Lecture 3

Topics Covered :-

- Writing our First Java Program in Text Editor
- Understanding Compilation & Execution Flow of Program
- Understanding the Structure of first Java Program
- Platform Dependency vs. Platform Independency
- JShell Introduction

First Java Program :-

few commands :-

Open Command prompt & write :-

Two Commands → { javac -version or java -version

to check we have correct JDK version installed on our laptop

in my case (laptop) it is showing javac 21.0.4 for the first command & java version "21.0.4" for second command

So now we will write our first Java Program in any Text editor. Let's open NotePad & write.

Before writing our java program, first create a directory (folder) in which we will save our all java programs.

Command :→ `md javaprograms`

↳ name of the folder (directory)

or `mkdir javaprograms`

We can use either md or mkdir command to make directory (folder)

keyword → class FirstProgram ← classname

```
{
    public static void main (String[] args)
    {
        System.out.println ("Hello Welcome to Jenny's Lectures");
    }
}
```

Save it as FirstProgram.java

Questions? ① Why class name & file/Program name is same?

Ans IMP It need not be same for every java program. Here we can have a different name to save this file

→ But if the class is public then it is necessary to have both class name & file/program name same for better organization of the code

→ And Generally we declare class public to make it accessible everywhere; if we dont declare it public then the class would be package-private means it can not be used in another package.

⇒ [packages helps group related classes, making it easier to manage & maintain code, avoid naming conflicts and control access]

we will see packages in detail later.

NOTE:- Conceptually, it is similar to folders on your laptop

So this same name convention was set by Sun, so that everyone who codes in Java will have a consistant way of naming files.

(2) Why haven't we included any header or package here like we include in C & C++ (#include <stdio.h> & #include <iostream>) to program work?

Ans:- The Java.lang package is automatically imported by the Java compiler for every java program. So no need to import it explicitly. If you want you can write:

```
import java.lang.*;
```

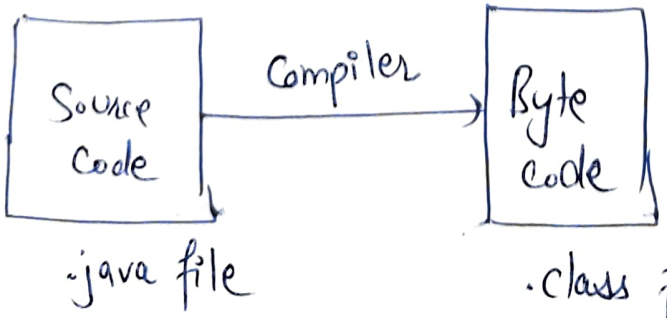
→ But if we want to use classes from other packages, then we have to import those explicitly.

→ java.lang package is one of the core packages in java & it contains fundamental classes that are essential for java programming eg:- System, String, Object, Thread, Math (to perform basic mathematical functions), Exception etc.

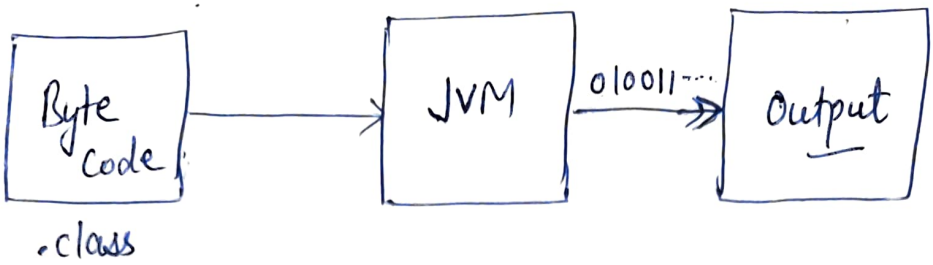
Understanding the Compilation & Execution Flow of the Program:-

This is 2-step process: ① Compilation
② Execution.

① Compilation:-



Step 2 - Execution :-



→ Bytecode is an instruction set of JVM & each instruction is represented by 8 bits (one byte). Hence it is known as bytecode. It is compact form of data.

How JVM converts bytecode | How JVM works ?

→ JVM is abstract computing ~~machine~~ machine having instruction set (bytecode) & manipulates memory cells at run time.

→ JVM knows nothing of Java Programming language but it knows the class file format. A class file contains Java virtual Machine instructions (or bytecodes) and a symbol table. and some other ancillary information

When we execute the command `java HelloJenny` then JVM would be launched means an instance of JVM would be created in memory and it also has some JVM memory from OS

→ Now class Loader loads the .class file into memory (JVM memory). Now the bytecode would be verified by bytecode verifier to ensure that the ~~bycode~~ bytecode adheres to the rules of the java language & doesn't violate access rights and also do some other checking on bytecode.

Then if everything is ok in the bytecode, execution engine executes the instruction with the help of interpreter & JIT Compiler. Interpreter interprets the code line by line & execute & for some frequently called methods JIT compiler comes into picture & may compile it into machine code & then this machine code is cached in memory. Subsequent call to that method will execute the native code instead of interpreting the bytecode again, significantly speeding up execution.
And then you get the output.

Understanding the Structure of a class / of first Program -

~~public class Hello~~

```
class FirstProgram {
```

↓ keyword / reserved word

↳ Name of the class

```
    : :
}
```

[while naming the class, we should follow Upper CamelCase or Pascal Case Convention]

main method

```
Public static void main (String [] args)
{
    : :
}
```

→ main method is public so that everyone can access it. If we don't make it public JVM will not be able to access it.

program will be compiled correctly. but will you run it will give error "main method not found". So we have to make it public to make it accessible for JVM.

* Static:- Program execution starts from main() method. Declaring it static allows the JVM to invoke the main method without creating an instance of the class containing it.
  → If main were not static, the JVM would need to create an object of the class before invoking the method., which would lead to complexities.
  → The main method is directly called by the JVM when program starts. The static main method is a convention or standard entry point in Java as well as in some other programming languages., making it easier for the JVM & developers to recognise the program's starting point.
→ void is just a return type, void means there is no return value. main is the name of this method.
⇒ If we dont declare main method as static, program will be compiled correctly but it will not run. It will give error "main method is not static."

* String[] args:- these are arguments to main method. This method must be given an array of strings and the array name is args. (we can take any array name other the args if we want)

  eg:- String[] a is also fine.

It is a way to pass command line arguments [we will discuss this thing in later videos]

The String[] args parameter in the main method is part of the standard Java method signature for the program entry point. Without it JVM won't recognise the main method & may not be able to start the program.
Without String[] args the program will compile successfully but will not run because of the violation of the conventional signature expected by JVM.

⇒ [System.out.println("Hello Welcome to Jenny's Lectures");
↳ this is a way to print output to standard output in Java.

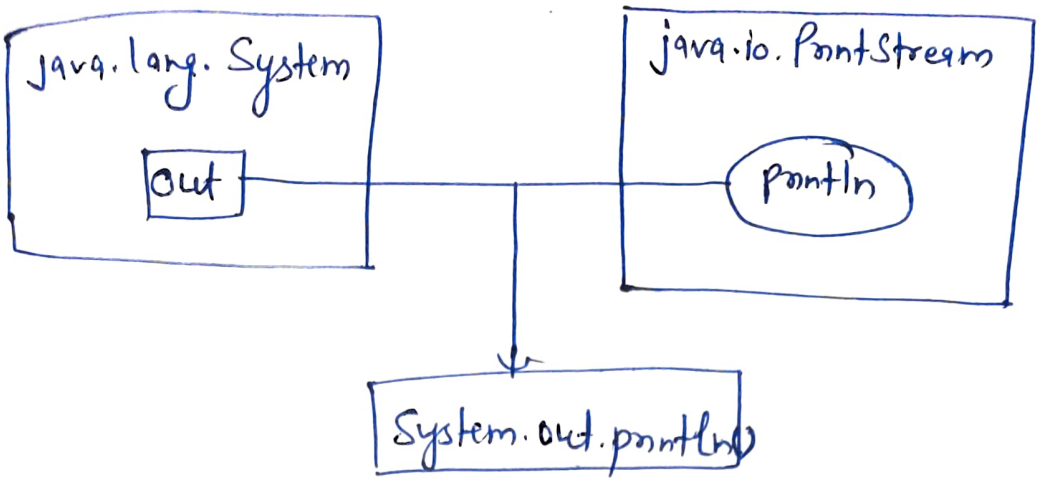
→ println() is a method in PrintStream class used to print any argument passed to it & adds a new line to the output

out is an instance/object of PrintStream class and out is a static member of System class. out member/field represents the standard output stream (usually the console) to which text and data can be printed.

System is a final class in the java.lang package. Since java.lang package is imported automatically by the JVM, we don't need to explicitly import System. System class provides various fields & methods for standard input, output & error streams & much more.

→ So Basically System.out refers to the standard output stream typically connected to the console. We can redirect this output to a file or another output destination if necessary

→ we can use print() method rather than println(). print() also prints the arguments passed to it but it doesn't add a newline after the output.



Practice Questions:-

Que ① Write a program to print your name, age & hobby on separate lines.

Que ② Write a program to print the following pattern:-

(i)
```
*
* *
* * *
```

(ii)
```
  *
 * *
* * *
```

Que ③ Write a program to print sum of integers (10+15) using System.out.print() & System.out.println()

Que ④ What would be the output of the following code:-

```
class PracticeProgram
{
    public static void main( String[] args)
    {
        System.out.print (10+15);
        System.out.println ("10");
        System.out.println ("10+15");
        System.out.println (12.156);
        System.out.print('a');
    }
}
```

Introduction to JShell

→ JShell is an interactive tool for learning the Java programming language and prototyping Java code.

→ JShell is a Read-Evaluate-Print Loop (REPL), which evaluates declaration, statements & expressions as they are entered & immediately show the results

→ This tool is run from the command line

JShell was introduced in Java 9

* Starting and Stopping JShell →

→ To start JShell, enter the jshell command on the command line

> NOTE:- JDK 9 or higher versions must be installed on your system.

To exit JShell, enter /exit

To start JShell in verbose mode, use the -v option:

```
jshell -v
```

Code Snippets:-

* JShell accepts Java statements; methods, variable, & class definition; import & expressions. These pieces of Java code is known as snippets

eg:-

```
jshell > int a = 5;
a ==> 5
| created variable a : int
```

⇒ a description of what occured is shown as we are in verbose mode.

NOTE:- Semicolons are automatically added to the end of a complete snippet if not entered.

→ When an expression is entered that doesn't have a named variable, a scratch variable is created so that its value can be referenced later.

eg:-
```
jshell > 3+5
$1 ==> 8
| created scratch variable $1 : int
```

→
```
jshell > void display ( ) {
...> System.out.print ("Hello Welcome to Jenny's lectures");
...> }
| created method display ()

jshell > display ()
Hello Welcome to Jenny's Lectures
```

There are many Jshell commands used to control the environment and display information

eg:- /vars - gives information about current variables
/methods - gives info. about current methods
/list - gives a list of entered snippets

NOTE:- Jshell has a default startup script that is silently & automatically executed before Jshell starts, so that we can get to work quickly. To show all the entries from startup we use /list -all or /list -start command

/help → list of all the Jshell commands

(2) So JShell is a REPL for Java which means -

(i) it reads the command or code segment we type
(ii) it evaluates and execute the code
(iii) it prints the result/output without making the developer write code to output the results.
(iv) And then it loops right back for more input

NOTE JShell doesn't replace an IDE.

Few Definitions :-

Statement - It's a complete command to be executed.
It is an instruction for computer to do something
e.g:- System.out.print(3+5);

Expression:- expressions are part of statements
An expression is a combination of variables, values & operands that produce a result or value when evaluated.

eg:- 3+5,
or (9+5)*3

expression always return a value and can be part of statements

```
int a = 5+3;   ] = Statement
int a;         } statements
x = 10;
```
↳ 10 is an expression

So Expressions produce a value when evaluated & Statements perform an action and may or may not contain expressions

Que Can't we have more than one class in a java file?

Ans. Yes, you can have but only one class can be public & the name of public class & name of java file must be same.

But it is not compulsory to have a public class, all classes can be without public specifier.

Que:- Suppose we have 3 classes in one java file. So after compilation three class files would be generated. So in this case we are not able to identify which class contains main method (entry point) for JVM to execute the program?

One Solution is check the source code & find out which class has main method but this is tedious work for any one. So we are forced to follow some convention rules set by Sun, just for better organization of the code.

So it's recommended to have one public class per java file but if you have multiple classes in single java file then make one class public which is having main method (entry point) & save the file with same name as public class name so that while running its easy to identify which class has main & to be executed.

Better to have nested classes if you need multiple classes in a java file.

It is not compulsory that the public class should have main method.