Java 9

Key Features of Java 9

1. Private Methods in Interfaces
2. Immutable Collections
3. Stream API Improvements
4. Optional Class Improvements
5. JShell - The Interactive Java Shell
6. Module System (Project Jigsaw)
7. HTTP/2 Client
8. Process API Improvements
9. Miscellaneous Changes

1. Private Methods in Interfaces

What are Private Methods in Interfaces?

Java 9 allows interfaces to have private methods, which can be used to share code between default methods. This feature promotes code reusability within interfaces. Using private methods, now **encapsulation** is possible in interfaces as well.

Using private methods in interfaces have four rules :

- The private interface method cannot be **abstract**.
- A private method can be used only inside interface.
- A private static method can be used inside other static and non-static interface methods.
- A private non-static method cannot be used inside private static methods.

2. Immutable Collections

What are Immutable Collections?

Java 9 introduces static factory methods for creating immutable collections. These methods provide a convenient way to create collections that cannot be modified.

Create Immutable List

Use *List.of()* static factory methods to create immutable lists. It has following different overloaded versions –

```
static <E> List<E>  of()
static <E> List<E>  of(E e1)
static <E> List<E>  of(E e1, E e2)
static <E> List<E>  of(E e1, E e2, E e3)
static <E> List<E>  of(E e1, E e2, E e3, E e4)
static <E> List<E>  of(E e1, E e2, E e3, E e4, E e5)
static <E> List<E>  of(E e1, E e2, E e3, E e4, E e5, E e6)
static <E> List<E>  of(E e1, E e2, E e3, E e4, E e5, E e6, E e7)
```

```
static <E> List<E>  of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8)
static <E> List<E>  of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9)
static <E> List<E>  of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9, E e10)
//varargs
static <E> List<E>  of(E... elements)
```

The List instances created by these methods have the following characteristics:

1. These lists are immutable. Elements cannot be added, removed, or replaced in these lists. Calling any mutator method (i.e. add, addAll, clear, remove, removeAll, replaceAll) will always cause *UnsupportedOperationException* to be thrown.
2. They do not allow null elements. Attempts to add null elements result in *NullPointerException*.
3. They are serializable if all elements are serializable.
4. The order of elements in the list is the same as the order of the provided arguments, or of the elements in the provided array.

We can also create set and map as immutable collection using Set.of() and Map.of() methods.

3. Stream API Improvements

What are Stream API Improvements?

Java 9 enhances the Stream API with new methods like takeWhile(), dropWhile(), and iterate(). These methods make it easier to work with streams in a functional style.

Streams were introduced in Java to help developers perform aggregate operations from a sequence of objects. With Java 9, few more methods are added to make streams better.

Java 9 improvements in Stream API:

- takeWhile(Predicate Interface)
- dropWhile(Predicate Interface)
- iterate()
- ofNullable()

Stream takeWhile(Predicate Interface):

The *takeWhile()* method takes all the values until the predicate returns false. It returns, in case of ordered stream, a stream consisting of the longest prefix of elements taken from this stream matching the given predicate.

Stream dropWhile(Predicate Interface) method:

The *dropWhile()* method throw away all the values at the start until the predicate returns true. It returns, in case of ordered stream, a stream consisting of the remaining elements of this stream after dropping the longest prefix of elements matching the given predicate.

Stream iterate() method

The *iterate()* method now has hasNext predicate as a parameter which stops the loop once hasNext predicate returns false.

Stream ofNullable() method:

In Java 9, the *ofNullable()* method lets you create a single-element stream that wraps a value if not null, or is an empty stream otherwise.

4. Optional Class Improvements

What are Optional Class Improvements?

Java 9 introduces new methods to the Optional class, such as ifPresentOrElse(), or(), and stream(). These methods enhance the functionality of the Optional class, providing more control and flexibility.

Optional stream() method:

If a value is present, it returns a sequential Stream containing only that value, otherwise returns an empty Stream.

Optional ifPresentOrElse() method:

If a value is present, perform the given action with the value, otherwise, perform the given empty-based action.

Optional or() method:

If a value is present, returns an Optional describing the value, otherwise returns an Optional produced by the supplying function.

5. JShell - The Interactive Java Shell

What is JShell?

JShell is an interactive tool introduced in Java 9 that allows developers to run Java code snippets quickly without creating a complete program. It is useful for testing code, experimenting with new features, and learning Java.

6. Module System (Project Jigsaw)

What is the Module System?

Java 9 introduces the module system, also known as Project Jigsaw, which provides a way to modularize applications and the JDK itself. The module system enhances encapsulation and improves the maintainability of large codebases.

Example: Defining a Module

Create a module by defining a module-info.java file:

```
module com.example.myapp {
    requires java.base;
    exports com.example.myapp;
}
```

Explanation

- module com.example.myapp declares a module named com.example.myapp.
- requires java.base specifies that the module depends on the java.base module.
- exports com.example.myapp makes the package com.example.myapp accessible to other modules.

7. HTTP/2 Client

What is the HTTP/2 Client?

Java 9 introduces a new HTTP/2 client API that supports HTTP/2, WebSocket, and asynchronous requests. This client provides a more modern and efficient way to handle HTTP communications.

Explanation

- HttpClient.newHttpClient() creates a new HTTP/2 client.
- HttpRequest.newBuilder() constructs an HTTP request.
- client.send() sends the request and receives the response.

8. Process API Improvements

What are Process API Improvements?

Java 9 enhances the Process API to provide more control over native processes. The improvements include methods for managing process trees, getting process information, and handling process output.

Explanation

- ProcessBuilder starts a new native process (notepad.exe in this example).
- ProcessHandle provides information about the process, such as its ID and status.

9. Miscellaneous Changes:

Java 9 introduces several other changes, including:

- **Multi-Release JARs**: Support for JAR files that can contain classes for different Java versions.
- **Unified JVM Logging**: A new logging framework for the JVM, providing a consistent way to log messages.
- **Enhanced @Deprecated**: The @Deprecated annotation now supports since and forRemoval attributes.
- **Compact Strings**: Optimizes memory usage for String objects by using byte arrays instead of character arrays.