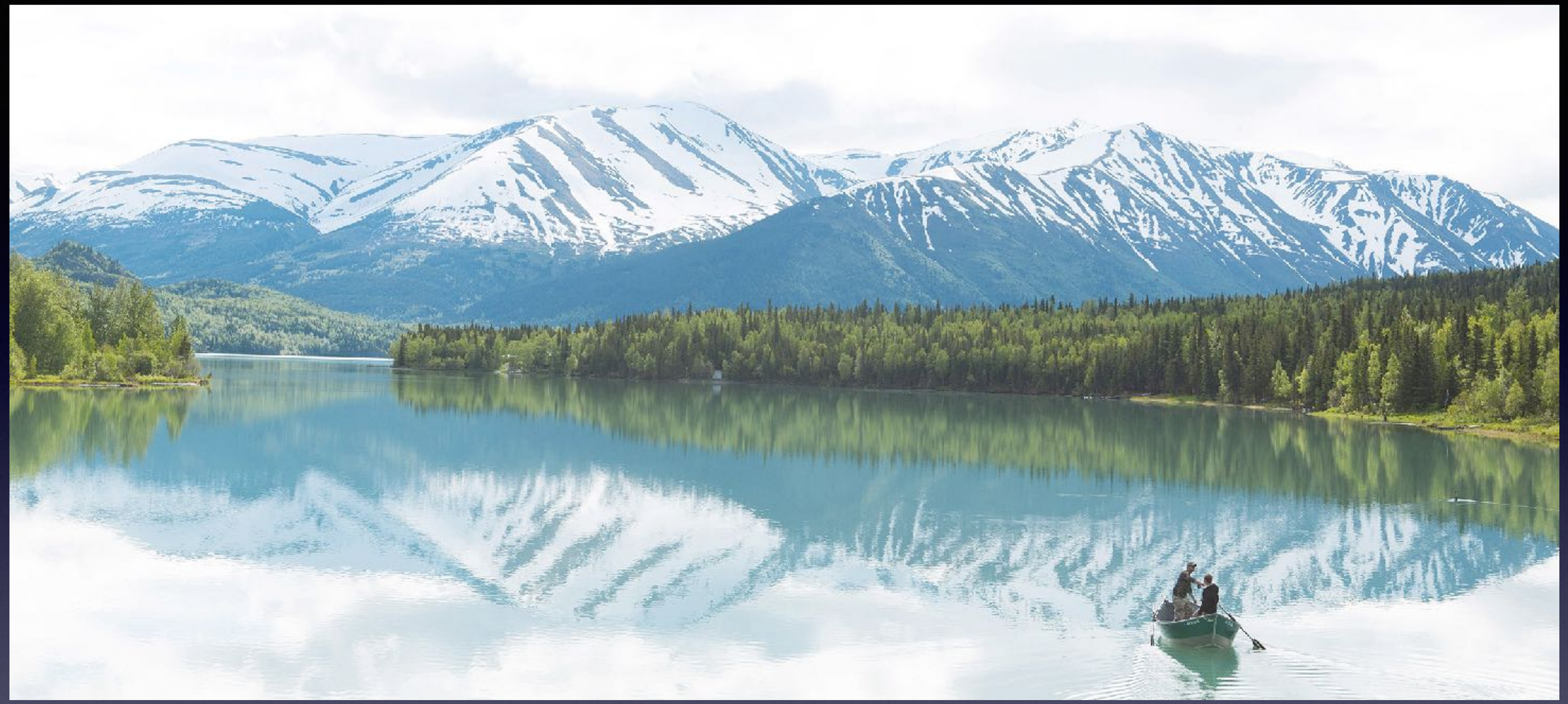


Java Training

Jan - Feb 2023

Java 8

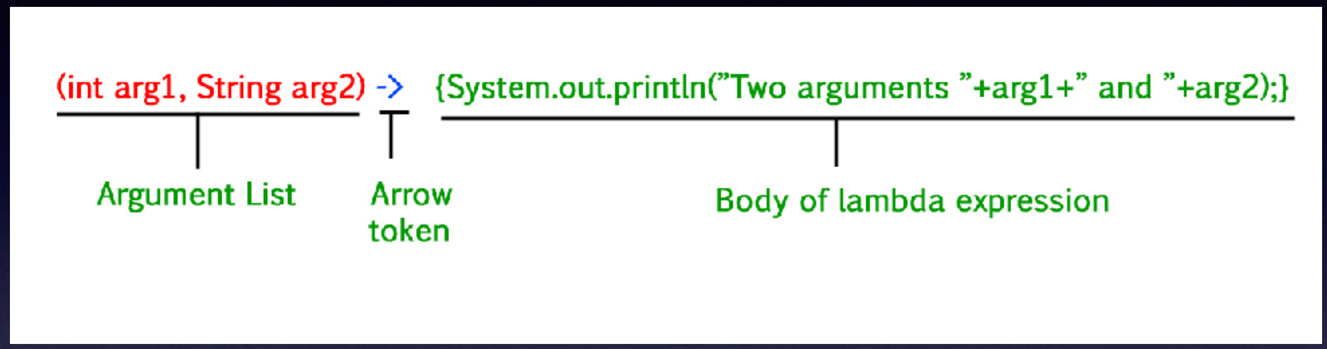
- Lambda Expressions
- Method Reference
- Functional Interface
- Optional
- Predicate
- Default Method
- Base64
- Stream
- String Joiner
- Collectors
- forEach
- Date/Time API
- Nashorn Script Engine
- I/O & Security Improvements

Lambda Expressions

- Helps us to write our code in functional style.
- Describe what you want, rather than how to get it.
- Provides a clear way to implement Single Abstract Method by using an expression
- It is very useful in collection library in which it helps to iterate, filter and extract data.

Lambda Expressions Syntax

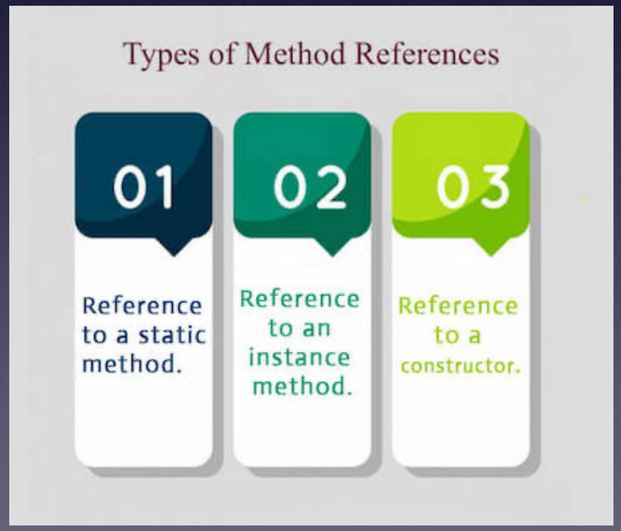
- Standard Syntax
- Parameter Type
- Multiple Lines of Code
- Single Parameter with Inferred Type
- Method References
- No Parameter



Method Reference

- It is used to refer method of functional interface .
- It is compact and easy form of lambda expression.

- 01: ContainingClass::staticMethodName
- 02: containingObject::instanceMethodName
- 03: ClassName::**new**



Functional Interface

- An Interface that contains only one abstract method.
- It can have any number of default and static methods.
- It can also declare methods of object class.
- Also known as Single Abstract Method Interfaces.

Optional

- It is a public final class.
- Which is used to deal with NullPointerException in Java application.
- Provides methods to check the presence of value.

Predicate

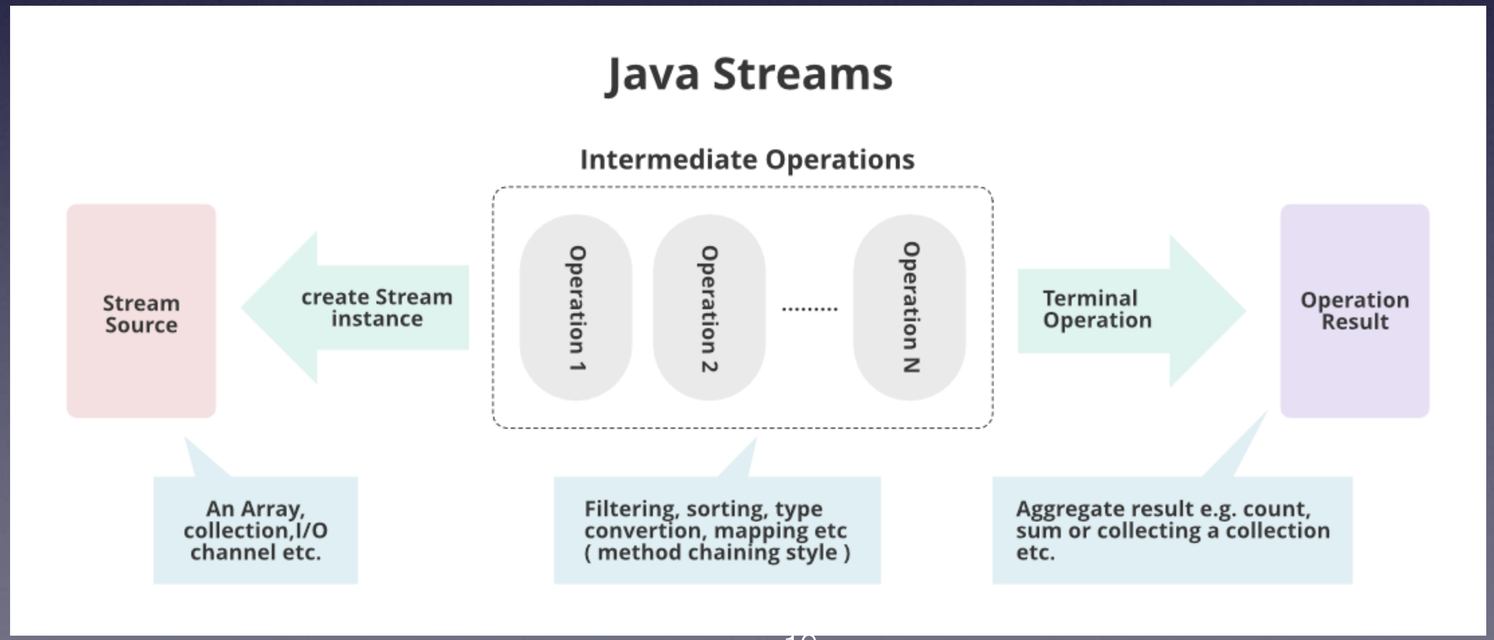
- defined in the java.util.function package.
- It improves manageability of code, helps in unit-testing them separately
- Contain some methods like:
    - isEqual(Object targetRef)
    - and(Predicate other)
    - negate()
    - or(Predicate other)
    - test(T t)

Default Methods

- Java provides a facility to create default methods inside the interface.
- Methods which are defined inside the interface and tagged with default keyword are known as default methods.
- These methods are non-abstract methods and can have method body.

Stream

- Java 8 java.util.stream package consists of classes, interfaces and an enum to allow functional-style operations on the elements.
- It performs lazy computation. So, it executes only when it requires



StringJoiner

- Used for joining Strings making use of a delimiter, prefix, & suffix.
- The default value is returned only when the StringJoiner is empty.
- merge() : It adds the contents of the given StringJoiner **without prefix and suffix** as the next element.
- Collectors.joining() internally uses StringJoiner to perform the joining operation.

Collectors

- Collectors is a final class that extends Object class.
- It provides reduction operations, such as accumulating elements into collections, summarizing elements according to various criteria etc.

forEach

- Java provides a new method forEach() to iterate the elements.
- It is defined in Iterable and Stream interfaces.
- It is a default method defined in the Iterable interface.
- Collection classes which extends Iterable interface can use forEach() method to iterate elements.
- This method takes a single parameter which is a functional interface.
- So, you can pass lambda expression as an argument.

Date/Time API

- Java has introduced a new Date and Time API since Java 8.
- The java.time package contains Java 8 Date and Time classes.

Drawbacks of existing Date/Time API's

- **Thread safety:** The existing classes such as Date and Calendar does not provide thread safety. Hence it leads to hard-to-debug concurrency issues that are needed to be taken care by developers. The new Date and Time APIs of Java 8 provide thread safety and are immutable, hence avoiding the concurrency issue from developers.

- **Bad API designing:** The classic Date and Calendar APIs does not provide methods to perform basic day-to-day functionalities. The Date and Time classes introduced in Java 8 are ISO-centric and provides number of different methods for performing operations regarding date, time, duration and periods.

- **Difficult time zone handling:** To handle the time-zone using classic Date and Calendar classes is difficult because the developers were supposed to write the logic for it. With the new APIs, the time-zone handling can be easily done with Local and ZonedDate/Time APIs.

Why Default Method ?

- Before Java 8, interfaces could have only abstract methods.
- The implementation of these methods has to be provided in a separate class.
- So, if a new method is to be added in an interface, then its implementation code has to be provided in the class implementing the same interface.
- To overcome this issue, Java 8 has introduced the concept of default methods which allow the interfaces to have methods with implementation without affecting the classes that implement the interface.

Nashorn JavaScript Engine

- Nashorn is a JavaScript engine.
- It is used to execute JavaScript code dynamically at JVM.
- You can execute JavaScript code by two ways
    - Using jjs command-line tool, and
    - By embedding into Java source code.

Nashorn cont.

- The new default JavaScript engine for the JVM as of Java 8.
- Many sophisticated techniques have been used to make Nashorn orders of magnitude more performant than its predecessor called Rhino, so it is a worthwhile change.

Java Base64

- Basic Encoding and Decoding
  - A-Za-z0-9+/
- URL and Filename Encoding and Decoding
  - A-Za-z0-9+_
  - url and file name safe
- MIME
  - MIME friendly format
  - Represented in lines of no more than 76 characters each

Java Parallel Array Sorting

- The parallelSort() method has added to java.util.Arrays class
- It uses the JSR 166 Fork/Join parallelism common pool to provide sorting of arrays.
- It is an overloaded method.

Security Enhancements

- Java 8 : TLS 1.1, 1.2 (default) at client side
- Advanced Encryption Standard (AES) and Password-Based Encryption (PBE) algorithms
    - PBEWithSHA256AndAES_128
    - PBEWithSHA512AndAES_256
-

Java IO Improvements

- Files.list (Path dir): Returns a lazily filled stream, each element of which represents a directory entry.
- Files.lines (Path path): Reads all the lines from a stream.
- Files.find (): Returns a stream filled by a path after searching for files in the file tree rooted at a provided beginning file and many more.
- BufferedReader.lines (): Returns a stream containing all of the elements of BufferedReader’s lines and much more
- Buffered

Java 9

- Modular System
- A New HTTP Client
- Process API
- Try-With-Resources
- Diamond Operator Extension
- Interface Private Method
- JShell Command Line Tool
- JCMD Sub-Commands
- Multi-Resolution Image API
- Variable Handles
- Publish-Subscribe Framework
- Unified JVM Logging
- Immutable Set
- Optional to Stream

Modular System

- One of the big changes or java 9 feature is the Module System.
- Before Java SE 9 versions, we are using Monolithic Jars to develop Java-Based applications.
  - This architecture has lot of limitations and drawbacks. To avoid all these shortcomings, Java SE 9 comes with the Module System.
- Oracle Corp. introduced the following features as part of Jigsaw Project:
  - Modular JDK
  - Modular Java Source Code
  - Modular Run-time Images
  - Encapsulate Java Internal APIs
  - Java Platform Module System

Interface Private Methods

- Improve code re-usability inside interfaces
    - if two default methods needed to share code
- Using private methods in interfaces have four rules
    1. Private interface method cannot be abstract
    2. Private method can be used only inside interface
    3. Private static method can be used inside other static & non-stati methods
    4. Private non-static methods cannot be used inside private static methods

HTTP/2

- Support for stateful connections.
- Long-running connections.
- Multiplexing. Multiple requests are allowed at the same time, on the same connection.
- Binary format. More compact.
- Single Connection to the server reduces the number of round trips needed to set up multiple TCP connections.
- Bidirectional communication using push requests
- Data compression of HTTP headers.

HTTP/2 Client

- HTTPClient
  - For creating and sending requests
- HTTPRequest
  - To construct the request that is sent via HTTPClient
- HTTPResponse
  - To hold the response that is sent

Process API Updates

- Prior to Java 5, the only way to spawn a new process was to use the following method
  - Runtime.getRuntime().exec()
- Then in Java 5, ProcessBuilder API was introduced which supported a cleaner way of spawning new processes.
- Now Java 9 is adding a new way of getting information about current and any spawned process.
- To get information of any process, now you should use java.lang.ProcessHandle.Info interface.
- This interface can be useful in getting lots of information e.g.
  - the command used to start the process
  - the arguments of the command
  - time instant when the process was started
  - total time spent by it and the user who created it

Collection API Updates

- create immutable collections using new factory methods
    - immutable list
    - immutable set
    - immutable map

JShell

- Java Shell also known as REPL
- Many languages already feature an interactive Read-Eval-Print-Loop, and Java now joins this club.
- The main advantage of using jshell is that you can test your partial code (individual statements, methods etc.) here without writing the complete program and then check the various possible scenarios.
- You can launch jshell from the console and directly start typing and executing Java code.
- The immediate feedback of jshell makes it a great tool to explore APIs and try out language features.

Reactive Streams

- A specification for
    - Asynchronous stream processing
    - With non-blocking back pressure.
- This allows developers to handle
    - Large amounts of data
    - Complex operations
    - In a more efficient and responsive way.

Reactive Streams - Publisher

- Interface represents a source of stream data.
- It can be implemented by any class that can emit a stream of data
    - A database query
    - A web service call

Reactive Streams - Subscriber

- Interface represents a consumer of stream data.
- It can be implemented by any class that can receive and process a stream of data
    - A GUI component
    - A data storage class

Reactive Streams - Subscription

- Interface represents a connection between a publisher and a subscriber.
- It allows the subscriber to control the flow of data
    - Requesting more data
    - Cancelling the subscription

Try-With-Resources

- In Java 7, the try-with-resources syntax requires a fresh variable to be declared for each resource being managed by the statement.
- In Java 9 there is an additional refinement: if the resource is referenced by a final or effectively final variable, a try-with-resources statement can manage a resource without a new variable being declared

Anonymous Inner classes and Diamond Operator

- Diamond operator was introduced as a new feature in java SE 7. The purpose of diamond operator is to avoid redundant code by leaving the generic type in the right side of the expression.
- Java 7 allowed us to use diamond operator in normal classes but it didn’t allow us to use them in anonymous inner classes.
- Java 9 improved the use of diamond operator and allows us to use the diamond operator with anonymous inner classes.

Java 10

- Local Variable Type Inference
- Unmodifiable Collections

Java 11

- New String Methods
- New File Methods
- Pattern Recognizing Methods
- TimeUnit Conversion
- Launch Single-File Programs Without Compilation

Java 11
Removed Features and Options

- Removal of com.sun.awt.AWTUtilities Class
- Removal of Lucida Fonts from Oracle JDK
- Removal of appletviewer Launcher
- Oracle JDK's javax.imageio JPEG Plugin No Longer Supports Images with alpha
- Removal of sun.misc.Unsafe.defineClass
- Removal of Thread.destroy() and Thread.stop(Throwable) Methods
- Removal of sun.nio.ch.disableSystemWideOverlappingFileLockCheck Property
- Removal of sun.locale.formatasdefault Property
- Removal of JVM-MANAGEMENT-MIB.mib
- Removal of SNMP Agent
- Removal of Java Deployment Technologies
- Removal of JMC from the Oracle JDK
- Removal of JavaFX from the Oracle JDK
- JEP 320 Remove the Java EE and CORBA Module

Java 11
Deprecated Features and Options

- ThreadPoolExecutor Should Not Specify a Dependency on Finalization
- JEP 335 Deprecate the Nashorn JavaScript Engine
- Deprecate -XX+AggressiveOpts
- Obsolete Support for Commercial Features
- Deprecate Stream-Based GSSContext Methods
- JEP 336 Deprecate the Pack200 Tools and API

Java 12

- Language Changes and Features
- Switch Expressions
- Pattern Matching for instanceof
- JVM Changes
    - Shenandoah: A Low-Pause-Time Garbage Collector
        - Garbage collecting a 200 GB heap or a 2 GB heap should have a similar low pause behavior.
    - Default CDS Archives
        - The Class Data Sharing (CDS) feature helps reduce the startup time and memory footprint between multiple Java Virtual Machines.

Java 13

- Switch Expressions
- Text Blocks

Java 14

- Records
- Incubating Features
- JVM/HotSpot Features

Java 15

- Sealed Classes
- Hidden Classes

Java 16

- Day Period Support
- Add Stream.toList Method
- Vector API Incubator

Java 17

- LTS Definition