

SEng2202

Object Oriented Programming

CHAPTER - 1

Introduction

Computer System

Hardware and **software** are two essential components of any computer system.



Hardware

Software

Computer System

Hardware

- ✓ The physical devices that make up a computer, such as the CPU, motherboard, hard disk, keyboard, mouse, etc.

Software

- ✓ A set of **programs** that run on a computer, such as the operating system, applications, drivers, etc.

Computer System

Computer Program

- A set of instructions that tells a computer what to do.
- A sequence of steps that the computer follows to perform a task.
- Written in **programming languages**.
- Used to create all kinds of software, from operating systems and word processors to games and web browsers.

Computer System

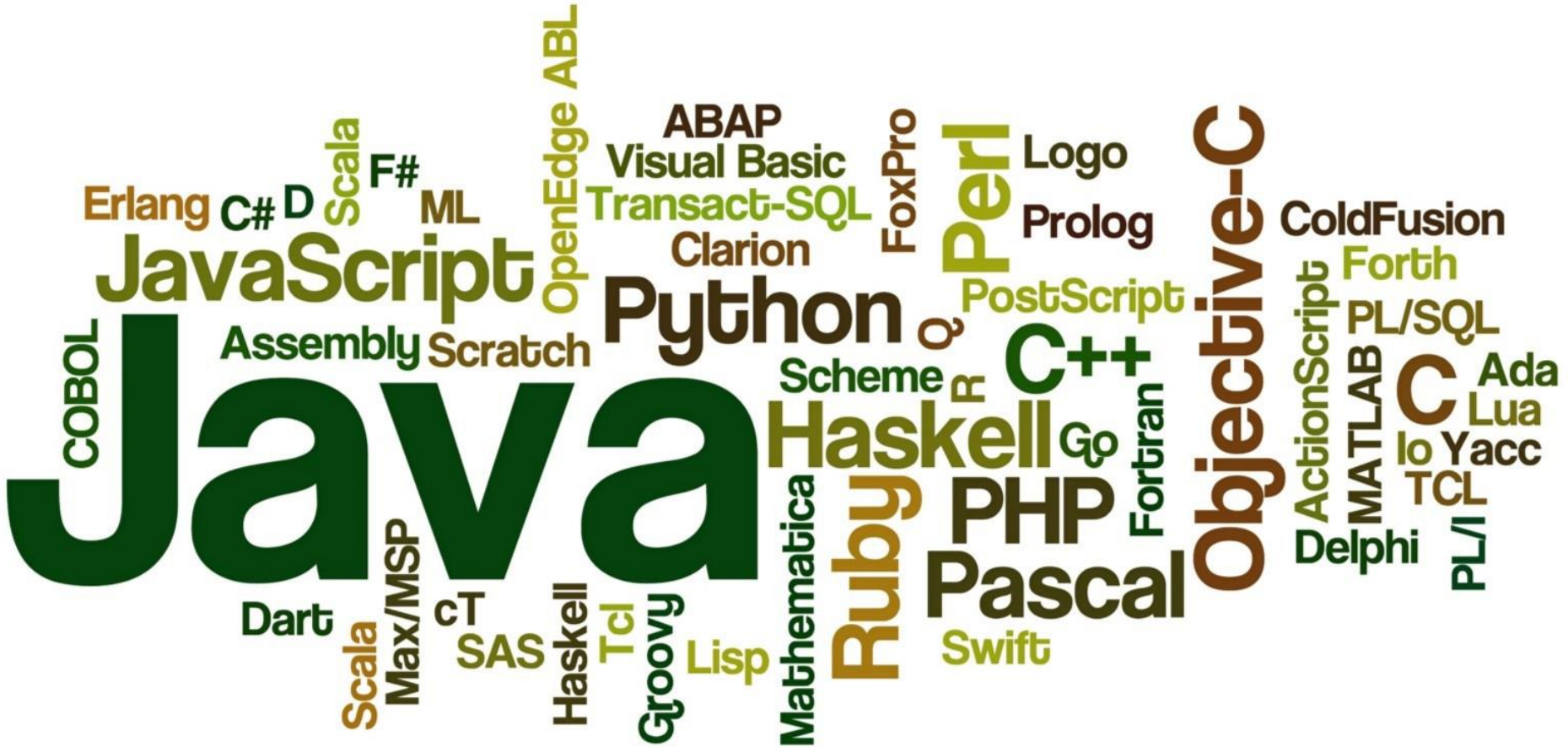
Programming language

- A formal language that specifies a set of instructions that can be used to produce various kinds of output.
- A computer language programmers use to develop software programs, scripts, or other sets of instructions for computers to execute.
- Each programming language has its own **syntax**.

Computer System

Programming languages

- Once a **programmer** learns the languages rules, syntax, and structure, they write the source code in a text editor or IDE.
- There are 1000+ programming languages.
- The choice of programming language depends on the specific application that is being developed, maintainability of the app , Scalability, Performance and Security of the app.



Programming Languages

Classifications of Programming Languages

Classification	Definition	Examples
Generation	Level of abstraction from machine code	Machine code, assembly language, high-level language, fourth-generation language
Purpose	Task for which the language is designed	General-purpose language, special-purpose language
Paradigm	Style of programming	Procedural programming, **object-oriented programming**, functional programming, logic programming

Programming Language Paradigms

Programming paradigm

- ✓ A style of programming that defines how code is written and structured.
- ✓ There are many different programming paradigms, each with its own strengths and weaknesses.

Programming Language Paradigms

Programming paradigm

Some of the most common programming paradigms include:

1. Procedural programming
2. Object-oriented programming
3. Functional programming
4. Logic programming

Programming Language Paradigms

1. Procedural programming

- Procedural programming is the oldest and most basic programming paradigm.
- It is based on the idea of writing code as a series of procedures, or functions, that each perform a specific task and can be called by other procedures or by the main program.

Programming Language Paradigms

1. Procedural programming

- Procedural programming follows a top-down approach, where the program is divided into smaller subproblems that are solved by procedures.
- Example : C, Pascal, and Fortran.

Programming Language Paradigms

2. Object-oriented programming

- Object-oriented programming (OOP) is a programming paradigm that uses **classes** and **objects** to structure a software program.
- Classes are blueprints that define the properties and methods of objects.
- Objects are instances of classes that contain data and code.
- OOP allows for creating reusable and modular code that can simplify complex problems.
- Example : Java, Smalltalk, ruby

Programming Language Paradigms

3. Functional programming

- Functional programming is a programming paradigm that is based on the idea of functions.
- Functions are self-contained units of code that take inputs and produce outputs.
- Functional programming does not use side effects, which means that a function's output does not depend on the state of the program outside of the function.
- Example : Haskell, Lisp, and Scala

Programming Language Paradigms

4. Logic programming

- Logic programming is a programming paradigm that is based on the principles of logic.
- Logic programming languages use a declarative programming style.
- The programmer specifies the desired output, rather than the steps that need to be taken to produce the output.
- Example : prolog

Object Oriented Programming



Principles of object-oriented programming (OOP)

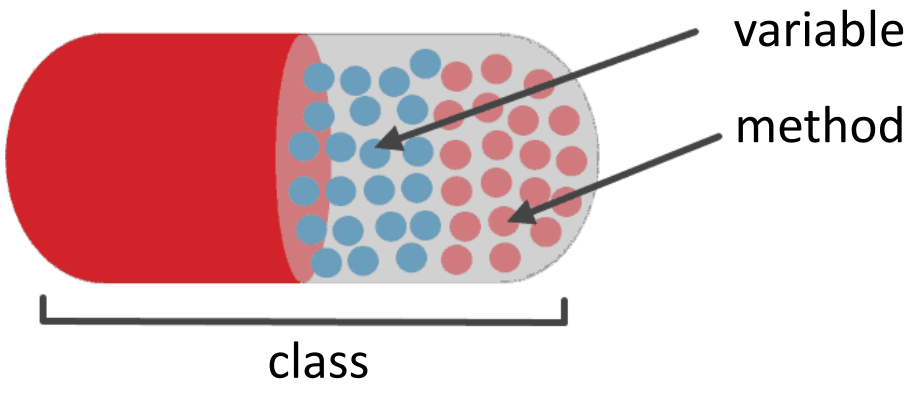
The four principles of object-oriented programming (OOP) are

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism.

Principles of object-oriented programming (OOP)

1. Encapsulation

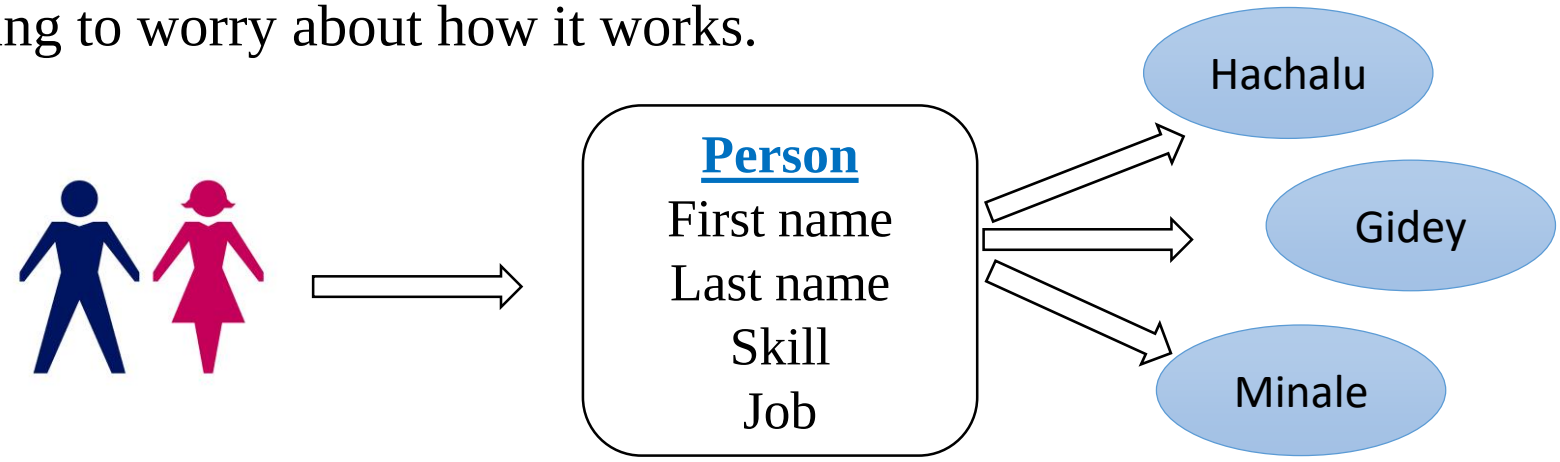
- The bundling of data and methods into a single unit called an object.
- This allows for data to be hidden from the outside world, and for methods to be accessed only through well-defined interfaces.



Principles of object-oriented programming (OOP)

2. Abstraction

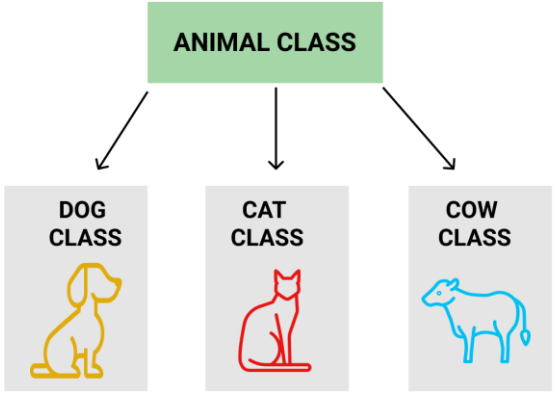
- The process of hiding the implementation details of a class from its users.
- This allows users to focus on the functionality of the class, without having to worry about how it works.



Principles of object-oriented programming (OOP)

3. Inheritance

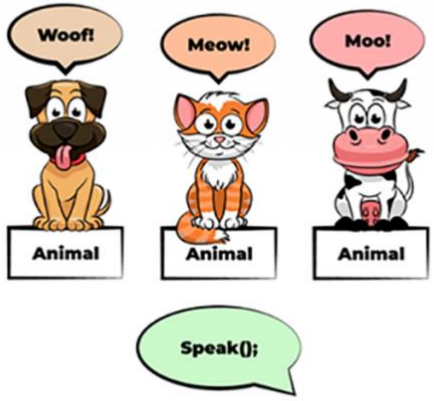
- The ability of a class to inherit the properties and methods of another class.
- This allows for code reuse and for the creation of more complex classes from simpler ones



Principles of object-oriented programming (OOP)

4. Polymorphism

- The ability of a single method to behave differently depending on the type of object it is called on.
- This allows for code to be written in a more general way, and for it to be more reusable.




Java™

Java - Early history

- Java was originally developed by **James Gosling**, Mike Sheridan, and Patrick Naughton at Sun Microsystems in 1991.
- Java was originally designed for interactive television, but it was too advanced for the digital cable television industry at the time.
- The language was initially called **Oak** after an oak tree that stood outside Gosling's office.

Java - Early history

- Later the project went by the name Green and was finally renamed Java, from Java coffee, a type of coffee from Indonesia.
- Gosling designed Java with a C/C++-style syntax that system and application programmers would find familiar.

Java 1.0

- Java was first released in 1995 as a core component of Sun Microsystems' Java platform.
- The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses.
- In 1999, Sun released the Java Development Kit (JDK) under the GNU General Public License (GPL), making Java one of the first widely-used programming languages to be released under an open-source license.

Java platform evolution

- The Java platform has evolved over time to meet the needs of developers.
- In **2004**, Sun released the Java Platform, Standard Edition 5 (Java SE 5), which introduced a number of new features, including generics, annotations, and autoboxing.
- In **2009**, Sun released Java SE 6, which added support for new features, such as the NIO.2 API and the JavaFX toolkit.
- In **2014**, Oracle released Java SE 8, which introduced a number of new features, such as lambda expressions, streams, and default methods.

Java platform evolution

- In **2017**, Oracle released Java SE 9, which added support for new features, such as modularization and the JShell interactive shell.
- In **2018**, Oracle released Java SE 10, which added support for new features, such as HTTP/2 client support and the ZGC garbage collector.
- In **2019**, Oracle released Java SE 11, which added support for new features, such as the Flight Recorder and the Records API.

Java today

- Java is a popular and versatile programming language that is used to develop a wide variety of applications.
- It is also a mature and stable platform that is backed by a large community of developers.
- If you are looking for a programming language that is powerful, versatile, and reliable, then Java is a great choice.

Java - Features

- **Object-oriented:** Java is an object-oriented programming language, which means that it is based on the concept of objects.
- **Platform-independent:** Java bytecode is platform-independent, which means that it can be run on any platform that has a Java Virtual Machine (JVM).
- **Secure:** Java is a secure language that has built-in security features, such as bytecode verification and sandboxing.

Java - Features

- **Robust:** Java is a robust language that is designed to be reliable and fault-tolerant.
- **Extensible:** Java is an extensible language that allows developers to add new features and functionality.
- **Open source:** Java is an open-source language that is freely available to anyone.

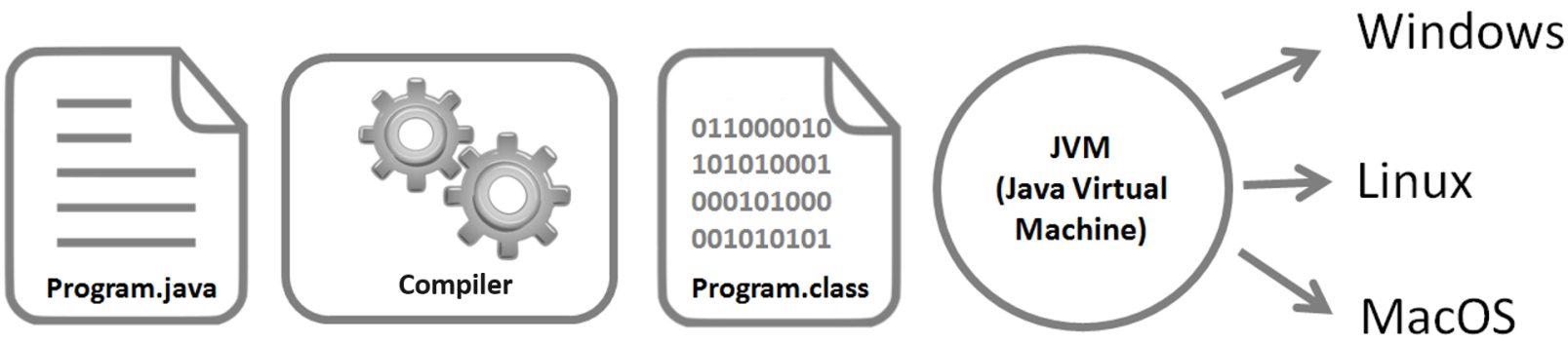
Java - The compilation process

- High level programming languages are not understood directly by a computer, so they must be converted somehow to **machine language**.
- This translation is done by
  - ✓ **compilers** (the whole program is translated into machine language and then executed) and
  - ✓ **interpreters** (the program is translated and executed one line at a time; much slower than compilers).
- Java uses a hybrid strategy.

Java - The compilation process

- Instead of compiling into machine language, Java programs are compiled into an intermediate language (**Java bytecodes**) .
- Java bytecodes serves as the machine language for a theoretical/virtual computer (**JVM** = Java Virtual Machine).
- Java then interprets those programs by simulating the JVM.

Java - The compilation process



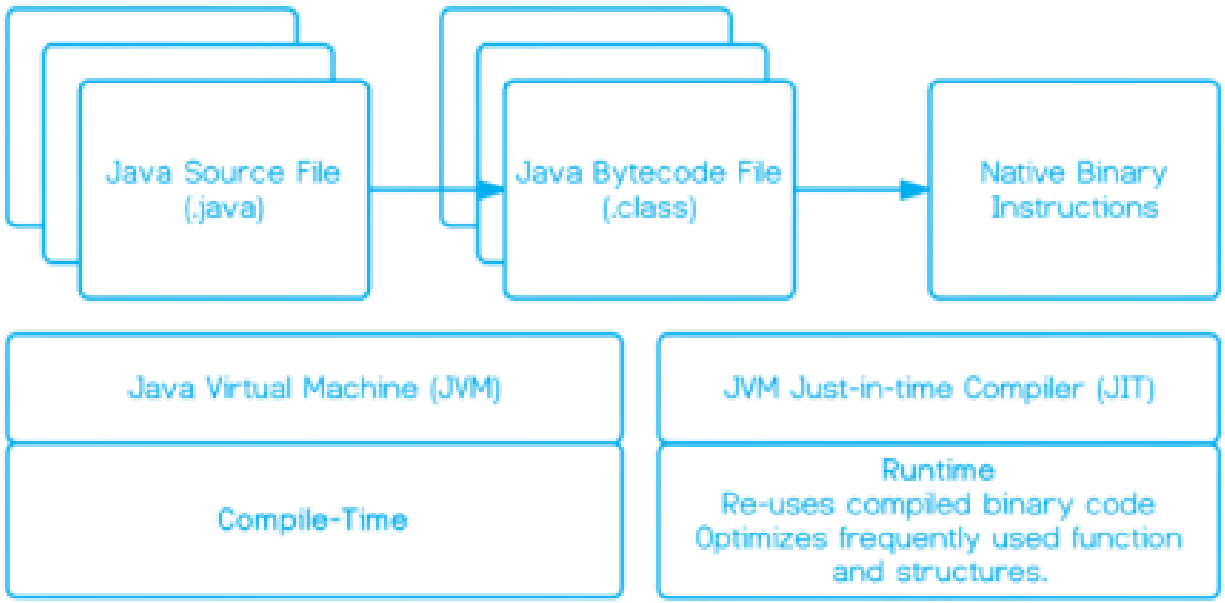
Java - JVM(Java Virtual Machine)

- **JVM** stands for Java Virtual Machine.
- It is software that provides a runtime environment to execute Java code or other programming languages that target the JVM.
- The JVM is responsible for interpreting and executing Java bytecode, which is a compiled version of the Java source code.
- It also provides features like memory management, security, and platform independence.

Java - JVM(Java Virtual Machine)

- One of the key advantages of using the JVM is that it allows developers to write code in Java or other JVM-compatible languages and run it on any platform that has a JVM installed, without having to worry about the underlying hardware or operating system.
- Some popular JVM languages include Java, Kotlin, Scala, and Groovy.
- The JVM is widely used in enterprise software development, web development, and mobile app development.

Java - JVM(Java Virtual Machine)



Java - JIT(Just-In-Time)

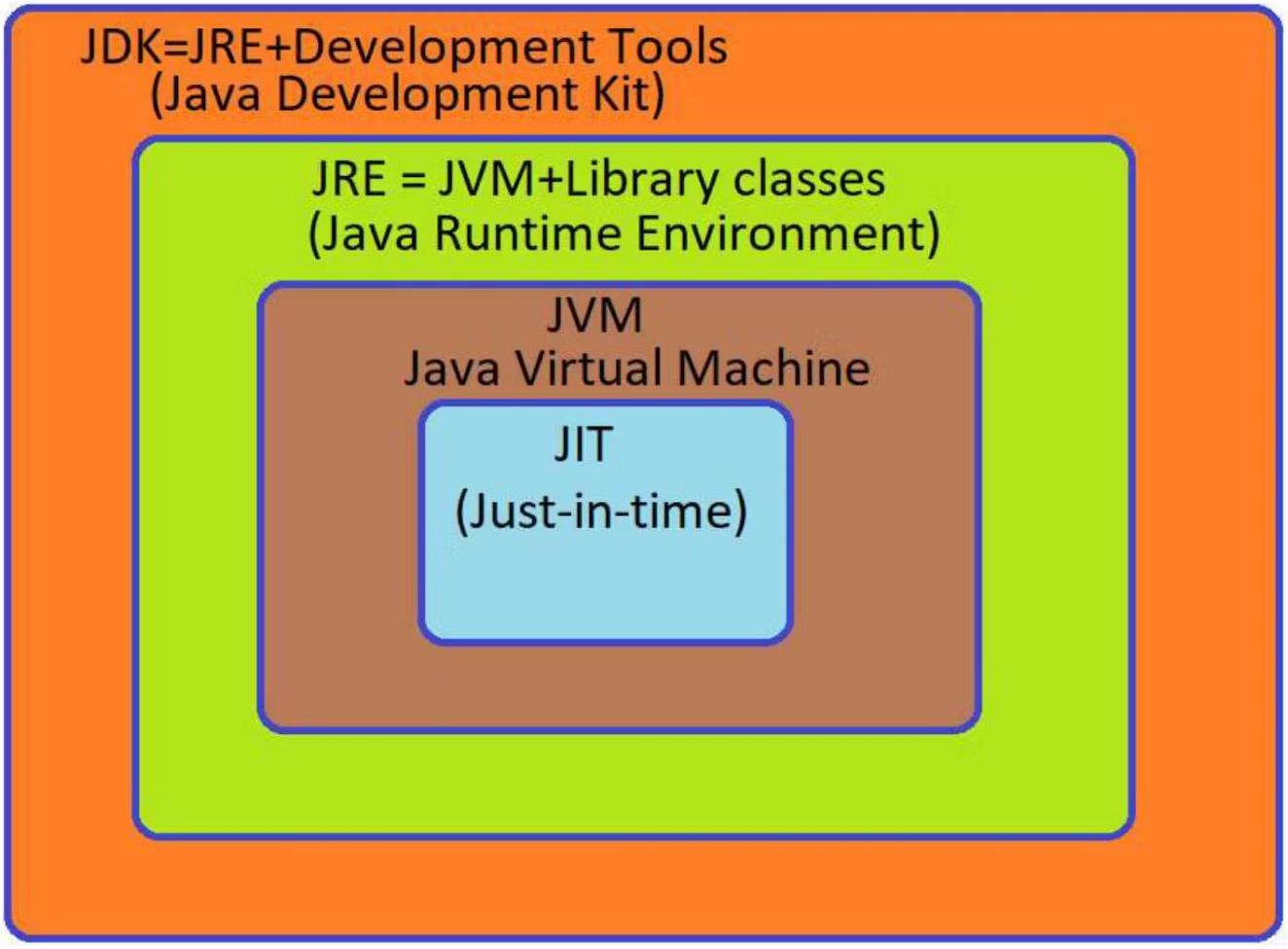
- Used to transform bytecodes into native machine code at runtime.
- the Just-In-Time (JIT) compiler, a part of the runtime environment, improves the performance of Java applications.
- A Java application runs slower than a native application due to the increased CPU and memory requirements during interpretation.
- By converting bytecodes into native machine code at run time, the JIT compiler contributes to the performance enhancement of Java applications.

Java - JIT(Just-In-Time)

- The speed of the Java code could theoretically approach that of a native application if compiling every method did not require processing time and memory use.
- JIT compilation does use memory and the processor. The JVM calls thousands of methods when it first starts up.
- Even if the software eventually achieves extremely good peak performance, compiling all of these techniques can greatly affect starting time.

Java - JDK

- The Java Software Development Kit (JDK) is a software development environment (IDE) for developing Java applications and applets.
- It includes the Java Runtime Environment (JRE), an interpreter/loader (Java Virtual Machine), a compiler, and other tools needed to develop, debug, and deploy Java programs.
- The JDK is available for Windows, macOS, and Linux. The latest version of the JDK is 17 LTS.



Java architecture

Java - Editions

There are four main editions of Java:

1. **Java SE (Java Standard Edition)**
   - ✓ The foundation of the Java platform.
   - ✓ It includes the core Java programming language, as well as APIs for things like graphics, networking, and database access.
   - ✓ Java SE is used for a wide variety of applications, from simple console programs to complex web applications.

Java - Editions

2. Java EE (Java Enterprise Edition)

- ✓ a collection of APIs that provide functionality for building enterprise-scale applications.
- ✓ Java EE includes APIs for things like web development, messaging, and security.
- ✓ Java EE is used by businesses to build applications that can handle large amounts of data and traffic.

Java - Editions

3. Java ME (Java Micro Edition)

- ✓ is a set of APIs for building small, lightweight applications that run on mobile devices and embedded systems.
- ✓ Java ME is used to build applications for smartphones, tablets, and other devices.

Java - Editions

4. JavaFX

- ✓ a set of APIs for building rich, interactive user interfaces for desktop, mobile, and web applications.
- ✓ JavaFX is a newer technology than Java SE, Java EE, and Java ME, but it is gaining popularity due to its powerful features and ease of use.

Java - Editions

Edition	Target audience	Features
Java SE	Developers	Core Java programming language, APIs for graphics, networking, database access
Java EE	Enterprise developers	APIs for web development, messaging, security
Java ME	Mobile and embedded developers	APIs for building small, lightweight applications
JavaFX	Desktop, mobile, and web developers	APIs for building rich, interactive user interfaces



Adama Science and Technology University

School of Electrical Engineering And Computing