Programming in Java
Introduction

Prep Term 2018-19

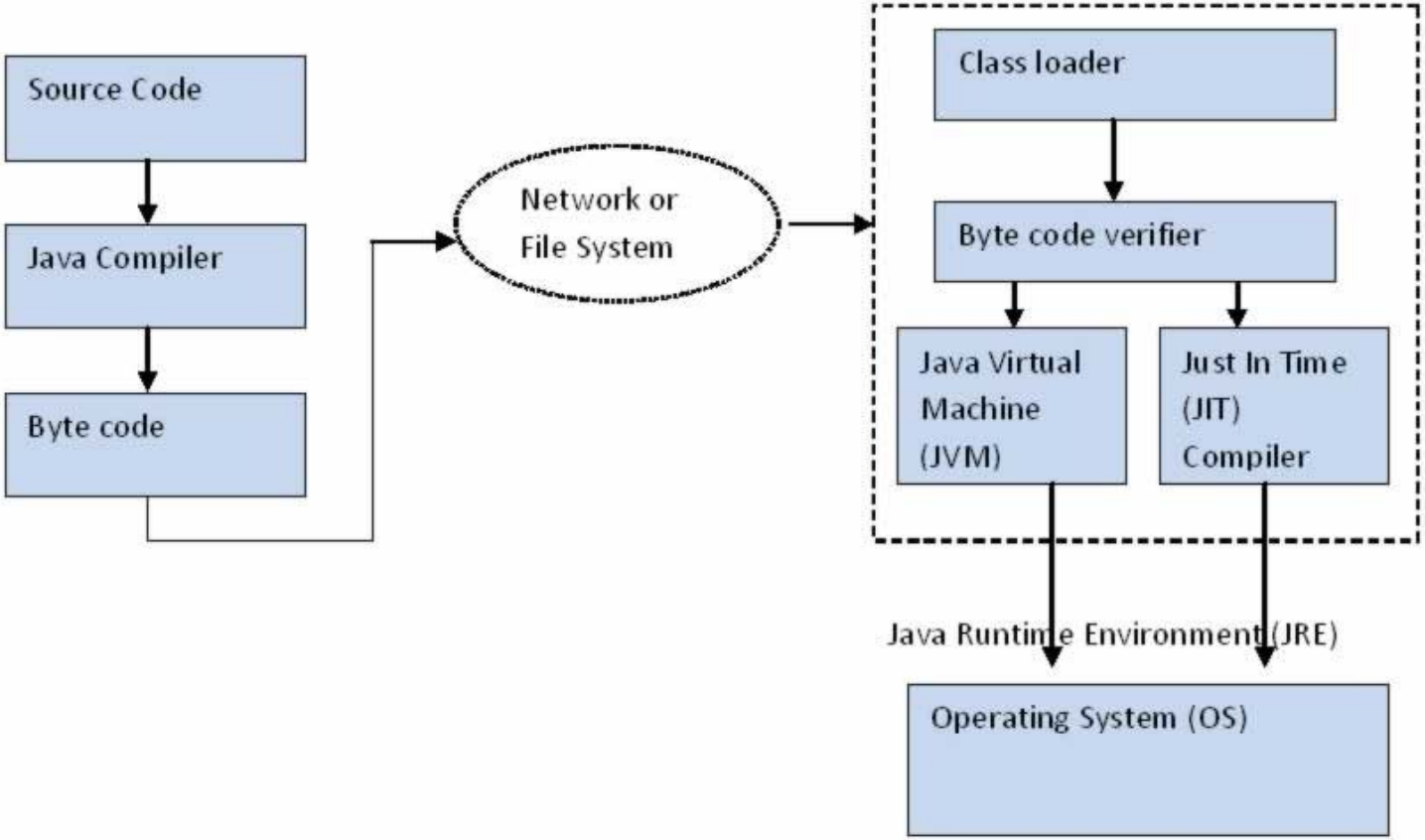
T K Srikanth

Java: Design Goals

“Java technology must enable the development of *secure, high performance,* and highly *robust* applications on *multiple platforms* in *heterogeneous*, *distributed networks*.” And be developer friendly

1. Simple and familiar. Leverage existing languages
2. Object-oriented. Standard set of APIs for basic and advanced capabilities
3. Robust and secure. Avoid ability to directly manipulate hardware/memory, strong type checking, make it difficult for other programs to impact a Java program (and vice versa)
4. Architecture neutral and portable: work on multiple hardware platforms, architectures and OS. “Write once, run anywhere”
5. High performance. Should not have performance challenges
6. Interpreted and dynamic. Ability to not have to go through compiling, global linking etc. Can ease the prototyping phase - quick edit/compile/link/run
7. Multi-threaded. Be able to multi-task and do multiple things at the same time.
8. Distributed computing. Run across machines, including across networks

Byte code and compilation process



Version	Release date	End of Public Updates[4]
JDK Beta	1995	?
JDK 1.0	1996	?
JDK 1.1	1997	?
J2SE 1.2	1998	?
Java SE 8 (LTS)	2014	January 2019 (commercial) December 2020 (non-commercial)
Java SE 9	2017	March 2018
Java SE 10 (18.3)	2018	September 2018
Java SE 11 (18.9 LTS)	2018	N/A

Source: Wikipedia

Hello World!

```
public class HelloWorld{
	public static void main(String[] args) {
		System.out.println(“Greetings!”);
	}
}
```

File: HelloWorld.java

Only one public class per file

Can we have main in multiple classes/files? Why would we do that?

Compilation and running

> javac HelloWorld.java

produces .class file(s)

> java HelloWorld

Runs the :main” in this class

Automatic compilation of referenced classes

Logical collection of classes/libraries can be packed into “jar” files

Java Class Library

- Provide the programmer with a well-known set of functions to perform common tasks, such as maintaining lists of items or performing complex string parsing.
- Provide an abstract interface to tasks that would normally depend heavily on the hardware and operating system.
- Some underlying platforms may not support all of the features a Java application expects. In these cases, the class libraries can either emulate those features using whatever is available, or provide a consistent way to check for the presence of a specific feature.

Types in Java

- reference
- primitives
- Array
- (a special type) void

Primitive types

Fixed size of basic types

boolean true, false (not equivalent to 0 or 1)

char 16 bits (unicode)

byte 8 bits
short 16 bits
int 32 bits
long 64 bits

float 32 bits
double 64 bits

all numeric types are “signed”. No “unsigned”

Objects and references

Everything is an “object” - an instance of a “class”

You manipulate objects through their references

Objects (state) may change, but references do not

Methods are passed references to objects (or primitives), so always “call by value”

Objects are allocated on the heap, references are on the program stack or heap (depending on whether they are local variables or members of objects)

Memory of objects are “garbage collected” when no longer referenced (How?)

String and Array

Both are classes

Specific methods, constructors, operations

String: immutable in size and content - can only be queried

Operators such as concatenation (+) defined.

Array:

Size defined at construct time

Can be queried for length (arr.length)