**JAVA 21 Features**

Java 21 is a LTS version. It was released on September 2023

**Features:**

* **Pattern Matching for Switch**

It simplifies switch statements, making them more concise and readable.The main goal of this feature is to allow patterns in *switch case* labels and improve the expressiveness of *switch* statements and expressions.  
  
Example:



* **String** **Template**

Java 21 offers several mechanisms for composing strings with string literals and expressions. Some of these are String concatenation. These complement Java’s existing string literals and text blocks by coupling literal text with template expressions and template processors to produce the desired results.

Example:

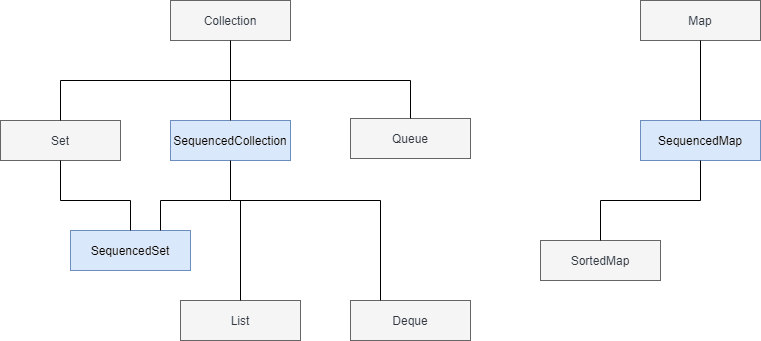


* **Sequenced Collections**

Until Java 21 in the collection framework, no collection type represents a sequence of elements with a defined encountered order.  For instance, *List* and *Deque* interfaces define an encounter order, but their common super type *Collection* doesn’t. In the same way, *Set* doesn’t define an encounter order, but subtypes such as *LinkedHashSet* or *SortedSet* do.

Java 21 introduced three new interfaces to represent sequenced collections, sequenced sets, and sequenced maps.

A sequenced collection is a collection whose elements have a defined encounter order. It has first and last elements, and the elements between them have successors and predecessors.



Example:



* **Unnamed Patterns and Variables:**

Java 21 introduces Unnamed Patterns and Variables, making the code more concise and expressive. Instead of variable we can replace it with \_

Example:



* **Performance Improvements**

Java has the feature of Z Garbage Collection (ZBC) which performs all the expensive works concurrently, without stopping the execution of application threads. Now with the improvement of this cleaning tool called ‘ZBC’, it cleans up the unused memory. It sorts stuff in the memory into ‘new’ and ‘old’ notes.

For example: Every program you run on your computer uses something called ‘memory’ to store and manage information. As the program runs, it creates objects like notes in this memory. Over time, many of these notes aren't needed anymore during the development, and if they're not removed, they'll clog up or block the memory, like unused papers filling up a desk, and making everything slower. So ZGC separates them into 'new' and 'old' notes. Where it works as in the following manner- newer notes are often not needed after a short time, while older ones might still be important. So, by targeting and cleaning up the new notes more frequently, it ensures that the memory desk remains less complex and more efficient.