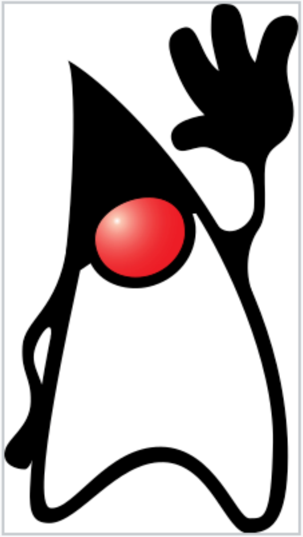
Object Oriented Programming with Java

01 - Introduction



Duke, the Java mascot

Programming Languages

- Machine language
  - "Natural language" of computer component
  - Machine dependent
- Assembly language
  - English-like abbreviations represent computer operations
  - Translator programs (assemblers) convert to machine language
- High-level language
  - Allows for writing more "English-like" instructions
    - Contains commonly used mathematical operations
  - Compiler converts to machine language
- Interpreter
  - Execute high-level language programs without compilation

Compiler - Interpreter

- Compiler
  - Program that converts the instruction sets that are written using a programming language into a form that can be executed by the computer (*called* the machine code -- 1s and 0s)
  - Compiled languages : C++, C#, Java, …
- Interpreter
  - Program that reads the instruction sets and runs without compiling
  - Interpreted languages: PHP, ASP, Javascript,..

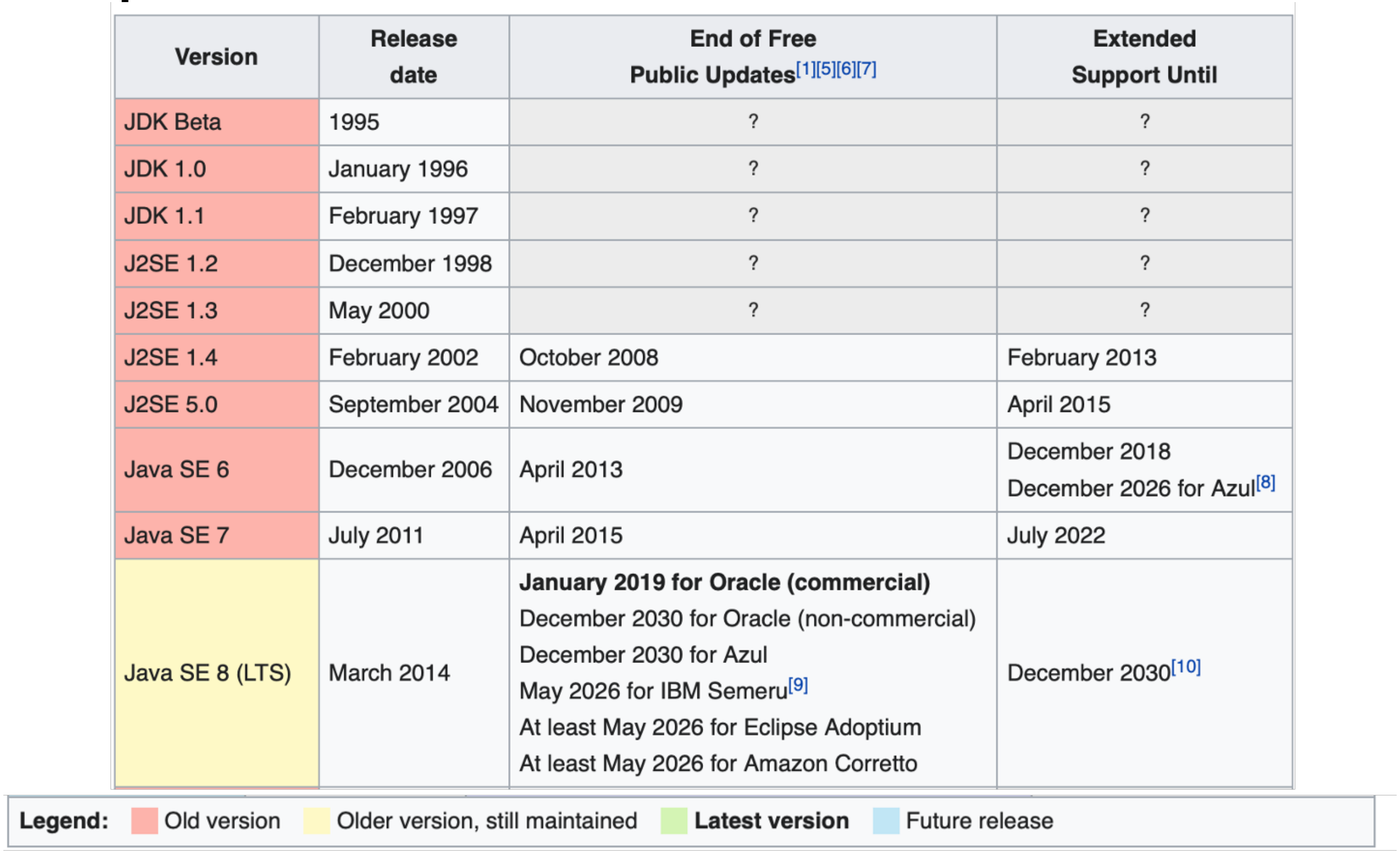
History of Java

- Java
  - Originally for intelligent consumer-electronic devices, especially TVs (The Green Team)
    - By Sun Microsystems in 1991
    - C++ based language created by James Gosling
    - Named “Oak”
  - Then used for creating web pages with dynamic content (Java Applets) - 1995
  - Now also used to:
    - Develop large-scale enterprise applications
    - Enhance web server functionality
    - Provide applications for consumer devices (cell phones, etc.)

Principles / Goals of Java

- It must be "simple, **object-oriented**, and familiar".
- It must be "**robust and secure**".
- It must be "**architecture-neutral** and portable".
- It must execute with "**high performance**".
- It must be "**interpreted**, **threaded**, and **dynamic**”. -> write once, run everywhere!

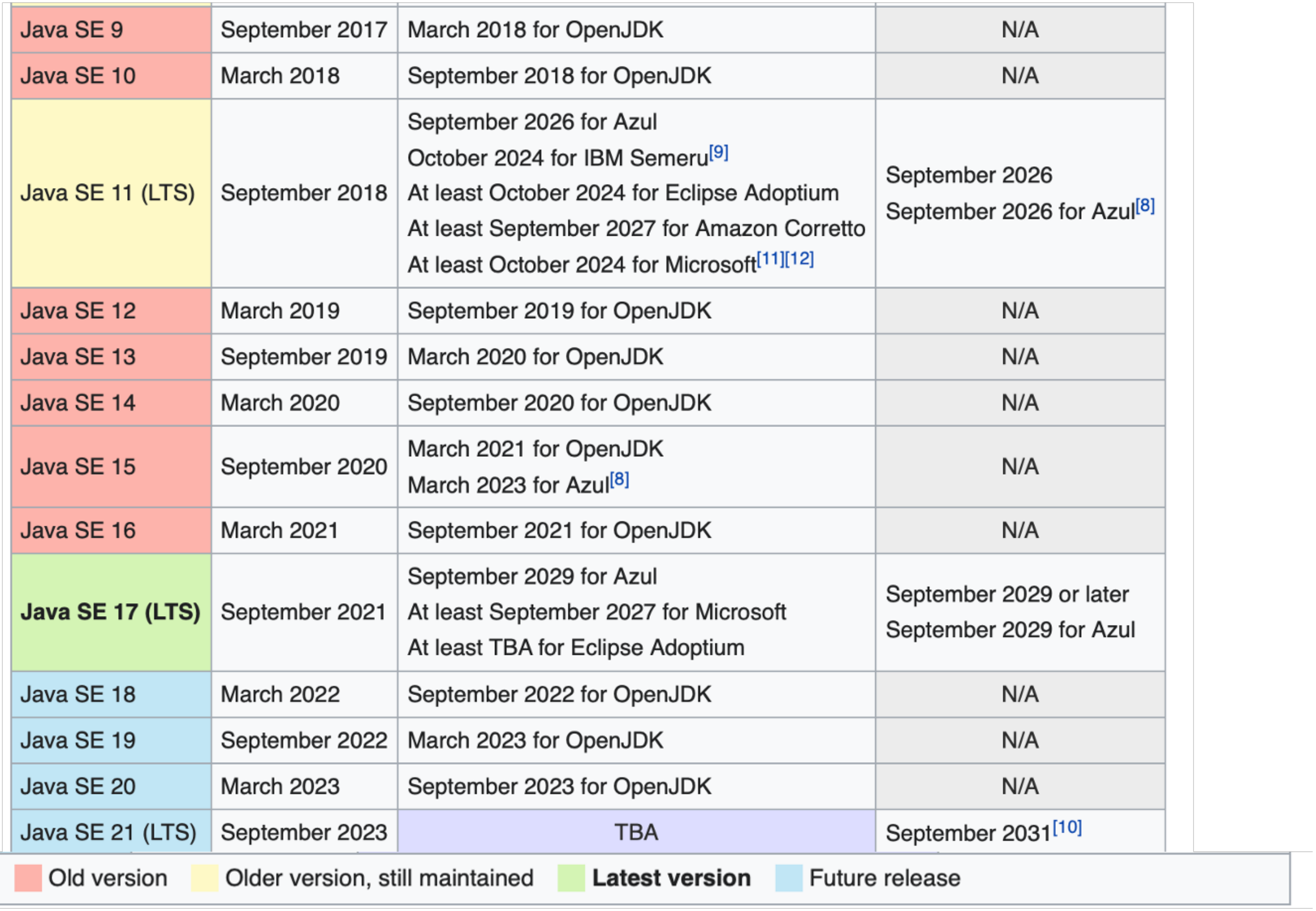
Java Versions



Version	Release date	End of Free Public Updates[1][5][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 December 2026 for Azul[8]
Java SE 7	July 2011	April 2015	July 2022
Java SE 8 (LTS)	March 2014	**January 2019 for Oracle (commercial)** December 2030 for Oracle (non-commercial) December 2030 for Azul May 2026 for IBM Semeru[9] At least May 2026 for Eclipse Adoptium At least May 2026 for Amazon Corretto	December 2030[10]

Legend: 🟥 Old version 🟨 Older version, still maintained 🟩 **Latest version** 🟦 Future release

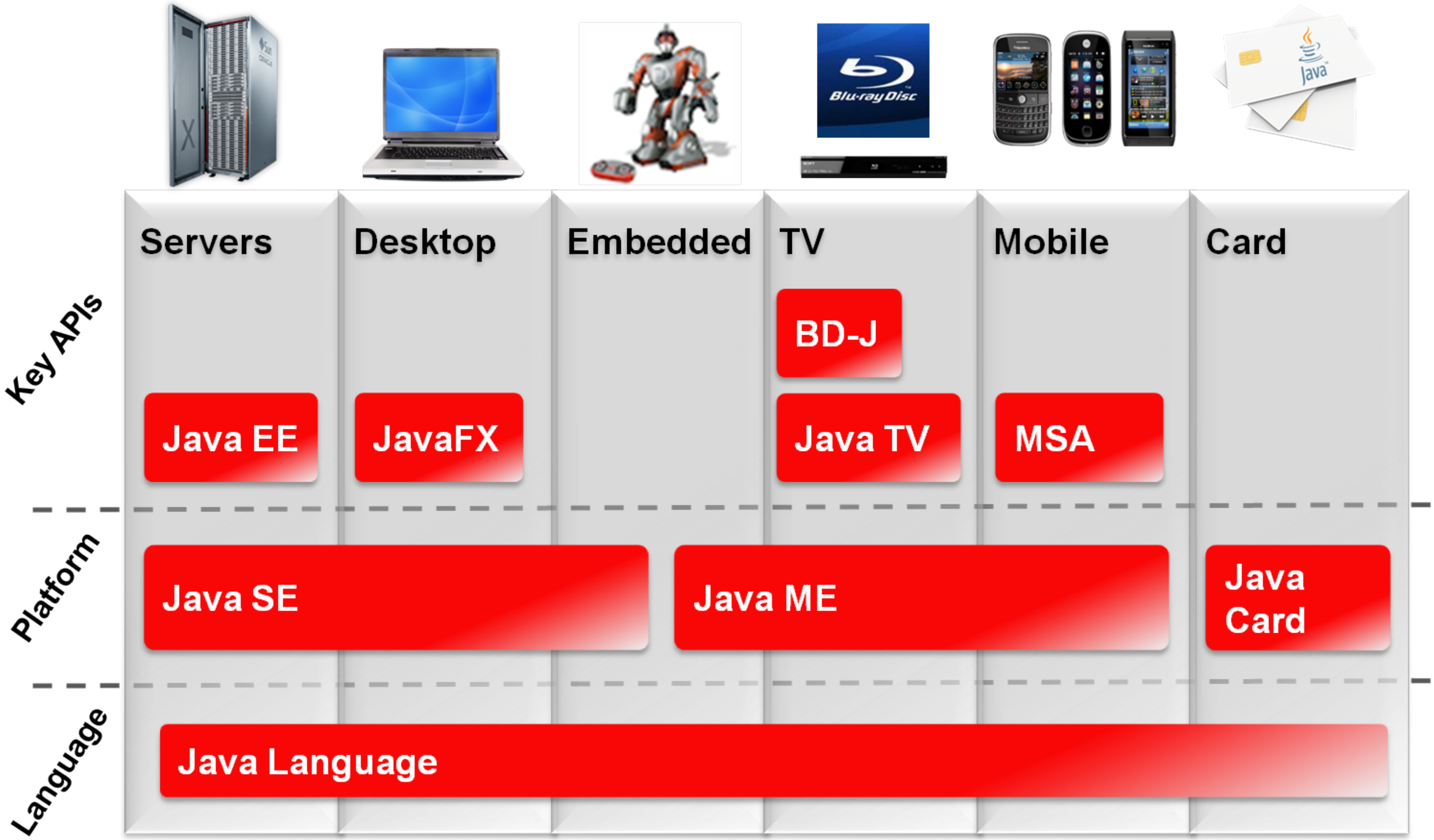
Java Versions



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	September 2026 for Azul October 2024 for IBM Semeru[9] At least October 2024 for Eclipse Adoptium At least September 2027 for Amazon Corretto At least October 2024 for Microsoft[11][12]	September 2026 September 2026 for Azul[8]
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 March 2023 for Azul[8]	N/A
Java SE 16	March 2021	September 2021 for OpenJDK	N/A
Java SE 17 (LTS)	September 2021	September 2029 for Azul At least September 2027 for Microsoft At least TBA for Eclipse Adoptium	September 2029 or later September 2029 for Azul
Java SE 18	March 2022	September 2022 for OpenJDK	N/A
Java SE 19	September 2022	March 2023 for OpenJDK	N/A
Java SE 20	March 2023	September 2023 for OpenJDK	N/A
Java SE 21 (LTS)	September 2023	TBA	September 2031[10]

Legend: 🟥 Old version 🟨 Older version, still maintained 🟩 **Latest version** 🟦 Future release

Java Technology Product Groups



Servers

Desktop

Embedded

TV

Mobile

Card

Key APIs

BD-J

Java EE

JavaFX

Java TV

MSA

Platform

Java SE

Java ME

Java Card

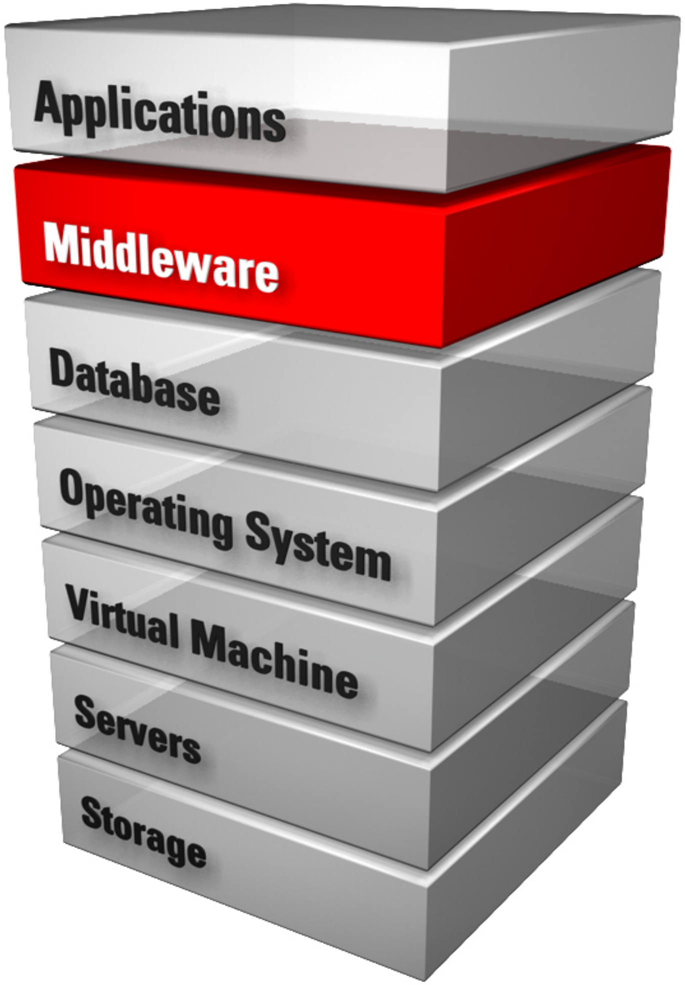
Language

Java Language

Java Technologies

- **Java SE**
  - Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on desktops and servers, as well as today's demanding Embedded and Real-Time environments.
- **Java EE**
  - Java Platform, Enterprise Edition (Java EE) builds on the solid foundation of Java Platform, Standard Edition (Java SE) and is the industry standard for implementing enterprise-class service-oriented architecture (SOA) and next-generation web applications.
- **Java ME**
  - Java Platform, Micro Edition (Java ME) provides a robust, flexible environment for applications running on mobile and other embedded devices

Java in Server Environments

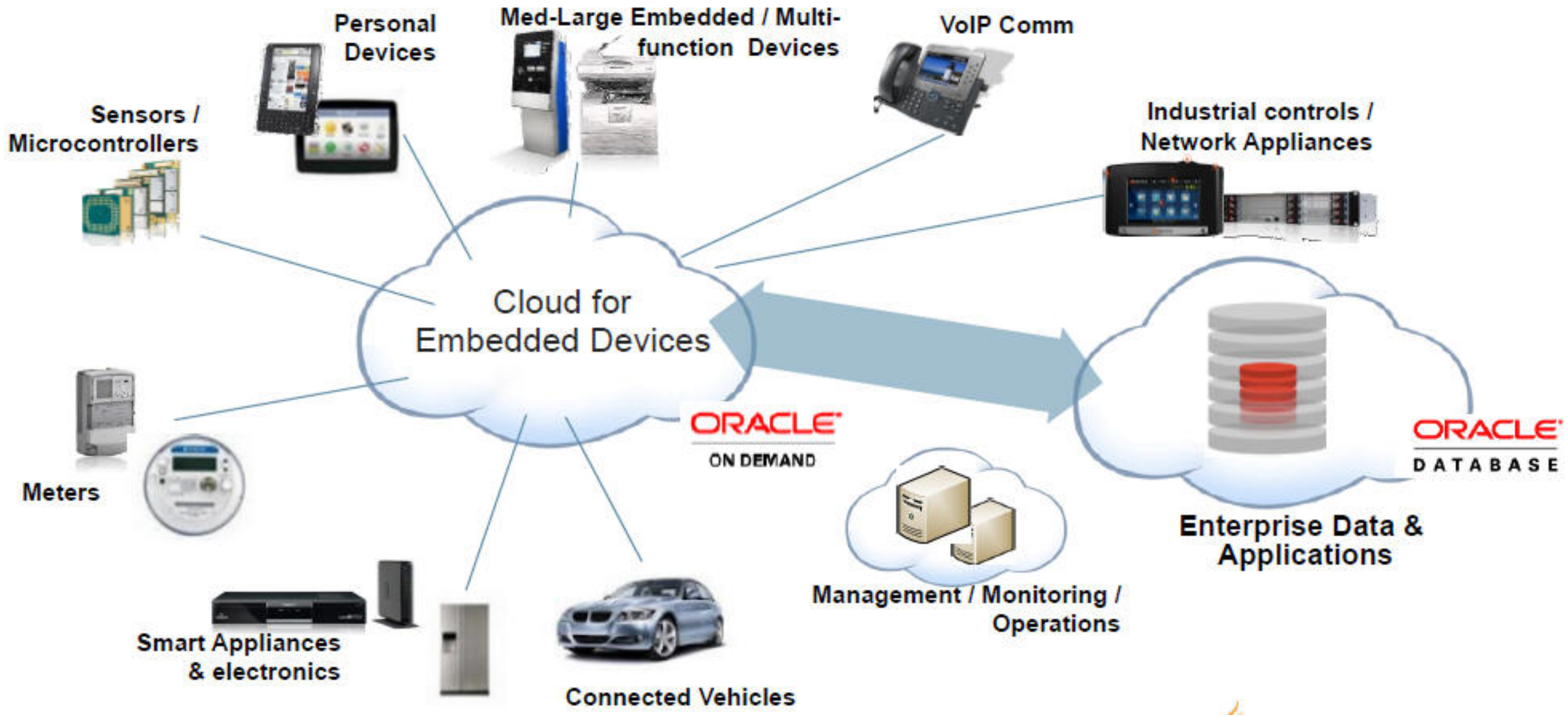


- Java is common in enterprise environments:
  - Oracle Fusion Middleware
    - Java application servers
      - GlassFish
      - WebLogic
  - Database servers
    - MySQL
    - Oracle Database

Copyright 2020 - Altuğ Tanaltay

The Internet of Things

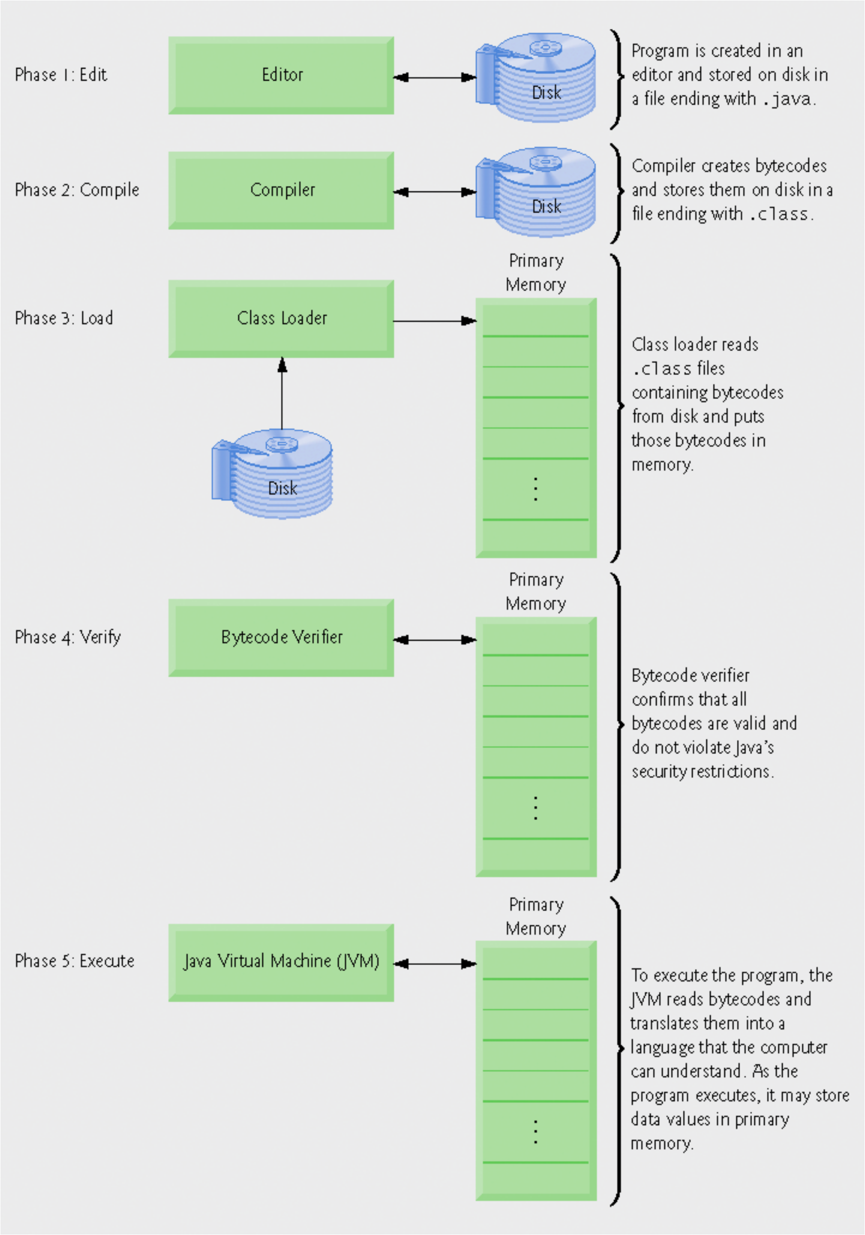
- Devices on the "edge" represent a huge growth opportunity.


Personal Devices
Med-Large Embedded / Multi-function Devices
VoIP Comm
Sensors / Microcontrollers
Industrial controls / Network Appliances
Cloud for Embedded Devices
ORACLE ON DEMAND
ORACLE DATABASE
Meters
Enterprise Data & Applications
Management / Monitoring / Operations
Smart Appliances & electronics
Connected Vehicles

The Java Community

What do we need?

- A text editor (Notepad,Emacs, Vi,…) or an IDE (Eclipse, Netbeans,…)
  - Eclipse: http://www.eclipse.org/downloads/
- Java Development Kit (JDK)
  - Includes developer tools, JVM and standard class libraries derived from Java API (Bundled together as *Java Runtime Environment* - JRE)
  - JDK : http://java.sun.com/
  - JDK>JRE>JVM



Program is created in an editor and stored on disk in a file ending with `.java`.

Compiler creates bytecodes and stores them on disk in a file ending with `.class`.

Class loader reads `.class` files containing bytecodes from disk and puts those bytecodes in memory.

Bytecode verifier confirms that all bytecodes are valid and do not violate Java's security restrictions.

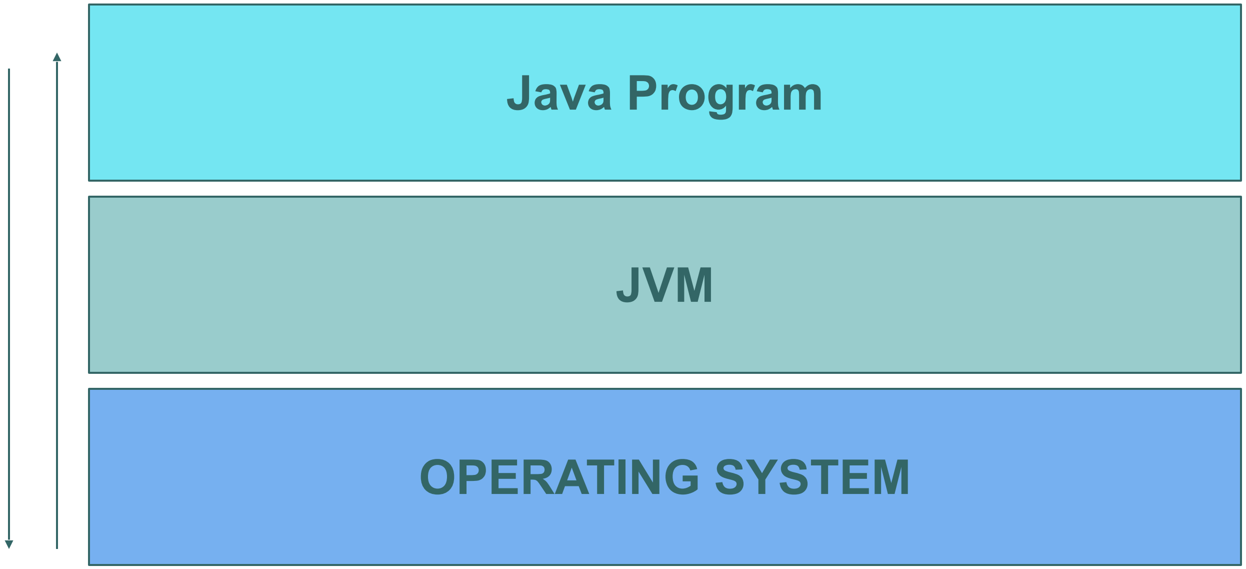
To execute the program, the JVM reads bytecodes and translates them into a language that the computer can understand. As the program executes, it may store data values in primary memory.

Typical Java development environment

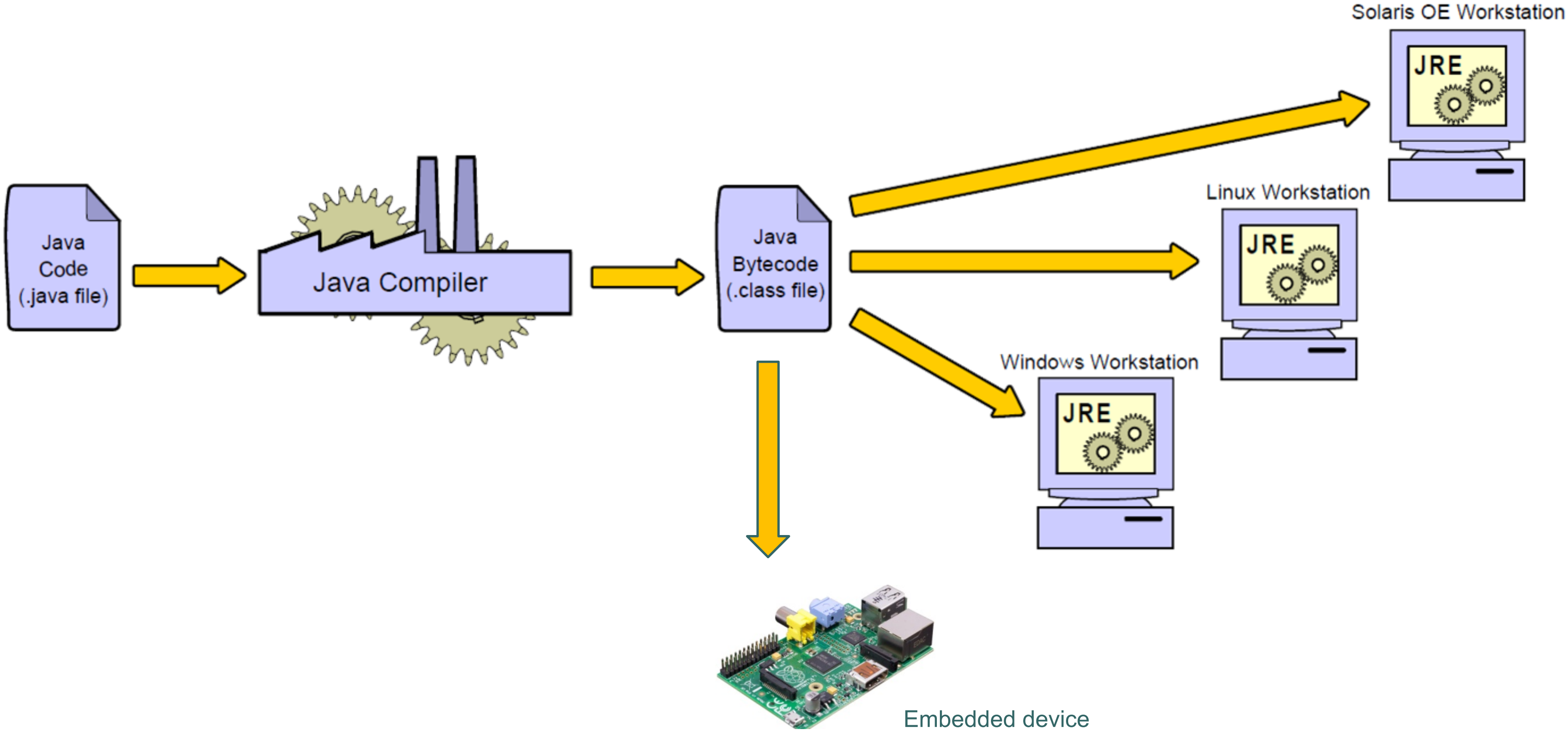
Typical Java Development Environment

- Java programs go through five phases
  - Edit
    - Programmer writes program using an editor; stores program on disk with the `.java` file name extension
  - Compile
    - Use javac (the Java compiler) to create bytecodes from source code program; bytecodes stored in `.class` files
  - Load
    - Class loader reads bytecodes from `.class` files into memory
  - Verify
    - Bytecode verifier examines bytecodes to ensure that they are valid and do not violate security restrictions
  - Execute
    - Java Virtual Machine (JVM) uses a combination of interpretation and just-in-time compilation to translate bytecodes into machine language

Interaction with OS



Java Programs Are Platform-Independent


Java
Code
(.java file)
Java Compiler
Java
Bytecode
(.class file)
Solaris OE Workstation
JRE
Linux Workstation
JRE
Windows Workstation
JRE
Embedded device