)

final public static void main(String[] args)

final strictfp public static void main(String[] args)   
  
java classes can be saved with different name then file if not public

set path=C:\Program Files\Java\jdk1.6.0\_23\bin

JVM, JRE, and JDK are platform dependent

JRE->JVM + libraries liie rt.jar  
jvm can not run independent requires jre or jdk

JDK->jre+devlopment tool to build SE,EE,ME,FX

Javac is part of jdk only

1. **byte** a=10;
2. **byte** b=10;
3. //byte c=a+b;//Compile Time Error: because a+b=20 will be int
4. **byte** c=(**byte**)(a+b);  // if a+b exceed byte range it starts from -128 for eg a+b->128 in byte It will be -128 129->-127

Short, byte and char converts to int when arithmetic operation occurs

System.***out***.print((String)**null**);

**int** a=129;**byte** b=(**byte**)a;System.***out***.println(b);

**char** c='c';**char** d='e';System.***out***.println(c+d);

System.***out***.println(~129);

**for**(**int** i=0,**int** j=0;1;i++,j++) {}// in java condition shoud only be boolean in c++ it can be integer and will run infinite in c++ 2 variables could be intialize in loop in java only one is supported but two variable can incremented in java

labels

a:

**for**(**int** i = 0; i<= 10; i++) {

b:

d:

**for**(**int** j = 0; j<=15;j++) {

c:

**for** (**int** k = 0; k<=20; k++) {

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

**if**(k==5) {

**break** b;

}

}

}

}

// one line comment  
/\*

Er

\*/ multiline

/\*

\*hello

\*/Javadoc comment

1. **{@docRoot}**: This tag is used to represent the relative path to the root directory of the generated document from any page. It's often used in documentation comments to reference resources or links within the generated documentation.
2. **@author**: Used to specify the author of the class or code snippet. It documents the authorship information in the generated documentation.
3. **@code**: Formats the enclosed text in code font without interpreting it as HTML markup or nested Javadoc tags. It's useful for displaying code snippets or specific identifiers within documentation comments.
4. **@version**: Specifies the version of the code or API. When the **-version** option is used with the Javadoc tool, it generates a "Version" subheading with the specified version text in the documentation.
5. **@since**: Indicates the version or release in which a particular feature or functionality was introduced. It's used to document the version from which certain aspects of the code are available or applicable.
6. **@param**: Documents a method parameter by specifying its name and description. It's used to describe the purpose of each parameter in a method, and this information appears in the "Parameters" section of the generated documentation.
7. **@return**: Describes the value or object returned by a method. It's used to provide information about the return value of a method, especially when the method doesn't return **void**.

\u000d Next line in unicode so the stetments shifts to nxt line and executes print statement

/ \u000d System.out.println("Java comment is executed!!");

RMI,EJB