**Java is divided into 4 platforms**

Java has evolved into multiple platforms like below

1. Java SE (Java Standard Edition)
2. Java EE (Java Enterprise Edition)
3. Java ME (Java Micro Edition)
4. Java FX(FXML)

**Difference Between Java SE, Java EE, and Java ME**

Java is one of the most popular programming languages in the world, and it powers everything from mobile applications to large-scale enterprise systems. Over the years, Java has evolved into multiple platforms tailored to different kinds of development needs. These platforms—Java SE (Standard Edition), Java EE (Enterprise Edition), and Java ME (Micro Edition).

serve different purposes, and each comes with its own set of libraries, frameworks, and tools. Understanding the differences between these editions is essential for developers to choose the right one for their projects.

**Java SE**

1. Java SE (Standard Edition) is the core of the Java programming language.
2. Java SE is often the first step for developers learning Java
3. It contains the foundational libraries, tools, and APIs (Application Programming Interfaces), which makes us to develop or build desktop applications, basic command-line utilities.

***Key Features***:

– Core Libraries: Java SE includes essential libraries for tasks such as file I/O, network communications, concurrency, and collections. These are the building blocks for any Java application.

– JVM (Java Virtual Machine): Java SE includes the JVM, which allows Java applications to run on different hardware platforms without modification.

– Swing and AWT: Java SE includes graphical user interface (GUI) toolkits like Swing and AWT (Abstract Window Toolkit), which are used to create desktop applications with graphical interfaces.

– Standard APIs: Core APIs like java.lang, java.util, and java.math provide a range of functionality from basic data manipulation to advanced mathematical operations.

– JDK (Java Development Kit): Java SE includes the JDK, which consists of tools like `javac` (the Java compiler), `java` (the Java runtime environment), and other utilities for compiling and running Java programs.

***Use Cases:***

– Desktop Applications: Java SE is ideal for building standalone applications like text editors, IDEs (Integrated Development Environments), and media players.

**What is Java EE?**

Java EE (Enterprise Edition) is built on top of Java SE and adds a set of features designed for building large-scale, distributed, and enterprise-level applications. Java EE introduces APIs for building web applications, handling large databases, managing distributed systems, and more. It’s the go-to platform for companies that need to build complex business applications.

***Key Features:***

– Servlets and JSP (JavaServer Pages): These are core technologies in Java EE for building dynamic web applications. Servlets handle requests and responses, while JSPs allow embedding Java code into HTML pages.

– Enterprise JavaBeans (EJB): EJB is used to build scalable and transactional components in enterprise applications. It simplifies the development of distributed, transactional, and secure applications.

– JPA (Java Persistence API): JPA is an ORM (Object-Relational Mapping) tool that simplifies database access by allowing developers to interact with databases using Java objects.

– Web Services Support: Java EE includes APIs for creating and consuming both SOAP and RESTful web services, making it easy to build service-oriented architectures.

– Dependency Injection (DI): This is a key feature in Java EE that allows for loosely coupled components, making code more modular and easier to maintain.

– Security and Transactions: Java EE includes robust support for declarative security and transaction management, essential for enterprise-level applications.

***Use Cases:***

– Enterprise Applications: Java EE is widely used in industries like banking, healthcare, and telecommunications to build large-scale applications that require robust security, scalability, and transactional support.

– Web Applications: Java EE is an excellent choice for building dynamic, data-driven websites and services.

***Advantages over Java SE:***

– Built for Scale: Java EE provides a rich set of features for handling complex, multi-tiered applications, making it more suitable for enterprise solutions than Java SE.

– Ecosystem of Tools: It integrates seamlessly with popular enterprise tools like application servers (e.g., GlassFish, WildFly) and databases (e.g., Oracle, MySQL).

Java ME (Micro Edition)

Java ME (Micro Edition) is designed for devices with limited resources, such as mobile phones, embedded systems, and IoT (Internet of Things) devices. It is a scaled-down version of Java SE, with a focus on minimizing memory usage and resource consumption.

Java ME was once a dominant platform for mobile development, especially before the rise of smartphones. Even today, it is used in embedded systems and low-power devices that require efficient use of hardware resources

***Key Features:***

– Connected Limited Device Configuration (CLDC): This configuration provides a stripped-down JVM suitable for devices with limited processing power and memory.

– Mobile Information Device Profile (MIDP): MIDP provides APIs for creating mobile applications, handling basic user input, and controlling network connections.

– Optimized for Embedded Systems: Java ME includes APIs for accessing hardware features like sensors, network modules, and low-level system functions.

– Lightweight: Java ME is designed to run on constrained devices with as little as a few hundred kilobytes of memory.

***Use Cases:***

– Embedded Systems: Java ME is commonly used in embedded systems such as smart appliances, industrial devices, and medical equipment.

– Feature Phones: While not as relevant today in the world of smartphones, Java ME was widely used to develop applications for early mobile phones.

– IoT Applications: With the growth of IoT, Java ME has seen a resurgence in use for small, connected devices that need to perform tasks with minimal power consumption.

***Advantages over Java SE and Java EE:***

– Resource Efficiency: Java ME is optimized for environments with tight memory and processing constraints, something neither Java SE nor Java EE is designed for.

– Portability: Java ME allows applications to run on a variety of low-resource devices without modification.

**JavaFX**

* **Purpose:** JavaFX is a modern framework for building **rich client** applications with advanced UI features.
* **Includes:** UI components (buttons, tables, charts, animations), media playback, CSS styling, FXML (XML-based UI design), and more.
* **Usage:** Used for developing **desktop applications with a modern UI**, supporting animations, CSS styling, and web integration.
* **Replaces Swing?:** While JavaFX is more advanced, **Swing is still used**, and JavaFX hasn’t fully replaced it.

**Key Differences**

|  |  |  |
| --- | --- | --- |
| Feature | Java SE (Swing/AWT) | JavaFX |
| UI Framework | Swing & AWT | JavaFX |
| Styling | Limited (basic) | CSS-based styling |
| Animations | Difficult to implement | Built-in support |
| FXML Support | No | Yes (separates UI from logic) |
| Performance | Less optimized for modern UIs | GPU-accelerated, better performance |
| Learning Curve | Easier (for Swing users) | Slightly higher due to new concepts |