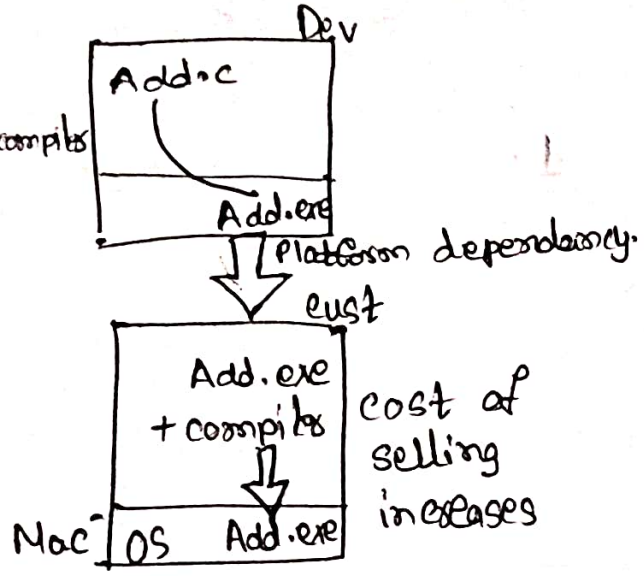
08/11

Ganesh Keerthi
(10 seconds)

Add.c ⇐ Dev ⇒ Source Code

⇩ Drawback ⇒ src to native
Compiler ⇒ .c to .exe

Add.exe ⇐ OS ⇒ Native/Exe/Op/Binary.



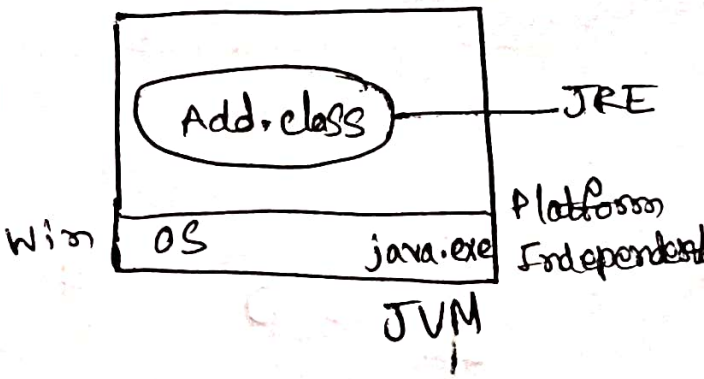
C/C++ not used in Industry due to its platform dependency where the compiler converted .exe file is understood by only one OS that is of the developer.

Add.java ⇐ Dev ⇒ Source Code

⇩

Add.class ⇐ Byte code

WORA



Add.exe ⇐ OS ⇒ Native code

Write Once Run Anywhere — WORA concept.

Files	Win OS	MAC OS	
.jpeg ⇐	Picture	Photos	Interpretors.
.mp3 ⇐	Music Player	iTunes	
.doc ⇐	Word	Pages	
.class			

Java / Python coders are preferred due to its platform independence

Drawback: Java/Python: Slow in execution - Since not running on the OS but on interpretors that act as OS.

C language is used in Device Specific operations.

Java Code execution:

>javac x.java

>java x

After compilation of source code the name of byte code will be equal to name of class written in source code.

2 Step process!:

src code:: file.java ⇒ ~~Interpreter~~ javac.exe

Byte code:: classname.class ⇒ JVM (java.exe).

The number of byte codes generated after compilation of source code will be equal to number of classes written in source code.

10Seconds\AJIET

A.class

B.class

C.class

c:\Program Files\JAVA\JDK 17

\bin	\lib	\src
.exe	.class	.java
javac	String	String
java	System	System

---

public class System{

}

public class String{

}

The names of source code and byte code should be same if a class is declared as public in its source code to

avoid confusion in debugging due to name errors.

Multiple classes can be written in the source code but only one class among should be public provided the name of sourcecode & public class name are same.

---

civil Engineer → Blueprint
↳ kitchen, Parking.
↙
logical construction.

Physical → Mason
↓
Site
} used by the owner.

Software developer → class
↳ variables, methods } members.
↙
logical construction

Physical → JVM (interpreted)
↓ executes
RAM

Types of variables: static, non-static, local
field = variable, States → values.

Working of JVM :: Physical Construction
⇒ OS loads JVM to Stack ⇒ JVM calls inbuilt class Classloader
⇒ ClassLoader → creates 'static context' in Heap → creates static variables and loads definitions of static method
→ labels the address of static context with ClassName
→ will give the address to JVM and exits

Why main() must always be static?

JVM in order to start the program it will go in search of main() in static context.

Why local variables of one method cannot be accessed by another method?

working of static methods

local variables :: directly

static variables :: directly
:: classname (if there is a conflict)

static methods :: directly, classname.

non-static variables ::

non-static methods ::

Callee method cannot call the variable of caller method.

Reason :: LIFO of stack Pointer

(CTE) Error : Variable Not found.

Caller method cannot call the variable of callee method.

Reason :: Life cycle of local variable
:: local variable will be destroyed even before the callee goes out of the stack and the control goes back to caller

CTE (Error) :: variable not found.
