

2

Day 2_Java



4 Sept 2024



Learn about

Native code when c++ data can be used ⇒ native method stacks

Java is interpreted or compiled ??

Learn about different Loaders.

General Info : — version 7

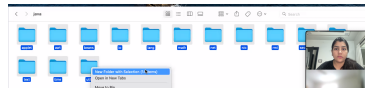
- Files with .java extension means they are source files .
- Java is a open source technology ⇒ source code is freely available
- JVM is also open source .
- In lang folder , we can find all the classes and find the source code of various classes.
- In java without new keyword , we can't make instance . other than it is reference variable but if the type of the var is a some class then it is called reference object of that class.

- The `println` method is a part of the `PrintStream` class, which is located in the `java.io` package.
 - Package:** `java.io`
 - Class:** `PrintStream`
 - Method:** `println`
- The `System.out` in Java refers to an instance of `PrintStream`, which is why you can use `System.out.println()` to print to the console.
- Inside jre / lib ⇒ we can get the compiled code of the source code of java api .
- During execution we need .class ⇒ that's why rt.jar file in the folder jre .
- src.jar file contains source code and rt.jar contains compiled source code .

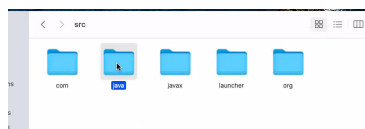
In Java, the `src.jar` and `rt.jar` files have distinct purposes:

- src.jar :**
 - Contents:** This file contains the source code of the Java standard library. It's a JAR (Java ARchive) file that includes the `.java` source files, which are the actual human-readable code.
 - Purpose:** It's primarily used for reference, debugging, or studying how the Java standard library is implemented. Developers can look at the source code to better understand how certain classes and methods work.
- rt.jar :**
 - Contents:** The `rt.jar` file contains the compiled bytecode of the Java standard library. It includes the `.class` files, which are the compiled versions of the Java source code that the Java Virtual Machine (JVM) executes.

These are sub packages



- These main packages when we unzip file the zip file src file(contain source code of java api) in lib folder .



Session Overview

- Components of JVM
- Java Buzzwords
- Java Modifier
- Access Modifier
- Java Virtual Machine Threads
- Entry Point Method
- Meaning of `System.out.println`
- Data Types
- Classification of Data Types
- Primitive Data Types
- Wrapper Class Hierarchy
- Overview of String
- Memory representation
- Widening Conversion
- Narrowing Conversion
- Command line Arguments

Java Platform SE 8

<https://docs.oracle.com/javase/8/docs/api/>

- **Purpose:** This file is essential for running Java programs. The JVM uses the classes in `rt.jar` during the execution of Java applications.

Note: Starting from Java 9, `rt.jar` was replaced by the module system (`jmod` files) with the introduction of the Java Platform Module System (JPMS). In newer versions of Java (9 and above), the Java runtime is modularized, and `rt.jar` no longer exists as a single file.

A Folder/directory which contains multiple projects is called as Workspace .

```
class Hello {
    Public static void main(String[] args(Can give anything as parameter)
    System.out.println("")
}

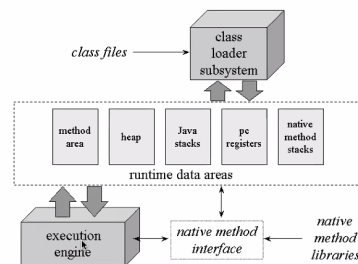
=> public static all are modifiers .
=> Package level private => default of the class var and method.
=> Type => enum , interface , class => all produce .class files
=> Java compiler generates .class file of every class present in a fi
=> For executing the file after compilation , we should use in termir
=> Only that .class will work which contain main method .
=> If the file is empty , having no class , no compile type error but
=> Giving different file anme and class name is not a good practice .
```

```
J Program.java
Day_2_1 > J Program.java @C
1 interface A( )
2
3 class B( )
4
5 enum C( )
6
```

```
J Program.java
Day_2_1 > J Program.java @C
1 class Hello{
2     public static void main( String[] args){
3         System.out.println("Hello World!");
4     }
5 }
```

Components of JVM

- **Class loader subsystem**
 1. Bootstrap class loader
 2. Extension class loader
 3. System class loader
 4. Custom class loader
- **Runtime data areas**
 1. Method area
 2. Heap
 3. Java Stacks
 4. PC Register
 5. Native method stacks
- **Execution engine**
 1. Interpreter
 2. Just In Time Compiler
 3. Garbage Collector



Learn about different Loaders.

home/jre/lib/rt.jar ⇒ Bootstrap path

home/jre/lib/ext ⇒ extension path

There are 12 modifiers in Java:

1. private	:	Access Modifier
2. protected	:	Access Modifier
3. public	:	Access Modifier
4. static	:	Non Access Modifier
5. final	:	Non Access Modifier
6. abstract	:	Non Access Modifier
7. interface	:	Non Access Modifier
8. transient	:	Non Access Modifier
9. synchronized	:	Non Access Modifier
10. volatile	:	Non Access Modifier
11. strictfp	:	Non Access Modifier
12. native	:	Non Access Modifier

What are buzzword : Used by owners for the marketing of the java tech

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Object oriented
- Distributed
- Multithreaded
- Dynamic
- Architecture neutral
- Portable
- High performance
- Robust
- Secure

• used **"jcmd"** ⇒ we can the process id

• using **jstack** ⇒ we can the thread dump

• If a program requires multiple threads to start execution, it is called multi-threaded. One of these threads is the main thread, which is responsible for starting the program.

• Default garbage collector ⇒ ParallelGC which is also preset in threaddump

Main Method

- Java compiler will not check whether main method is present or not
- it doesn't check the syntax of main method
- on executing we will get the error main method not found
- access modifier of the main should be public strictly
- main should be static
- and must return void only
- and name should also be in lowercase main
- we change the name of parameter in main and shift subscript position
- Can write (String... args).
- Cannot define multiple main with different syntax also in same class.
- Can define main method per class in a file, only one of them will be considered as
- entry point and rest of them will be same as normal functions.
- We can overload the main method in java.

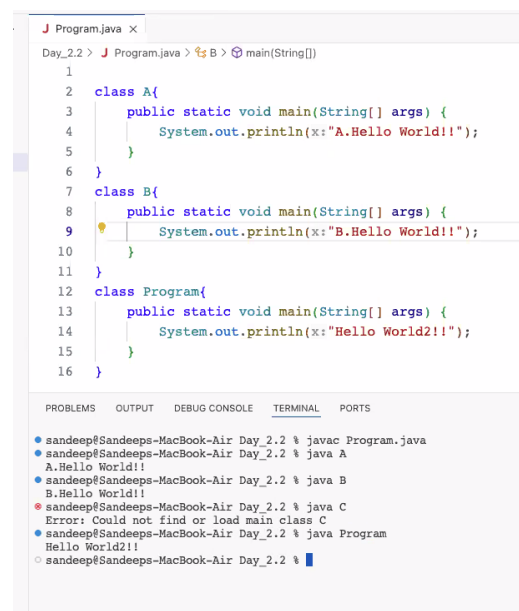


```
Day_2.2 > J Program.java > Program > main(String[])
1 class Program{
2     public static void main(String message) {
3         System.out.println(message);
4     }
5
6     public static void main(String[] args){
7         Program.main(message:"Hello World!!");
8     }
9 }
```

class
ava

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
• sandeep@sandeeps-MacBook-Air Day_2.2 % javac Program.java
• sandeep@sandeeps-MacBook-Air Day_2.2 % java Program
Hello World!!
○ sandeep@sandeeps-MacBook-Air Day_2.2 %
```



```
Day_2.2 > J Program.java > B > main(String[])
1
2 class A{
3     public static void main(String[] args) {
4         System.out.println(x:"A.Hello World!!");
5     }
6 }
7 class B{
8     public static void main(String[] args) {
9         System.out.println(x:"B.Hello World!!");
10    }
11 }
12 class Program{
13     public static void main(String[] args) {
14         System.out.println(x:"Hello World2!!");
15     }
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
• sandeep@sandeeps-MacBook-Air Day_2.2 % javac Program.java
• sandeep@sandeeps-MacBook-Air Day_2.2 % java A
A.Hello World!!
• sandeep@sandeeps-MacBook-Air Day_2.2 % java B
B.Hello World!!
• sandeep@sandeeps-MacBook-Air Day_2.2 % java C
Error: Could not find or load main class C
• sandeep@sandeeps-MacBook-Air Day_2.2 % java Program
Hello World2!!
○ sandeep@sandeeps-MacBook-Air Day_2.2 %
```

Day 2 Recording

https://us02web.zoom.us/rec/share/ctP6sK27-beN-PyhLJ0WAHYVxA1m6Vk56T8Z0sd9keahEVn2tWotCibF3QcEicGn.kbL89e_liryJWGhS

TODO List

1. Java Modifiers.

- a. They are of two types
 - i. **Access Modifier** :They define the security / accessibility / Scope of the field , method , constructor or class .
 1. **Public**
 2. **Private**
 3. **Default**
 4. **Protected**

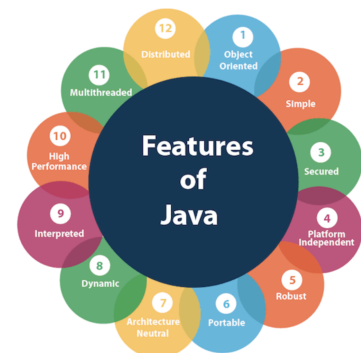
Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

ii. **Non-access Modifier** : **Purple** : Only be used with **Orange** : Can be used with variable and methods .

1. **final** : The class cannot be inherited by other classes
2. **abstract** : The class cannot be used to create objects (To access an abstract class, it must be inherited from another class.
3. **Final** : Attributes and methods cannot be overridden/modified.
4. **Abstract** : It can only be used with methods and they cant have a body and the definition has to be given in the sub class .
5. **static** : This means the var and methods only belongs to the class not the object of that class .
6. **transient**
7. **synchronized**
8. **volatile**
9. **native**

2. Java Buzzword.

- Simple** : Java was designed to be easy to learn and use. If you've seen other programming languages like C++, you'll find Java simpler. The syntax is straightforward, which makes it easier for beginners to start coding.
- OOP** : Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behavior.
⇒ Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.
⇒ Basic concepts of OOPs are:
 1. **Object**
 2. **Class**
 3. **Inheritance**
 4. **Polymorphism**
 5. **Abstraction**
 6. **Encapsulation**
- Platform Independent** : JDK and JVM is platform dependent but the class file which are generated are platform independent. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere (WORA).
- Secured** : It is secured . Java doesn't support pointers . Classloader in Java is a part of the Java Runtime Environment (JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.
- Robust** is strong. Java is robust because:
 - There is a lack of pointers that avoids security problems.
 - Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
 - There are exception handling and the type checking mechanism in Java.
- Architecture-neutral** :



- Java is architecture neutral because there are no restrictions for specific bits kind The size of data types is fixed.
 - In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture.
- g. **Portable** because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.
- h. **High-performance** :
- i. **Distributed** :
- j. **Multi-threaded** A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc. Java can handle multiple tasks at the same time, like multitasking on a computer, thanks to its multithreading capability.
- k. **Flexible** and can adapt as your program runs. It can load new code and use it without restarting the program, which is useful in big applications that need to keep running while being updated.

3 . Components of java :

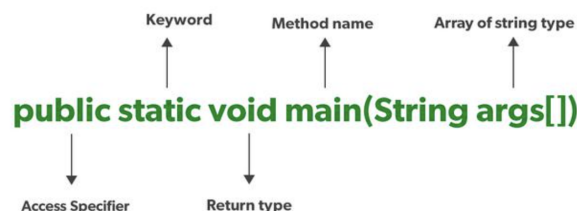
- Java Virtual Machine (JVM): **Java Virtual Machine (JVM)** is the component of the JRE that actually executes Java bytecode. It is an abstract computing machine that enables a computer to run Java programs. The JVM performs several key tasks:
- Java Runtime Environment (JRE): It is the intermediate layer and sole purpose is to run the code . Includes the JVM and essential libraries for running Java applications.
- Java Development Kit (JDK): Responsible for executing Java bytecode. **JDK is a software development kit** that you need to write and run Java programs. It includes tools like the Java compiler (`javac`), which turns your Java code into bytecode, and other utilities(.class file)

4. Meaning of main method. :

- It is the starting point of any program in java (JVMs starts the execution).
- JVM will not execute the code, if the program is missing the main method. It is most important methods of Java.
- The Java compiler or JVM looks for the main method when it starts executing a Java program. The signature of the main method needs to be in a specific way for the JVM to recognize that method as its entry point.

Syntax of main() Method

Syntax of the main() method is always written as:



5. Java Virtual Thread. ⇒ ????

6. Meaning of System in and System out.

In Java, `System.in` and `System.out` are two commonly used objects that are part of the `java.lang.System` class, and they are used for input and output operations.

• System Class:

- `System` is a final class in the `java.lang` package. It has class fields and methods, which are all static, meaning they can be accessed directly through the class without creating an instance of it . In this there exist a in and out object reference variable of `InputStream` (in) and `PrintStream` (out). and these `InputStream` `PrintStream` classes are present are `java.io` package .

7. Size of data types

- **8 bits (1 byte):** `byte`
- **16 bits (2 bytes):** `short`, `char`
- **32 bits (4 bytes):** `int`, `float`
- **64 bits (8 bytes):** `long`, `double`
- **1 bit :** `boolean`

8. Loader in java. →

- a. Bootstrap Class Loader
- b. Extension Class Loader
- c. System Class Loader

9. **Garbage collector** → In Java, the **Garbage Collector (GC)** is a tool within the Java Virtual Machine (JVM) that automatically manages memory for your program. Its main job is to reclaim memory that is no longer being used by the program, so you don't have to manually manage memory as you would in some other programming languages.