ESS 201: Programming II
Java

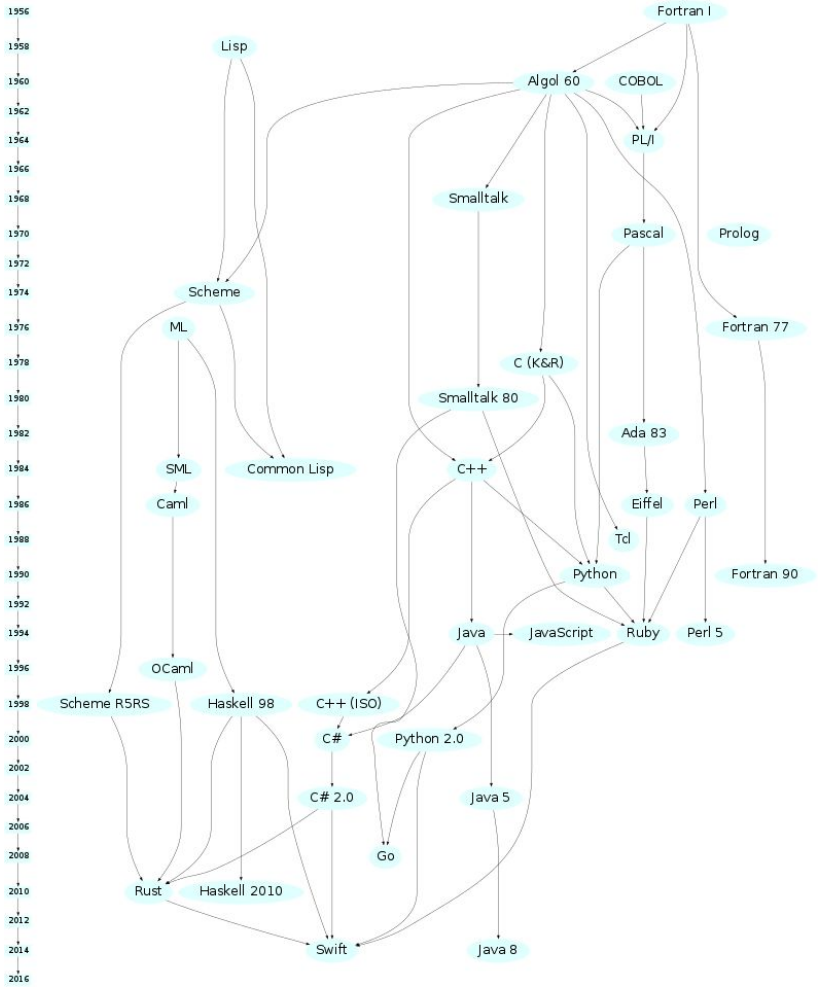
Term 1, 2020-21

Language features - Introduction

T K Srikanth
IIIT-B

History of languages

Indicative - not necessarily complete



rigaux.org/language-study/diagram.html

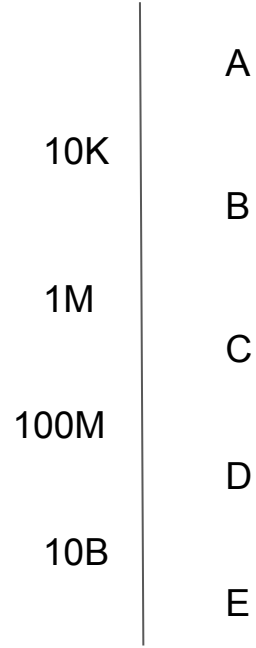
Code size trends

How big (lines of code) is a typical program?

Code size trends

How big (lines of code) is a typical program?
In a large system?

Linux Kernel
A typical phone app
Android
Facebook
A commercial aircraft
A high end car
Google



https://informationisbeautiful.net/visualizations/million-lines-of-code/

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

Version	Release date	End of Free Public Updates[6][7]	Extended Support Until
JDK Beta	1995	?	?
JDK 1.0	January 1996	?	?
JDK 1.1	February 1997	?	?
J2SE 1.2	December 1998	?	?
J2SE 1.3	May 2000	?	?
J2SE 1.4	February 2002	October 2008	February 2013
J2SE 5.0	September 2004	November 2009	April 2015
Java SE 6	December 2006	April 2013	December 2018
Java SE 7	July 2011	April 2015	July 2022
Java SE 8 (LTS)	March 2014	**January 2019 for Oracle (commercial)** December 2020 for Oracle (personal use) At least May 2026 for AdoptOpenJDK At least June 2023[8] for Amazon Corretto	December 2030
Java SE 9	September 2017	March 2018 for OpenJDK	N/A
Java SE 10	March 2018	September 2018 for OpenJDK	N/A
Java SE 11 (LTS)	September 2018	At least August 2024[8] for Amazon Corretto October 2024 for AdoptOpenJDK	September 2026
Java SE 12	March 2019	September 2019 for OpenJDK	N/A
Java SE 13	September 2019	March 2020 for OpenJDK	N/A
Java SE 14	March 2020	September 2020 for OpenJDK	N/A
Java SE 15	September 2020	March 2021 for OpenJDK	N/A
Java SE 16	March 2021	September 2021 for OpenJDK	N/A
Java SE 17 (LTS)	September 2021	TBA	TBA

Legend: Old version | Older version, still maintained | **Latest version** | Future release

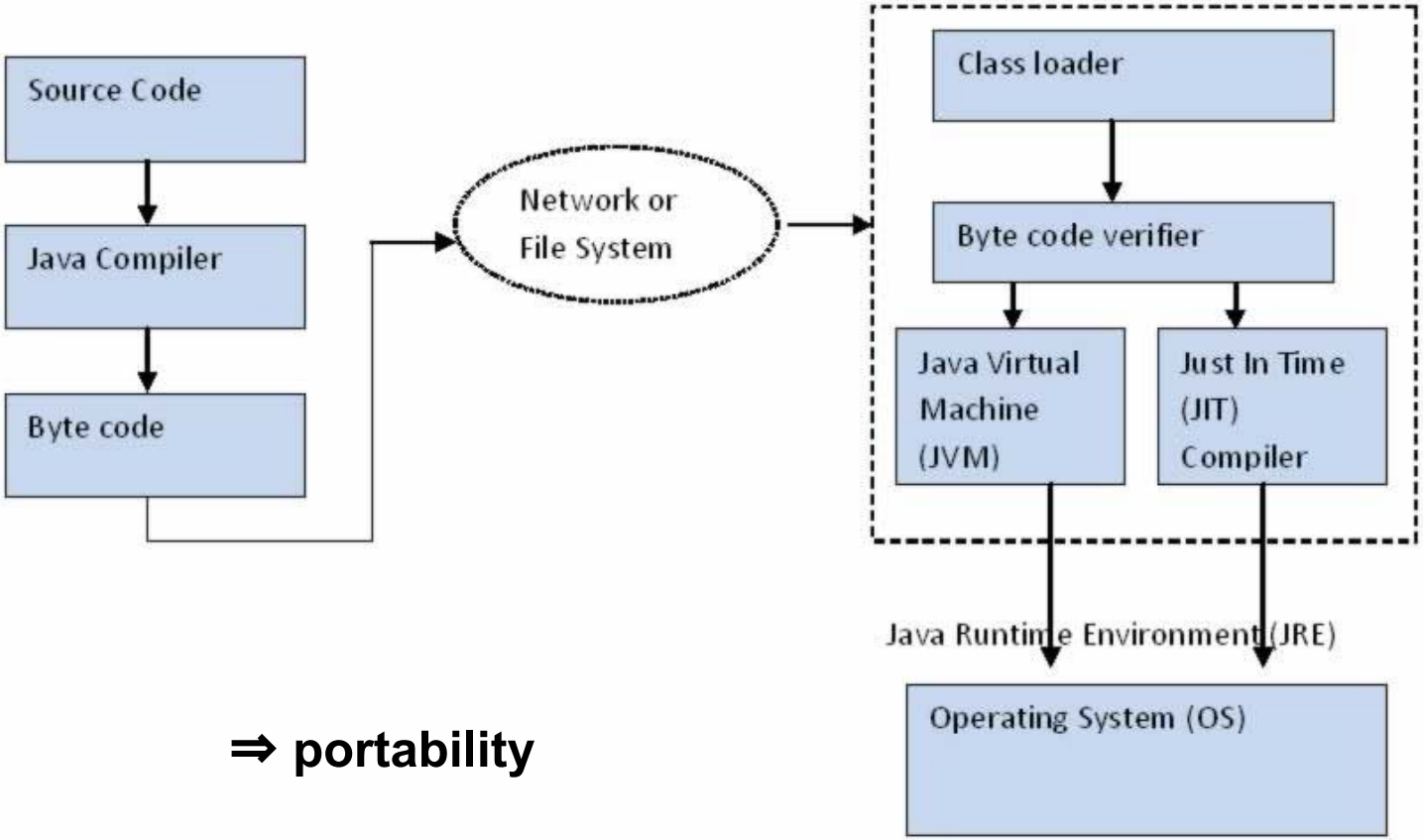
Source: Wikipedia

Originally designed for appliances like set top boxes (1990), then targeted to web browsers (1994), and then to general applications across a range of devices

Some useful features of Java - as seen in Lab 1

- Familiarity of syntax (for C/C++ programmers)
- Data initialization
- Compile-time detection of uninitialized variables
- Out of bounds access
- Strong type checking. (Limited use of casts)
- Memory management and Garbage collection
- Portability - across architectures and OS’s
- “Enforce” Object-oriented design
- Error handling
- …
-

Byte code and compilation process



⇒ portability

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

- primitives
- reference
- 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”
Note: primitives **are NOT** objects

Incidentally, Java does not have a *sizeof* method/operator. Why?

An example of why it is important to understand the size of data types

Or, the perils of “data conversion”: A Bug and a Crash, James Gleick, 1996
https://around.com/ariane.html

“This shutdown occurred 36.7 seconds after launch, when the guidance system's own computer tried to convert one piece of data -- the sideways velocity of the rocket -- from a 64-bit format to a 16-bit format. The number was too big, and an overflow error resulted. ...

… the programmers had decided that this particular velocity figure would never be large enough to cause trouble. After all, it never had been before. Unluckily, Ariane 5 was a faster rocket than Ariane 4. “