Contents

[General 2](#_Toc185075996)

[Links 2](#_Toc185075997)

[Multi module maven project 2](#_Toc185075998)

[Pass compiler arguments in Maven 4](#_Toc185075999)

[Fork / Join framework in Java 5](#_Toc185076000)

[Reflection vs Proxies vs Method handles 6](#_Toc185076001)

[Functional 7](#_Toc185076002)

[Functional Interfaces 7](#_Toc185076003)

[Functional Composition 8](#_Toc185076004)

[Immutability 9](#_Toc185076005)

[3 ways of immutable collection 9](#_Toc185076006)

[Unmodifiable collection 9](#_Toc185076007)

[Immutable collection factory methods 9](#_Toc185076008)

[Immutable copies 9](#_Toc185076009)

[Other immutable options 11](#_Toc185076010)

[Streams 12](#_Toc185076011)

# General

## Links

Java certification -> [Become a Java SE 21 Developer - Oracle MyLearn](https://mylearn.oracle.com/ou/learning-path/become-a-java-se-21-developer/138845)

## Multi module maven project

Parent:

mvn archetype:generate -DgroupId=com.javalab -DartifactId=parent-project

pom.xml --> <packaging>pom</packaging>

<modules>

<module>core</module>

<module>api</module>

</modules>

Children:

cd parent-project

mvn archetype:generate -DgroupId=com.javalab -DartifactId=core

mvn archetype:generate -DgroupId=com.javalab -DartifactId=api

<parent>

<artifactId>parent-project</artifactId>

<groupId>com.javalab</groupId>

<version>1.0-SNAPSHOT</version>

</parent>

Ref: [Multi-Module Project with Maven | Baeldung](https://www.baeldung.com/maven-multi-module)

Add a dependency of core to the api module

## Pass compiler arguments in Maven

In some situation we may need to pass some Java compiler arguments during maven compile process.

<plugin>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.13.0</version>  
 <configuration>  
 <compilerArgs>  
 <arg>-Xlint:deprecation</arg>  
 </compilerArgs>  
 </configuration>  
</plugin>

Output:

[WARNING] /C:/work/personal/repos/javalab/performance/src/main/java/com/javalab/gc/HelloClass.java:[16,20] finalize() in java.lang.Object has been deprecated

[WARNING] /C:/work/personal/repos/javalab/performance/src/main/java/com/javalab/gc/HelloClass.java:[18,14] finalize() in java.lang.Object has been deprecated

Ref: [Apache Maven Compiler Plugin – Pass Compiler Arguments](https://maven.apache.org/plugins/maven-compiler-plugin/examples/pass-compiler-arguments.html)

## Fork / Join framework in Java

<Real work example should be prepared>

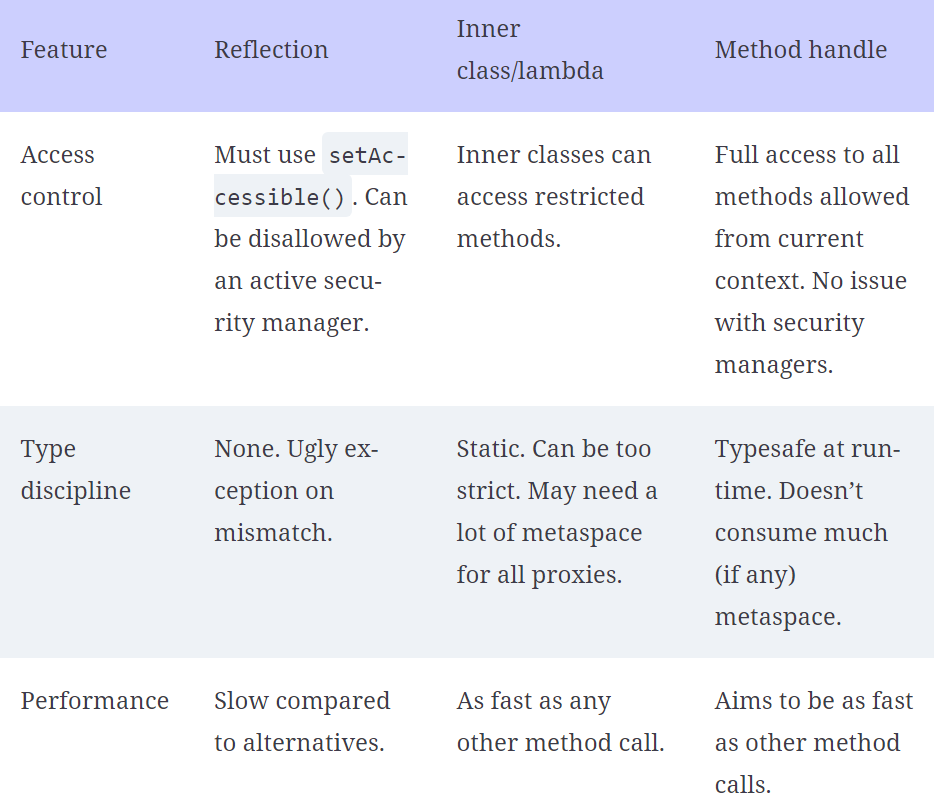
Ref: [Guide to the Fork/Join Framework in Java | Baeldung](https://www.baeldung.com/java-fork-join)

[Java Fork-Join pool. The purpose of a Fork-Join Pool in Java… | by Gathila Harischandra | Medium](https://medium.com/@gathilaharism/java-fork-join-pool-with-an-example-320a0d3d5b4c)

## Reflection vs Proxies vs Method handles

Why method handlers are a better choice?

ThreeTypeAccess.Java example



Ref: [17 Modern internals | The Well-Grounded Java Developer, Second Edition](https://learning.oreilly.com/library/view/the-well-grounded-java/9781617298875/OEBPS/Text/17.htm#heading_id_13) (17.3.4)

# Functional

<https://learning.oreilly.com/library/view/a-functional-approach/9781098109912/>

## Functional Interfaces

* 40+ functional interfaces are provided OOTB in JDK ([java.util.function (Java SE 23 & JDK 23)](https://docs.oracle.com/en/java/javase/23/docs/api/java.base/java/util/function/package-summary.html))

4 categories of functional interfaces

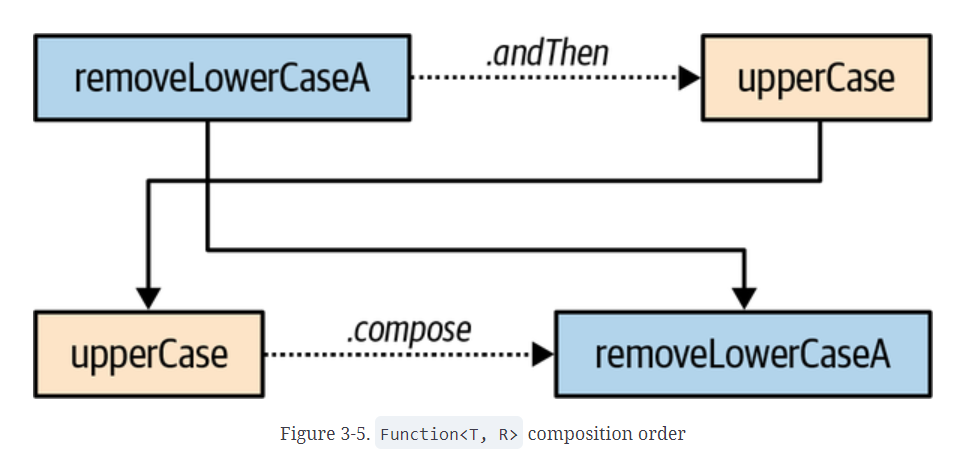
|  |  |
| --- | --- |
| **Functions**  Accepts arguments and return results | **Consumers**  Only accept arguments but do not return results |
| **Suppliers**  Do not accept arguments and only return results | **Predicates**  Accept arguments to test against an expression and return a Boolean primitive as their result |

|  |  |  |
| --- | --- | --- |
| Fis | Input | Output |
| Function | Yes | Yes |
| Consumer | Yes | No |
| Supplier | No | Yes |
| Predicate | Yes | Yes (true / false) |

Ref: FourTypesFunctionalInterfaces.java

## Functional Composition

* Combine small functional units into bigger, more complex tasks
* There are 2 ways to address the functional composition,
  + compose
  + andThen



**Method Signature Type chain**

Function<V, R> compose(Function<V, T>) V → T → R

Function<T, V> andThen(Function<R, V>) T → R → V

Ref: FunctionalComposition.java

# Immutability

[4. Immutability | A Functional Approach to Java](https://learning.oreilly.com/library/view/a-functional-approach/9781098109912/ch04.html#id45)

## 3 ways of immutable collection

1. Unmodifiable collection
2. Immutable collection factory method (Java 9+)
3. Immutable copies (Java 10+)

Ref: ImmutableList.java

### Unmodifiable collection

List<String> unmodifiable = Collections.unmodifiableList(original);

A diagram of a memory layout

Description automatically generated

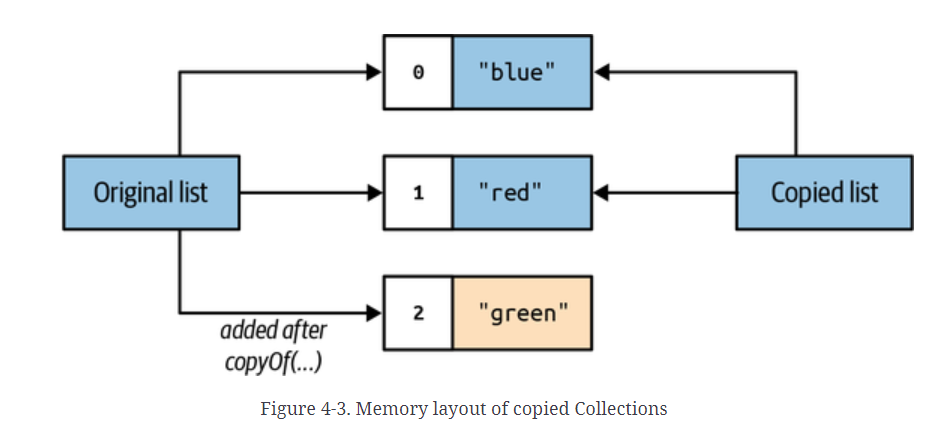
The common use for unmodifiable views is to freeze collections for unwanted changes before using them as a return value

### Immutable collection factory methods

* This is available from Java 9+
* Not based on pre-existing collection
* List<Integer> numbers = List.of(1, 2, 3);

### Immutable copies

* Available from Java 10+
* Provides a deeper level of immutability
* Create a new list from the original list
* List<Integer> unmodifiable = List.copyOf(numbers);

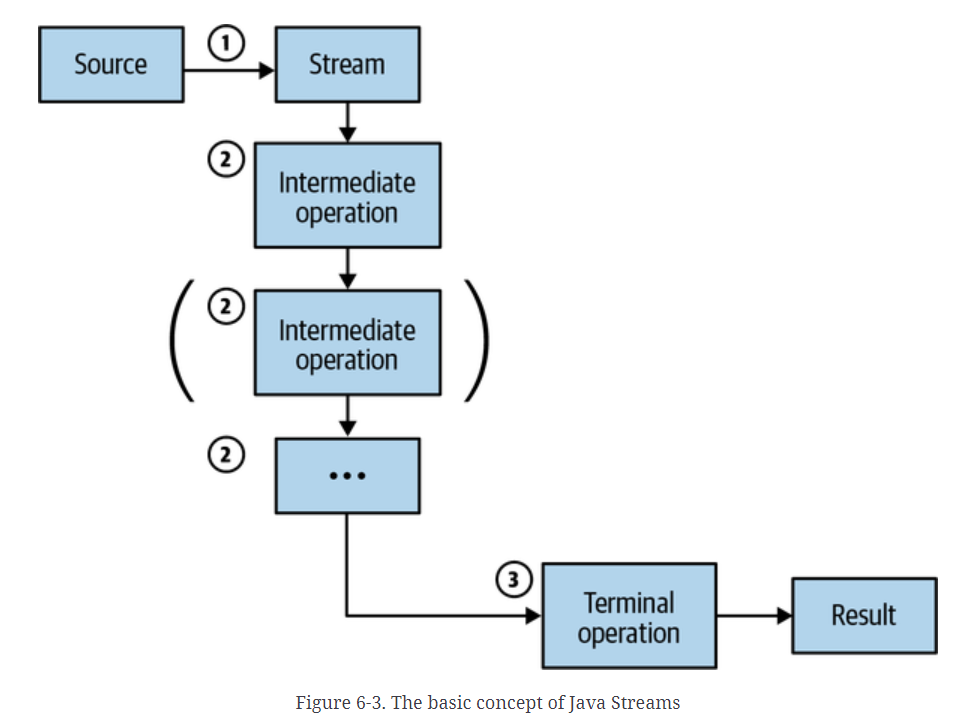


## Other immutable options

* Immutable math
* Java Time API
* Enum
* The final keyword
* Record

# Streams

## The basics of Java Streams



|  |  |
| --- | --- |
| Creating a stream | * Creating a stream from the data source |
| Doing the work | * Intermediate operations * Higher order functions available as part of java.util.stream.Stream<T> * Filtering, mapping, sorting … |
| Getting a result | * Terminal operation is required to get the result |

* Stream focuses on ‘What’ is happening in a more expressive and fluent way
* The normal flow focuses mainly on the ‘How’ part

private void doStream() {  
 List<Book> filteredBooks = BooksGenerator  
 .*books*()  
 .stream()  
 .filter(book -> book.year() < 2002)  
 .filter(book -> book.genre() == Genre.*FICTION*)  
 .sorted(Comparator.*comparing*(Book::title))  
 .limit(3L)  
 .toList();  
  
 System.*out*.println("\*\*\* doStream() \*\*\*");  
 System.*out*.println(filteredBooks);  
}

