Contents

[General 2](#_Toc184903938)

[Multi module maven project 2](#_Toc184903939)

[Fork / Join framework in Java 3](#_Toc184903940)

[Reflection vs Proxies vs Method handles 4](#_Toc184903941)

[Functional 5](#_Toc184903942)

[Functional Interfaces 5](#_Toc184903943)

[Functional Composition 6](#_Toc184903944)

[Immutability 7](#_Toc184903945)

[3 ways of immutable collection 7](#_Toc184903946)

[Unmodifiable collection 7](#_Toc184903947)

# General

## 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

## Fork / Join framework in Java

<Real work example should be prepared>

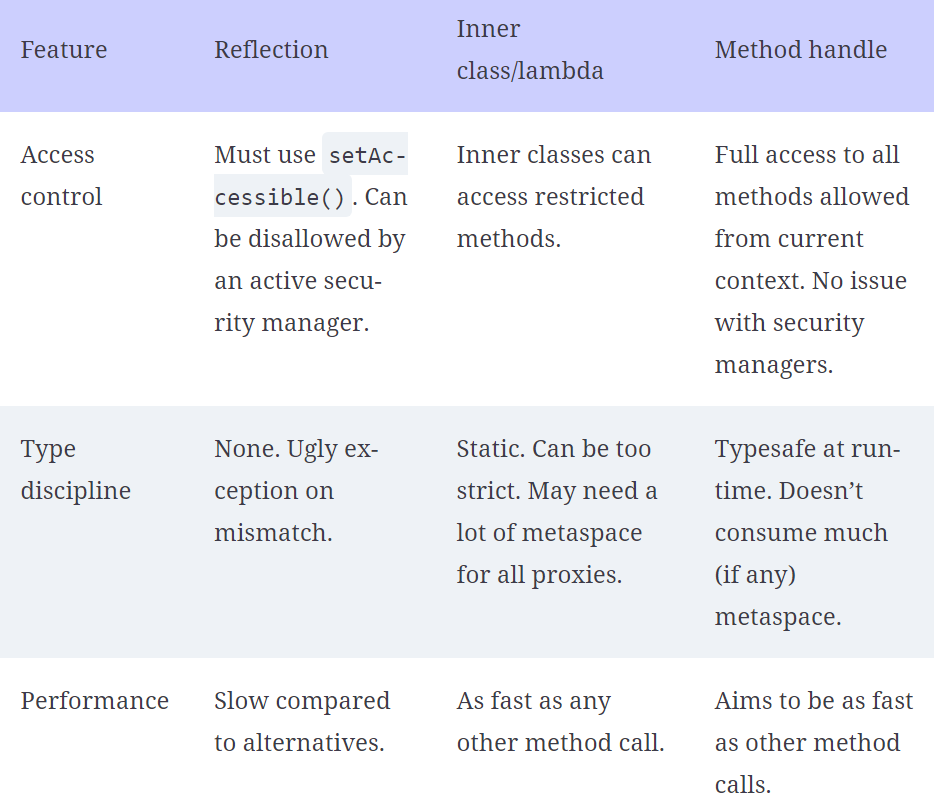
Ref: [Guide to the Fork/Join Framework in Java | Baeldung](https://www.baeldung.com/java-fork-join)

[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

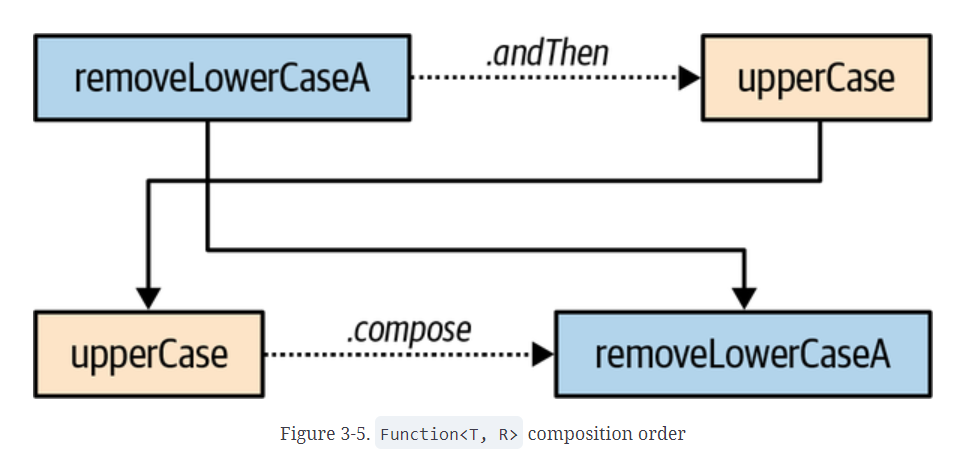
|  |  |
| --- | --- |
| **Functions**  Accepts arguments and return results | **Consumers**  Only accept arguments but do not return results |
| **Suppliers**  Do not accept arguments and only return results | **Predicates**  Accept arguments to test against an expression and return a Boolean primitive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