UNIVERSITY OF MISKOLC
FACULTY OF MECHANICAL ENGINEERING
AND INFORMATICS

Java programming

GEIAL31A-B2a

Basic Concept of Java

Tamás Tompa, PhD
assistant professor
Department of Information Technology

Based on materials prepared by Miklós Szűcs

Miskolc, 2026

Properties of the Java language

- **Object-oriented programming language**
  - **implements the OOP principles**
  - the parts that cause complications are omitted,
    - e.g. multiple inheritance (more difficult implementation, code reuse)
    - operator overloading (ambiguous usage, less readable code)
  - strictly OOP (no out-of-class declarations, statements)
- Similarity to C++
  - The syntax is similary to C++
  - No pointers
- Rich Class Library (217 packages)
  - Approx. 5000 classes and interfaces in JDK 9

Versions of the Java language

- 1990-1992: **SUN's internal project** led by James Gosling (+Patrick Naughton, Mike Sheridan, ...)
  - **Programming language for Set Top Boxes**
  - Demand: platform-independent technology – **do not depend on the hardware of the box for programming**
- None of the existing languages were suitable, so a new one was developed
- The project (and with it the new language) slowly died

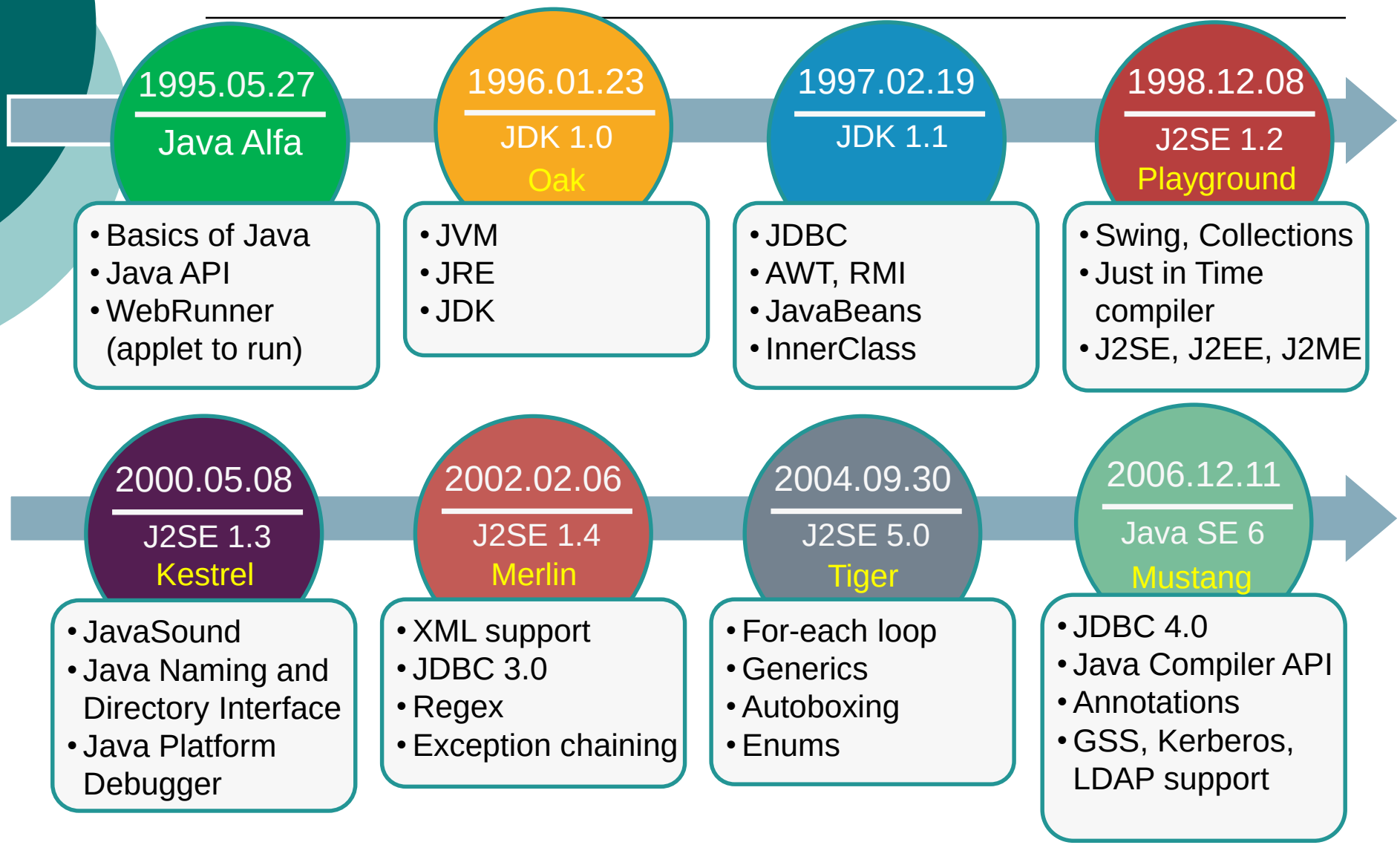


Versions of the Java language

- 1995: The **rapid spread of the Internet** raised the **need** for **platform-independent technology** again
- The **project was revived**, but with a more general objective
- The result was a programming language called **Oak**, **which** was the basis and **first version of the Java** language
- **Since the name Oak was already existing, this programming language was renamed to Java** and the development environment was renamed JDK (Java Development Kit).
  - The **first coffee called Java coffee was made on the island of Java**, the first platform-independent programming language was named Java, and the symbol of the language was a cup of coffee

The development team of the SUN Green project in 1997

Versions of the Java language



Tux and Duke memorate about the Sun

Versions of the Java language

- 4/20/2009: Oracle buys the Sun Microsystems
- 10 new versions are developed by Oracle

2011.07.28 — Java SE 7 Dolphin

- String in Switch
- New exception handling
- java.nio package

2014.03.18 — Java SE 8 Spider

- Lambda Expressions
- Pipelines and Streams
- Date and Time API

2017.09.27 — JDK 9 Jigshow

- Modularity
- JVM options
- jShell

2018.03.20 — Java SE 10

- Type conclusion
- Parallel running, garbage picking

Versions of the Java language

2018.09.25 — Java SE 11

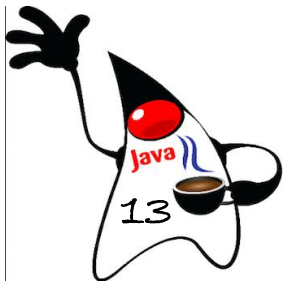
- Not free for business!
- Enterprise platform omitted
- Unicode 10

2019.03.19 — Java SE 12

- Many small innovations

2019.09.17 — Java SE 13

- Many small innovations



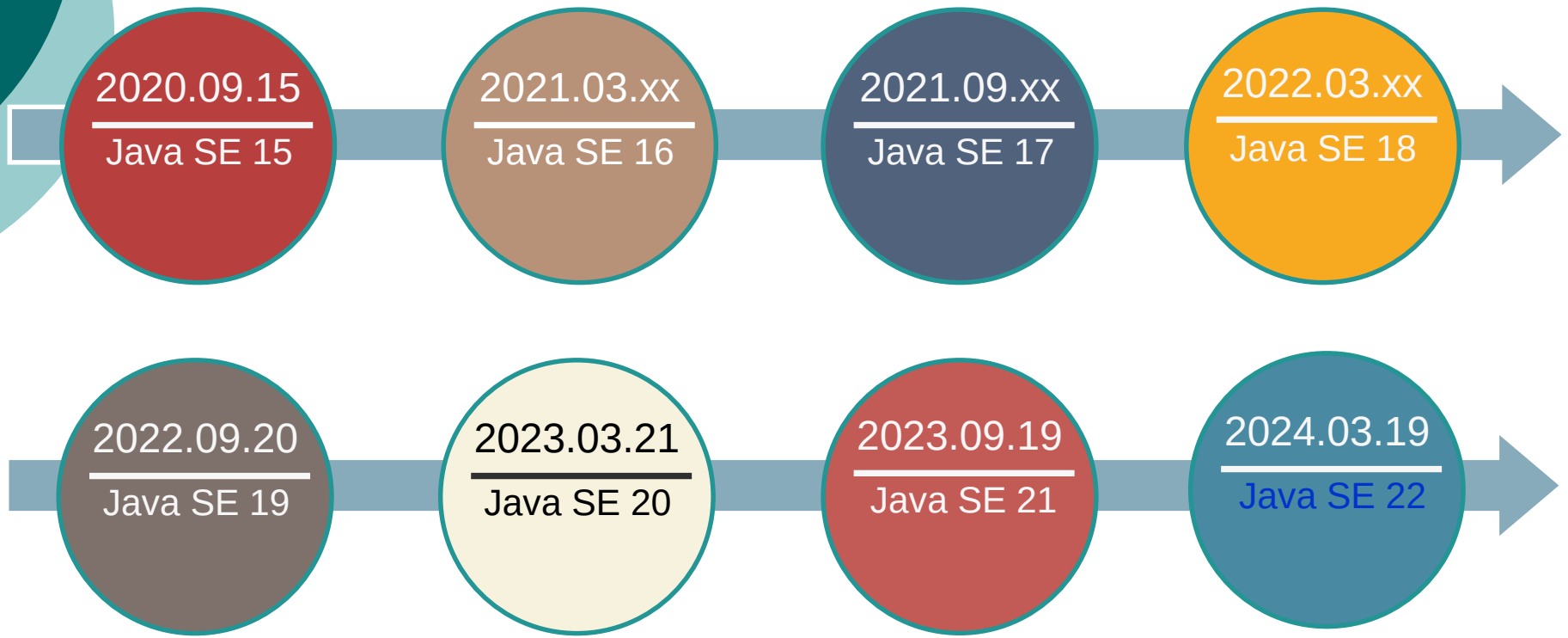
2020.03.17 — Java SE 14

Java 14

From JDK 9:

- Semi-annual release cycle
- Only one version is supported during half a year
- Two updates during its lifecycle

Versions of the Java language

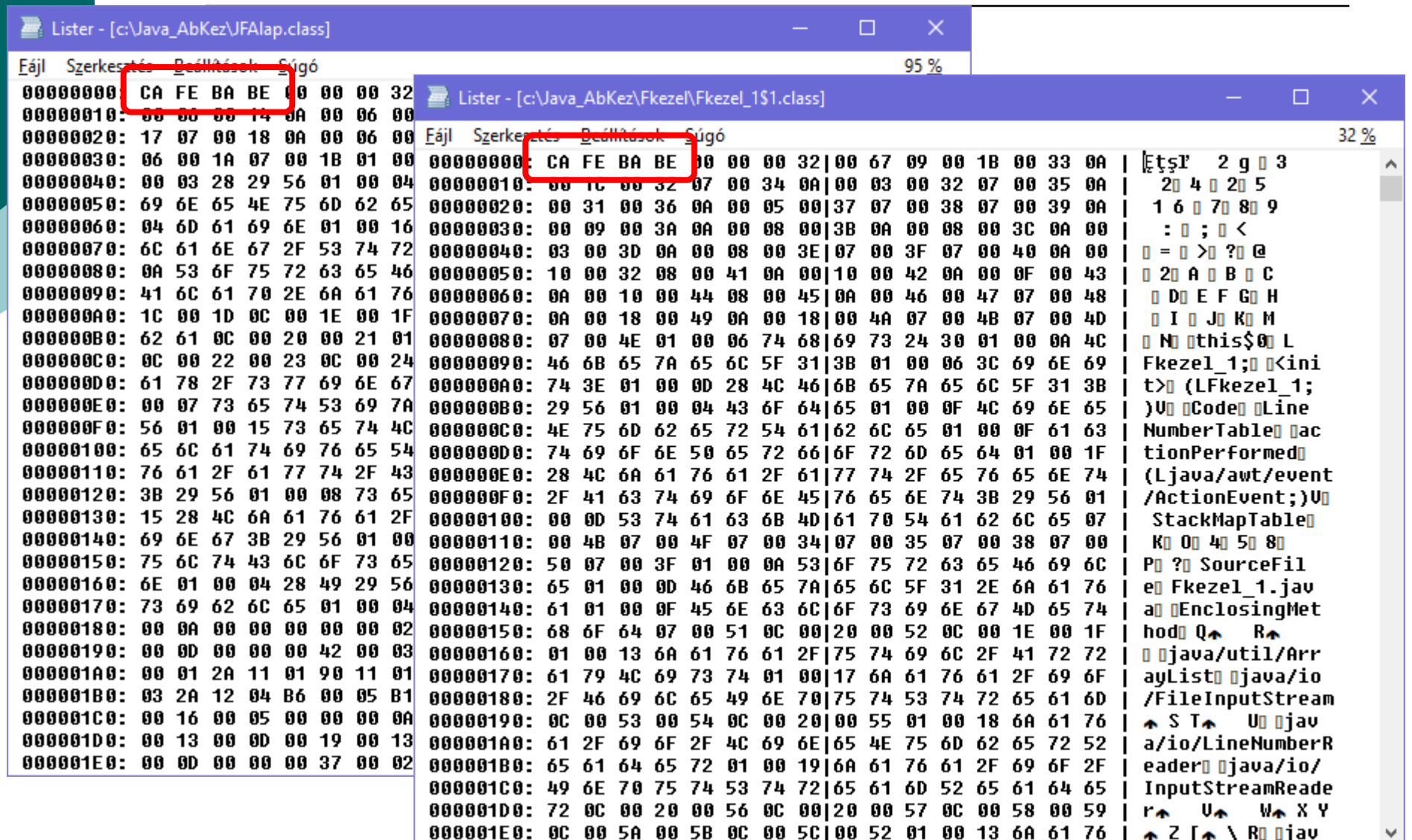


Versions of the Java language

Duke's look follows changes in Java

Coffee baby!

„Cafe babe”: The hex code for each class file starts with it!



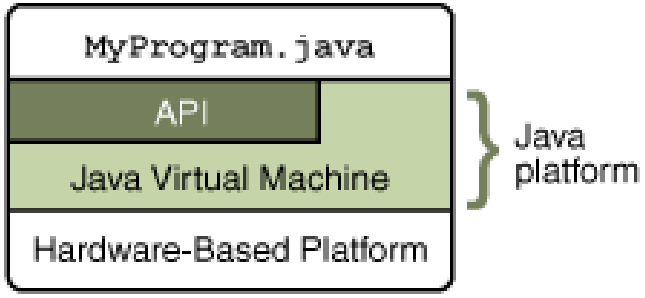
The programmer forgot to update Java
so he went to the coffee shop to get a fresh one.

Properties of the Java language

- **High-level programming language**
  - platform-independent
  - general purpose
  - fully object-oriented
  - simple
  - interpreted (but in a special way)
  - distributed
  - robust
  - safe
  - portable
  - multithreaded
  - It was born with knowledge of C++ and considered its formalism as a model (born based on C++)
  - It tried to avoid the disadvantages of C++

Properties of the Java language

- **High-level programming language**
  - **platform-independent**
    - **a platform is a hardware or software environment where programs run**
    - most platforms mean hardware and operating system together
    - **the Java platform differs from other platforms** that it is completely a software platform and is based on other hardware-based platforms
    - **the Java platform consists of two components:**
      - Java API
      - Java VM



Properties of the Java language

- **High-level programming language**
  - **platform-independent**
    - The Java platform consists of two components:
      - **Java API**: ready-to-use software components: **classes and interfaces organized into packages**
      - **Java VM**:
        - **a program that runs the generated platform-independent bytecode** on the given hardware
        - **Java bytecode is converted into machine code by the Java VM** (every Java interpreter, development environment and applet running in a browser, includes a Java VM)

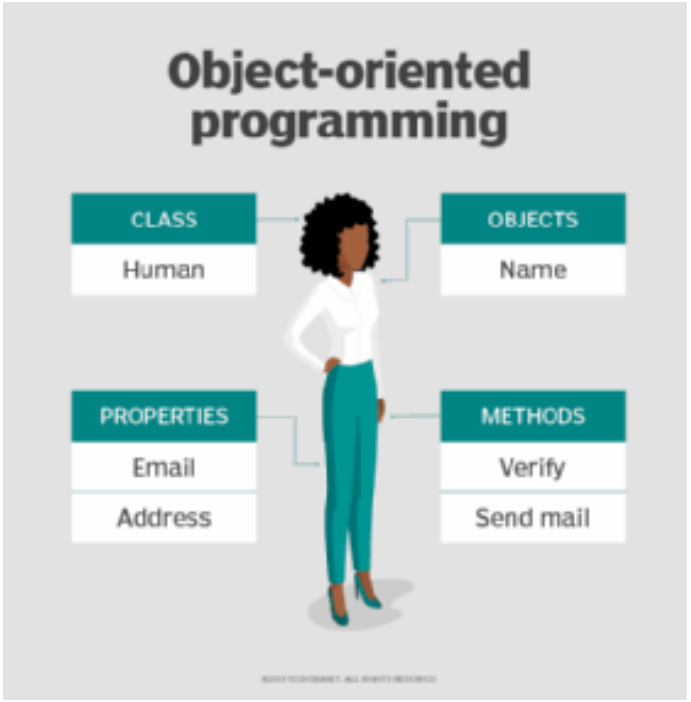
Properties of the Java language

- **High-level programming language**
  - **general purpose**
    - a programming language **that can be widely used to write softwares**
    - **can be create programs on any topic**
    - Java supports:
      - the use of computer networks
      - database management
      - international programming
    - Java is suitable for:
      - desktop applications, applets (programs that can be run in web browsers),servlets (server-side programs which generating dynamic content)

Properties of the Java language

- **High-level programming language**
  - **Fully object-oriented**
    - It implements all OOP principles
      - **1. Encapsulation**
      - **2. Inheritance**
      - **3. Polymorphism**

Polimorphism
Inheritance
Encapsulation

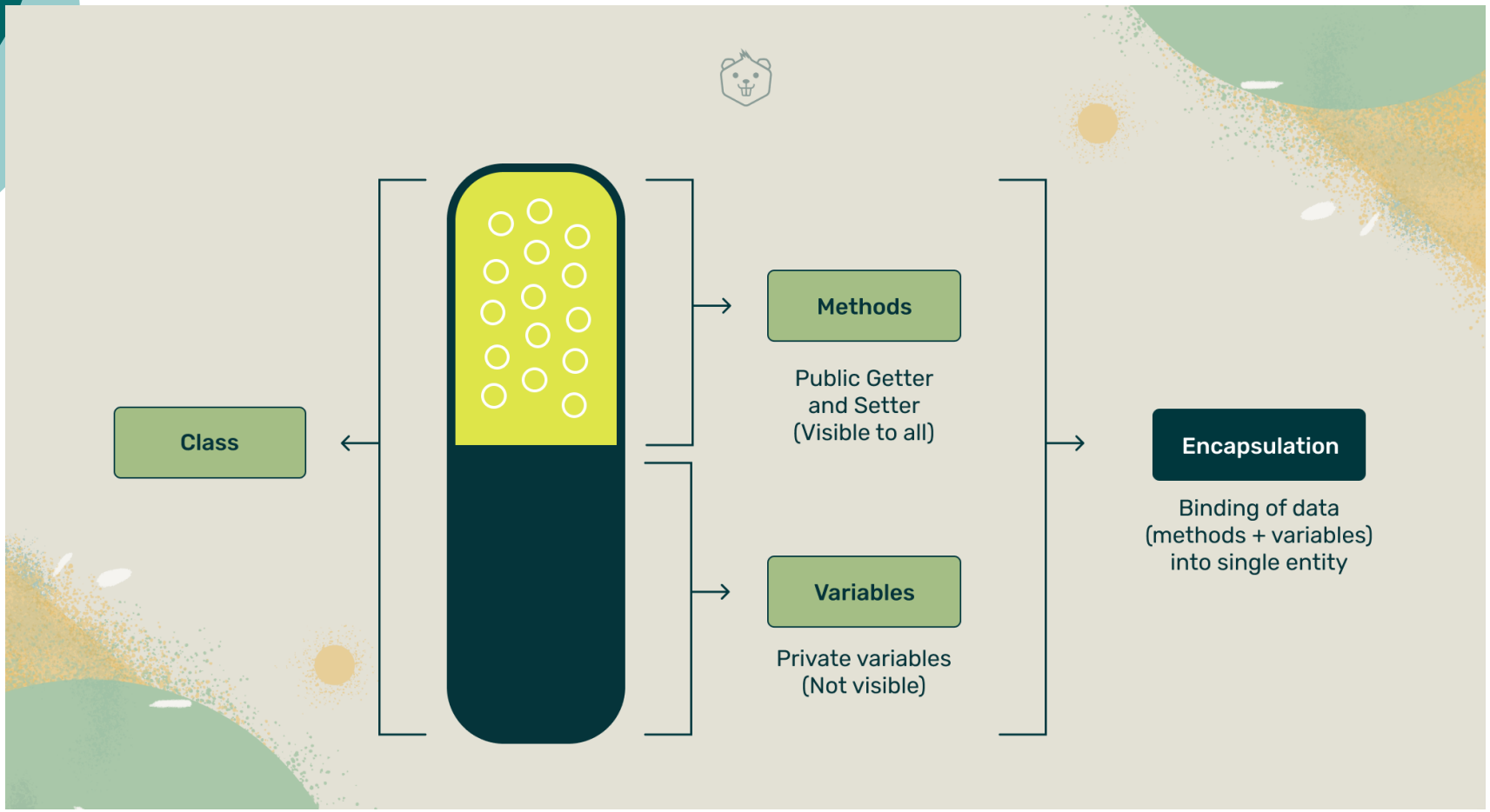


Properties of the Java language

- **High-level programming language**
  - **Fully object-oriented**
    - It implements all OOP principles
      - **1. Encapsulation**
        - **Data and related activities are a single unit**
        - Basic unit: class with the following building blocks:
          - fileds / member variables, methods
        - **The unit hides information**
          - **the data of an object is inaccessible to the outside world**
          - an object can only communicate with the outside world through its interface. (Interface: a set of methods available to the outside world.) The implementation of methods is hidden.

Properties of the Java language

Encapsulation

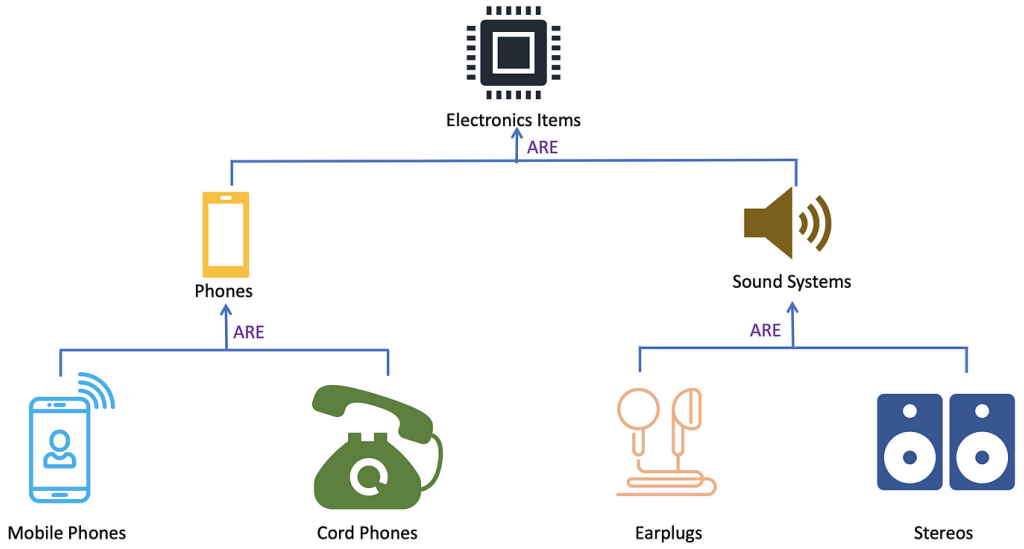


Properties of the Java language

Data hiding

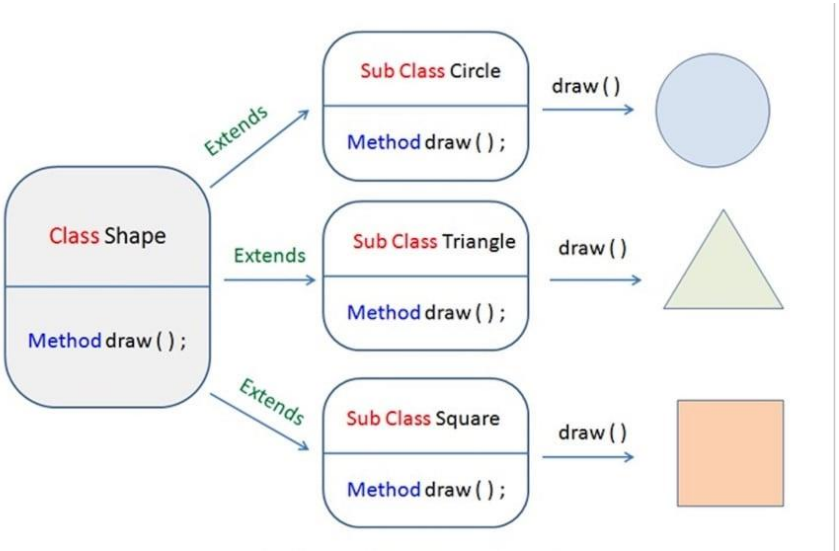
Properties of the Java language

- **High-level programming language**
  - **Fully object-oriented**
    - It implements all OOP principles
      - **2. Inheritance**
        - **classes can be created by extending existing classes by adding additional fields or methods**



Properties of the Java language

- **High-level programming language**
  - **Fully object-oriented**
    - It implements all OOP principles
      - **3. Polymorphism**
        - **derived classes can have methods with the same name but different content than in ancestor class =>** transparent code, don't need to modify the nameings



Properties of the Java language

- **High-level programming language**
  - **simple**
    - Java inherits its syntax from C and C++
      - but has a much simpler object model than C++
    - **the Java language is easy to learn**
      - **It does not mean that writing good Java codes is an easy task!**
    - the standard library set for the programming environment is very large!
      - this makes programming more efficient
      - don't need to write a lot of code
      - but it makes it difficult to learn the program

Properties of the Java language

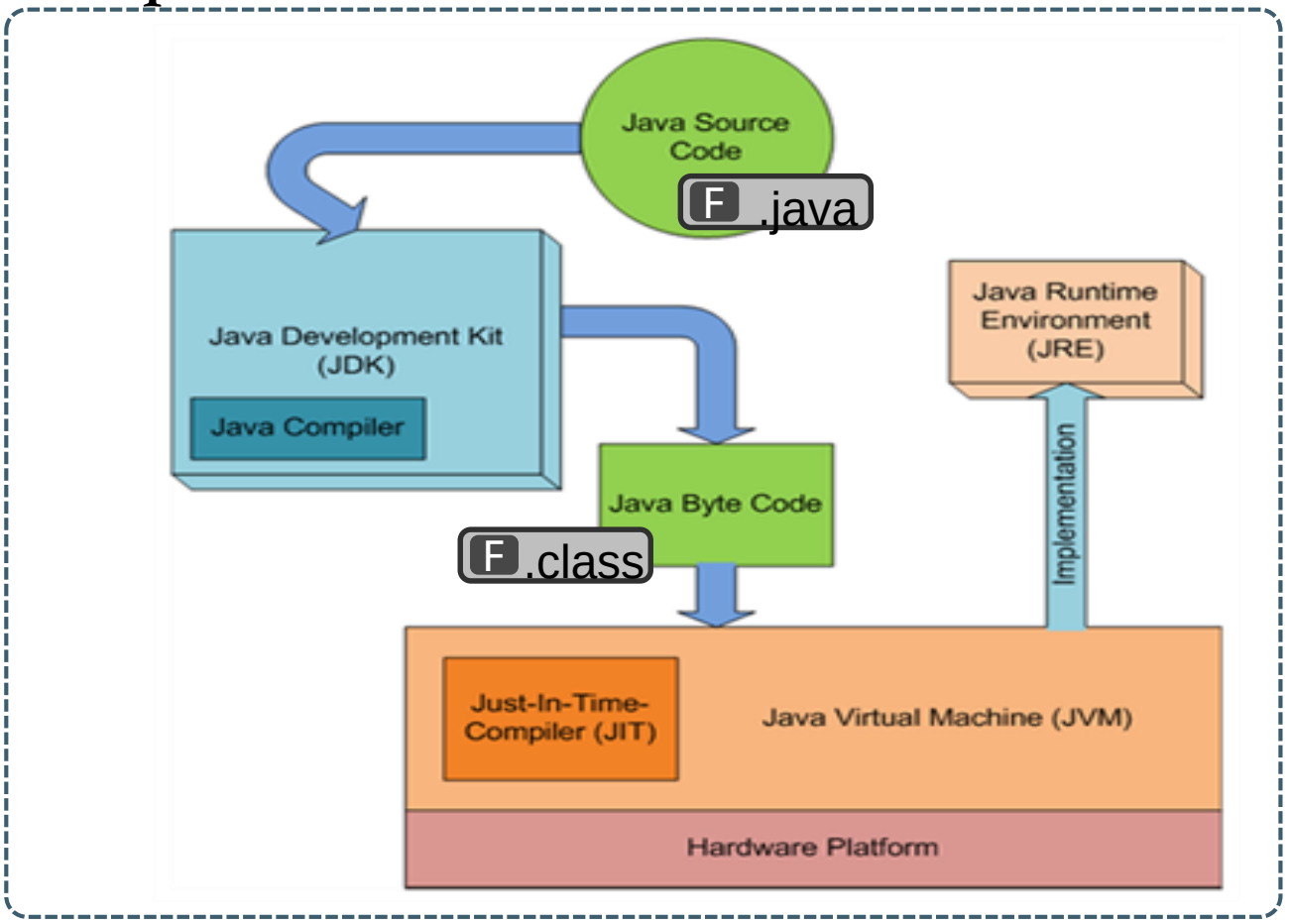
- **High-level programming language**
  - **interpreted**
    - **Java is both compiled and interpreted**
    - in case of most programming languages, perform compilation or interpretation before the program runs
    - Java uses a strange mixture of the two
    - the source program (`myProgram.java`) is compiled into an intermediate language by the compiler (javac.exe) to create Java bytecode (`myProgram.class`), and this platform-independent code is interpreted and run by the Java VM (java.exe)

Properties of the Java language

- **High-level programming language**
  - **interpreted**
    - **the compilation is exectued once, and the interpretation is performed each time the program is executed**
    - **Java bytecode is converted to machine code by the Java VM**
    - **the bytecode file is portable**
    - can also be used in a web environment
    - the interpreter program can be embedded in other programs or devices (browser, phone)
    - slow compared to compiler languages!

Properties of the Java language

- **High-level programming language**
  - **interpreted**



Properties of the Java language

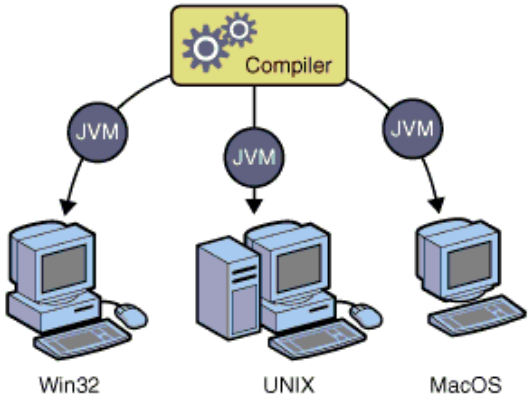
- **High-level programming language**
  - **distributed**
    - Java is suitable for building distributed systems
    - a distributed system is a computer system consisting of autonomous operation units connected via a communication channel, which is capable of solving IT tasks together by coordinating the components
    - **remote Method Invocation (RMI) is a tool in Java that allows to call methods of objects located in other VMs**
    - the RMI system was designed that the **behavior of distributed objects on the network are similar to the behavior of local objects**

Properties of the Java language

- **High-level programming language**
  - **robust**
    - a language is robust if it **prevents or filters out programming errors during runtime**
    - robustness at the linguistic level is manifested in strict, static typing
    - **all data has a well-defined type**, there are no automatic conversions, explicit conversion possibles in case of compatible types, otherwise it causes an exception during runtime
    - **dynamic garbage collection avoids the memory leak**

Properties of the Java language

- **High-level programming language**
  - **portable**
    - in order to the compiled program to run unchanged on different hardware architectures, the compiler compiles the program not into machine code of a specific processor, but into the instruction system of an imaginary hardware (VM)
    - the virtual machine is written in C
    - **to run Java programs on a new architecture, if implemented the virtual machine, including the system libraries**



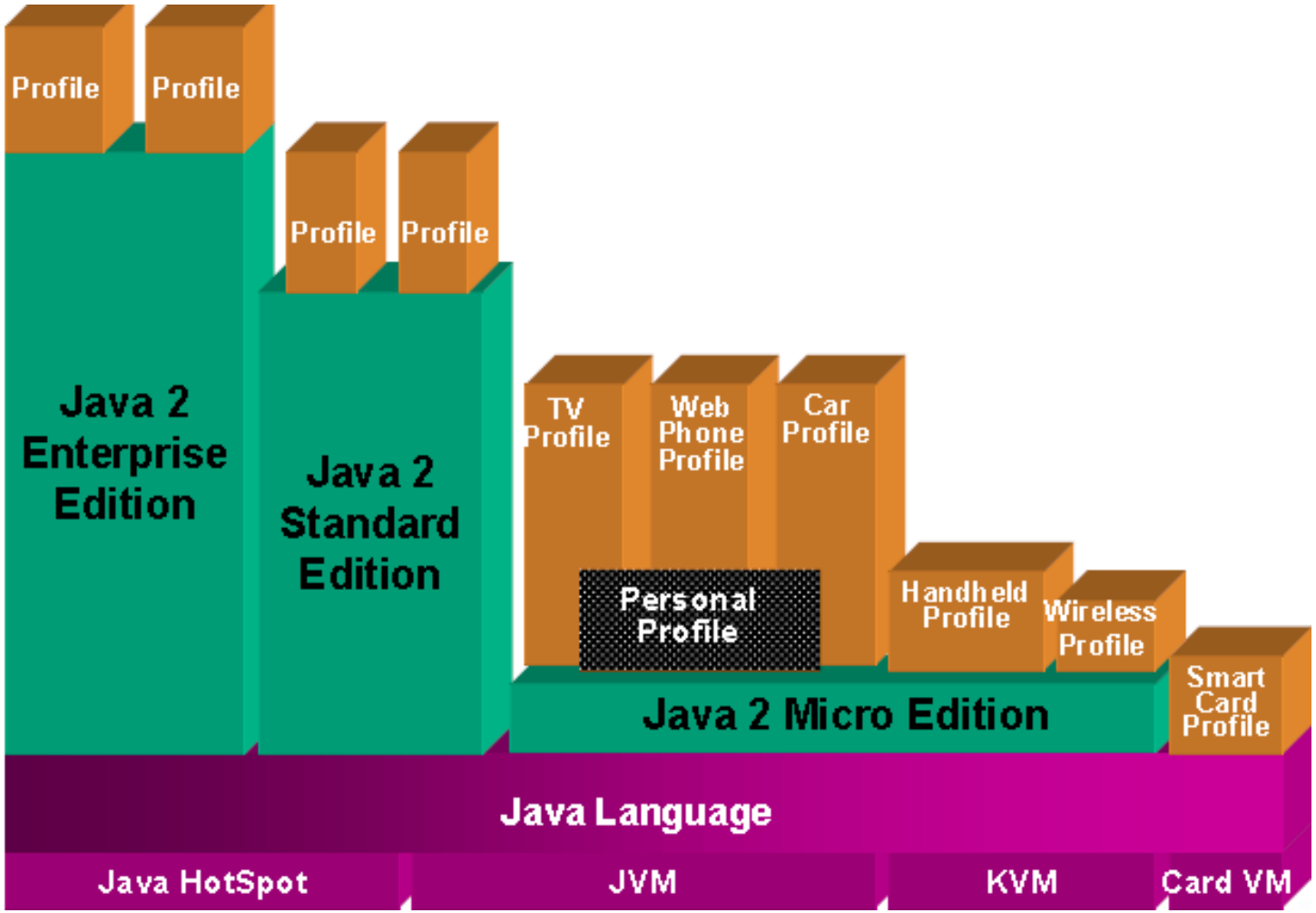
Properties of the Java language

- **High-level programming language**
  - **multithreaded**
    - a significant **part of the programs can be divide into parts that can be executed in parallel**
    - **a task that communicates with each other, divided into relatively loosely connected threads**
    - for multithreaded programming, **Java provides automatic mutual exclusion (mutex)** at the language level: for creating synchronized methods and instructions, as well as threads, the system library contains a thread class
    - the **Java virtual machine includes a preemptive scheduler based on priority**

Java development environments

- SUN divides the Java-based standard tools into the following releases:
  - **J2SE - Java 2 Platform, Standard Edition**
    - general development environment, all basic services that allow the creation of Java programs
  - **J2EE - Java 2 Platform, Enterprise Edition**
    - it includes J2SE features and component-oriented development, supports building large applications, building web service-based applications
  - **J2ME - Java 2 Platform, Micro Edition**
    - optimized for low-resource systems (phones, embedded systems)

Java development environments



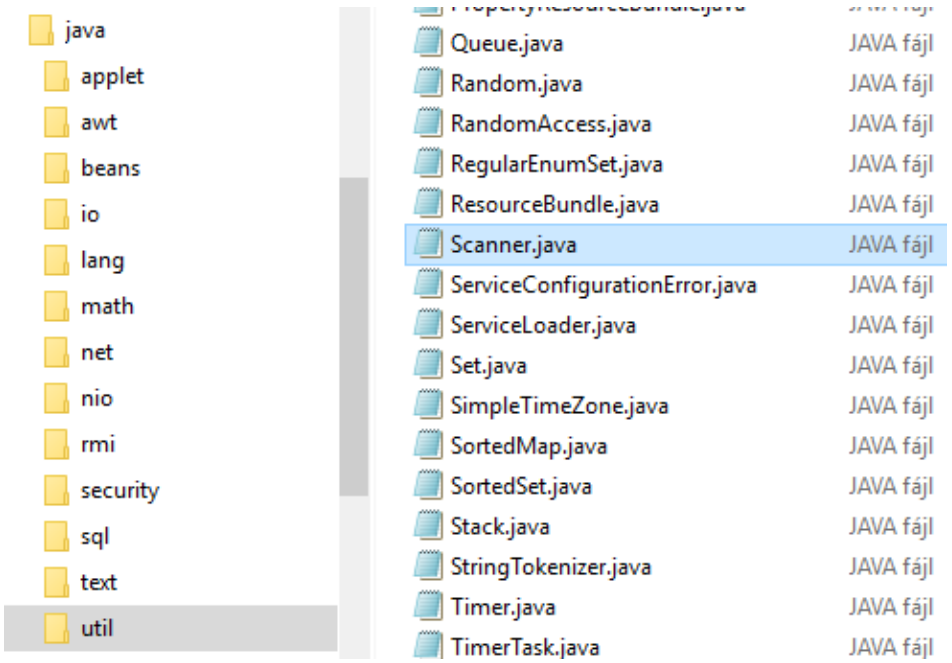
The structure of a Java program

```
package mypackage;

import java.lang.*;


public class Hello {

public static void main(String[ ] args) {
      System.out.println(”Hello world”);
   }
}
```

It must be in a file named Hello.java!

Package

- Where the class is stored in the file system



- Package import
  - the compiler searches for the referenced object in the specified directory

Libraries

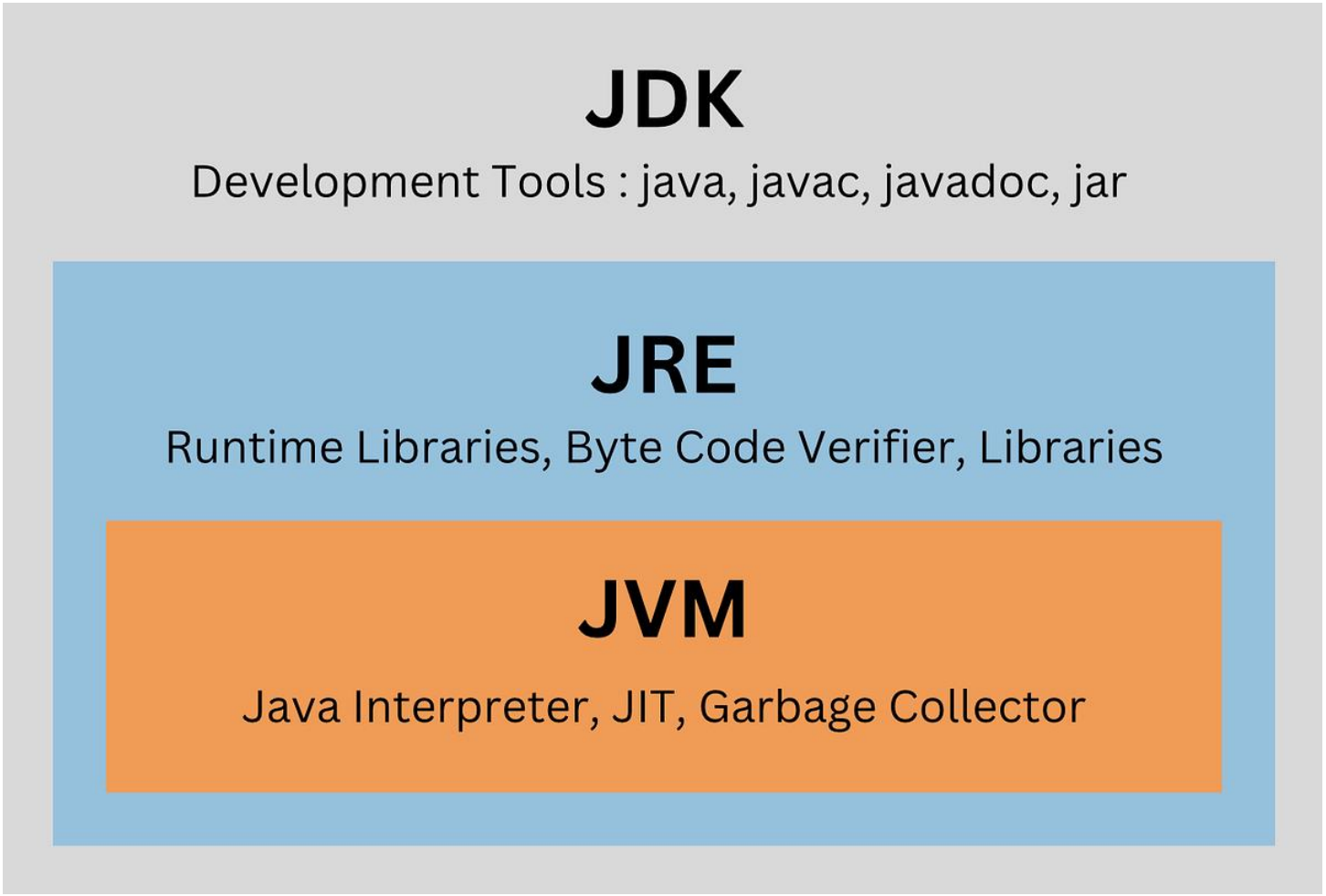
Package	Content and goal
java.awt	Basic graphic elements
java.awt.event	Graphic event handlers
javax.swing	Additional graphic elements
java.util	Important auxiliary departments (collections, scanner...)
java.io	Communication
java.text	Formats, formatting classes
javax.applet	Applets
java.lang	Basic language elements
java.math	Matematikai osztályok
java.net	Networking
java.rmi	Remote method call
java.sql	Database management

Development environments

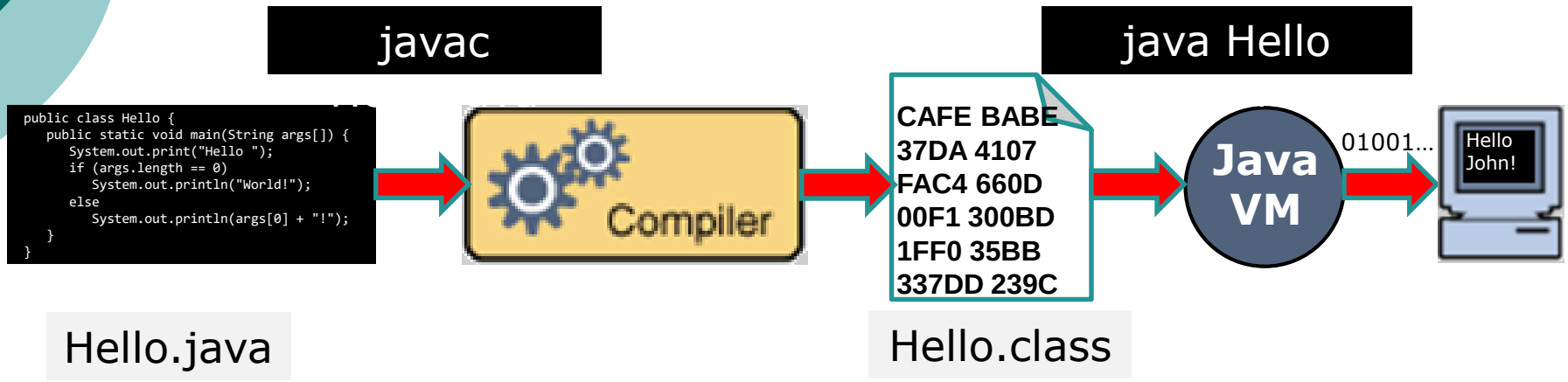
- **JDK**
  (https://www.oracle.com/java/technologies/downloads/)

- IDEs:
  - IntelliJ (https://www.jetbrains.com)
  - Eclipse (www.eclipse.org/downloads)
  - Oracle Jdeveloper (www.oracle.com/technetwork/ developer-tools/jdev/downloads)
  - Javelin (stepaheadsoftware.com/products/javelin)
  - JProfiler (www.ej-technologies.com/products/jprofiler)

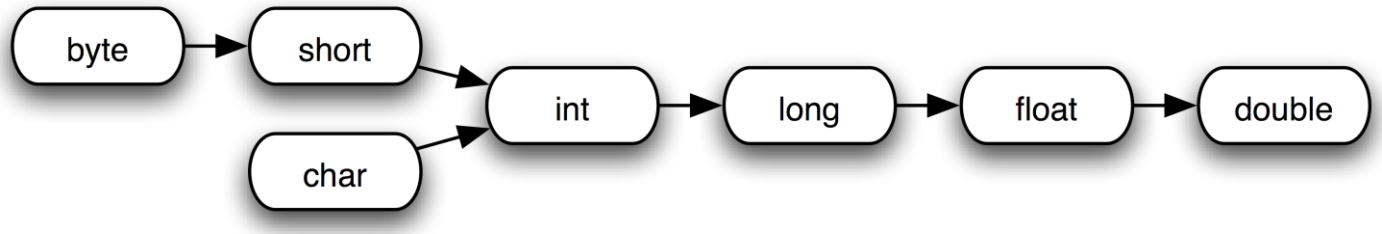
Development environments



Compiling steps



Type conversion

- Java is a **highly typed language**, and the compiler **checks type compatibility** whenever possible
- A type is compatible with another type if it can be replaced in that environment
- **Type conversions**
  - **Implicit (automatic)**: performed automatically by the compiler
  - 

  - **Explicit (forced)**: must be enforced by the programmer

Type conversion

```
double d;
int i = 5;
d = i;
```

Convert int to double implicitly. This is expansion.

```
5.0
```

```
int i;
double d = 79.4;
i  = (int) d;
```

Convert Double to int explicitly. This is narrowing.

```
79
```

```
int i;
double d = 79.4;
i  = d;
```

Syntax error!!

```
Error: incompatible types: possible
lossy conversion from double to int
```

Java language instructions

- Block
- Local variable declaration
- Blank instruction
- Expression statement
- Branches (if, switch)
- Cycles (while, do, for)
- Jump, continue, return instructions
- Exception handling instructions (throw, try-catch)

Java language instructions

- **Block**

```
{
   Instructions1
   Instructions2
   . . .
}
```

- **Local variable declaration**
  - String b= "apple";

Java language instructions

- **Blank instruction**
  - ;
- **Expression statement**
  - a+=2;

Java language instructions

- **Branches (if, switch)**

```
IF (condition) {
   Instructions
} ELSE IF (condition) {
   Instructions
} ELSE {
   Instructions
}
```

```
SWITCH (value) {
   CASE label1: instruction; BREAK;
   CASE label2: instruction; BREAK;
   CASE label3: instruction; BREAK;
   DEFAULT: instruction; BREAK;
}
```

Java language instructions

- **Cycles (while, do, for)**

```
WHILE (condition) {
    Instructions
}
```

```
DO {
    Instructions
} WHILE (condition)
```

```
FOR (Initialization; condition; increment) {
    instruction(s)
}
```

```
FOR (type element : storage) {
    instruction(s)
}
```

Java language instructions

- **Break, continue, return instructions**

```
BREAK label;
```

- the loop iterations stop and control returns from the loop immediately to the first statement after the loop

```
CONTINUE label;
```

- It breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop

```
RETURN expression;
```

- it exits the function and continues execution on the statement after the function call

Java language instructions

- **Exception handling instructions** (throw, try-catch)
  - THROW: throwing and generating an exception
  - TRY: select a protected code
  - CATCH: catching an exception, executing instructions defined in the block
  - FINALLY: finally, whether there was an exception or not, it executes

```
String s1 = sc.nextLine(); //read from a scanner
try {
    int num = Integer.parseInt(s1);
    System.out.println("Number is: "+num);
} catch (NumberFormatException nfe) {
    System.out.println("Error: "+nfe.getMessage());
}
```

Thank you for your attention!