Anleitung zum Ausprobieren der Beispielprogramme / Musterlösungen des Buchs „Java – Die Neuerungen in Version 17 LTS, 18 und 19“

Voraussetzungen

Java-Installationen

Sie benötigen zumindest ein installiertes JDK 17. Allerdings ist es ratsam, ein aktuelles JDK 18 zu verwenden. Unabdingbar ist das für spezielle Dinge aus modernem Java.

- https://www.oracle.com/java/technologies/javase-downloads.html

Java 19 ist derzeit (Stand August 2022) noch nicht offizielle verfügbar, sondern nur als Early-Access-Build:

- https://jdk.java.net/19/

IDEs

Zur Arbeit mit bzw. zum Import des Projekts in Eclipse empfiehlt sich eine aktuelle Version 2022-06 oder ein aktuelles IntelliJ 2022.2.x:

- https://eclipse.org/downloads/
- https://www.jetbrains.com/de-de/idea/download

Mitgelieferte Projekte

Folgende Projekte werden mitgeliefert

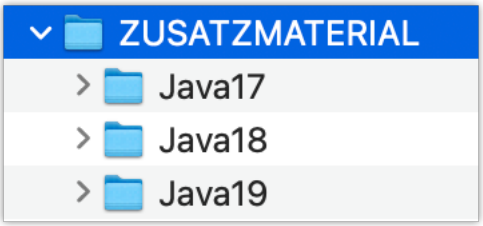
- **Java 17**
  Voraussetzung: Java 17 installiert
  Beispiele: Als Projekt zum Import, sowie als eigenständige Projekte:
  - `directompilation`
  - `JMH-Benchmarking`
  - `JAXBExample_JDK11`
  - `JAXBExample_JDK8`
  - `PackagingDemo`
  - `PackagingDemoWithGuava`

- **Java 18**
  Voraussetzung: Java 18 installiert
  Beispiele: Als Projekt zum Import

- **Java 19**
  Voraussetzung: Java 19 (Early-Access) installiert
  Beispiele: Es werden einzelne Java-Dateien geliefert.

Übungsaufgaben / Beispielprogramme „installieren“

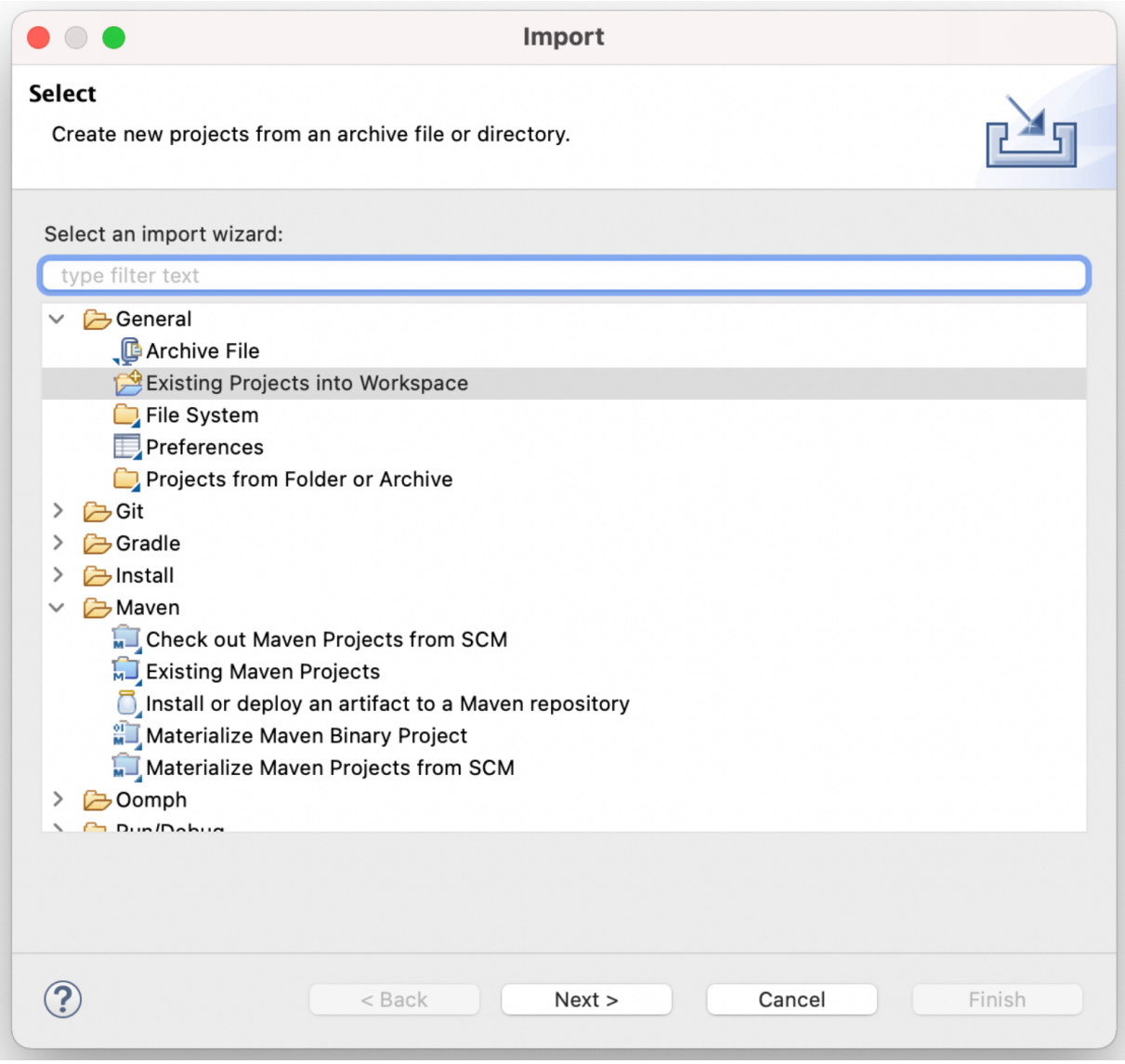
Laden Sie das Projekt mit den Übungsaufgaben und Beispielen als ZIP-Archiv herunter und entpacken Sie dieses in ein beliebiges Verzeichnis.



Danach finden Sie in den jeweiligen Ordnern die relevanten Dateien, zum Teil auf ZIP-Dateien, die dann nochmals entpackt werden müssen, wie es im jeweiligen Abschnitt beschrieben ist.

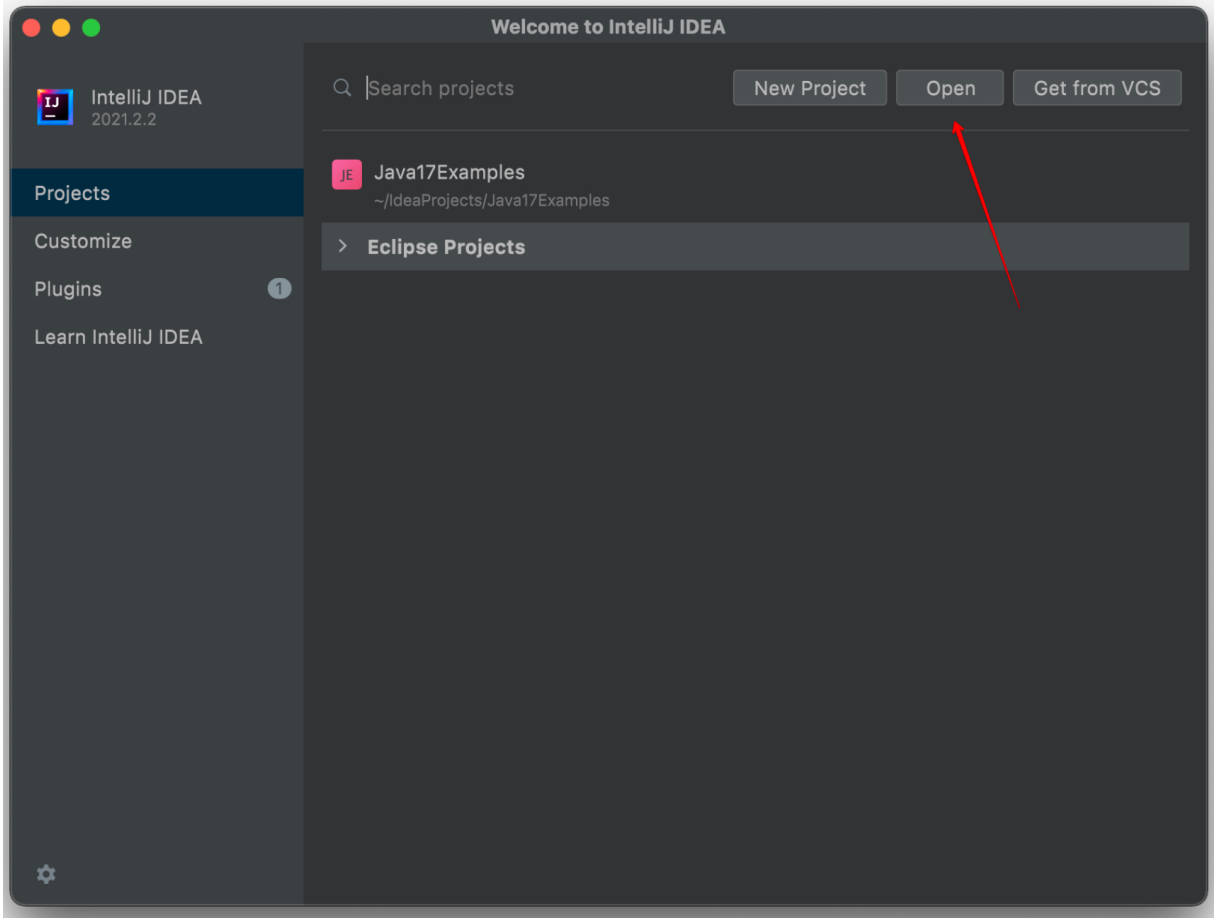
Import in Eclipse

Importieren Sie die jeweiligen Projekte in Eclipse (oder Ihre bevorzugte IDE). Dazu öffnen Sie in der Baumdarstellung links (Package Explorer) das Kontextmenü und wählen dort „Import…“. Dann erscheint ein Fenster analog zu folgendem. Dort wählen Sie „Existing Projects into Workspace“ und navigieren zu dem Speicherort Ihres heruntergeladenen Projekts.

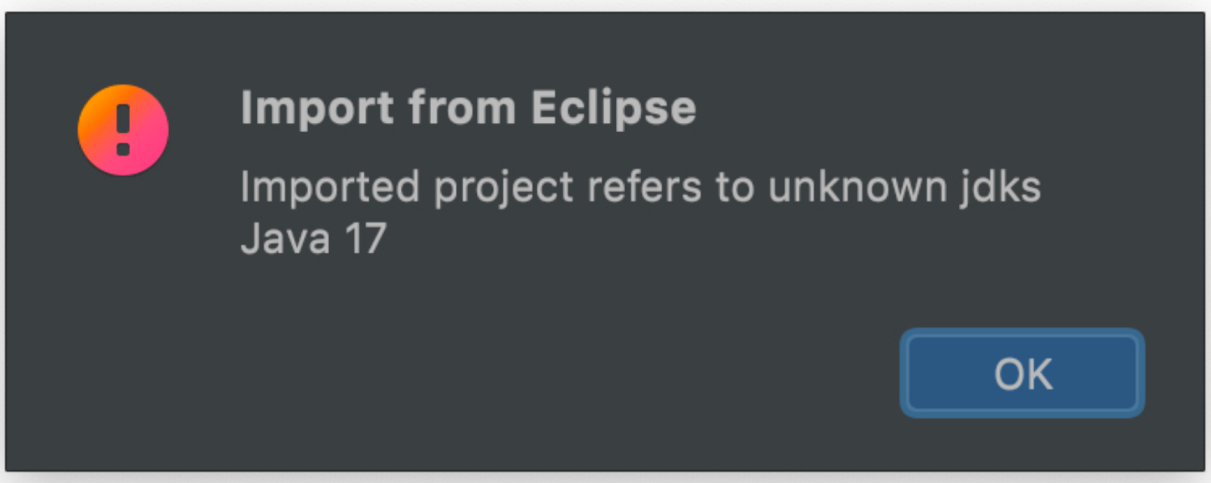


Projektimport mit IntelliJ

Als Alternative zu Eclipse können Sie auch IntelliJ verwenden. Dabei ist es möglich, Eclipse-Projekte oder als Maven bzw. Gradle zu importieren. Dazu klicken Sie auf „Open“.



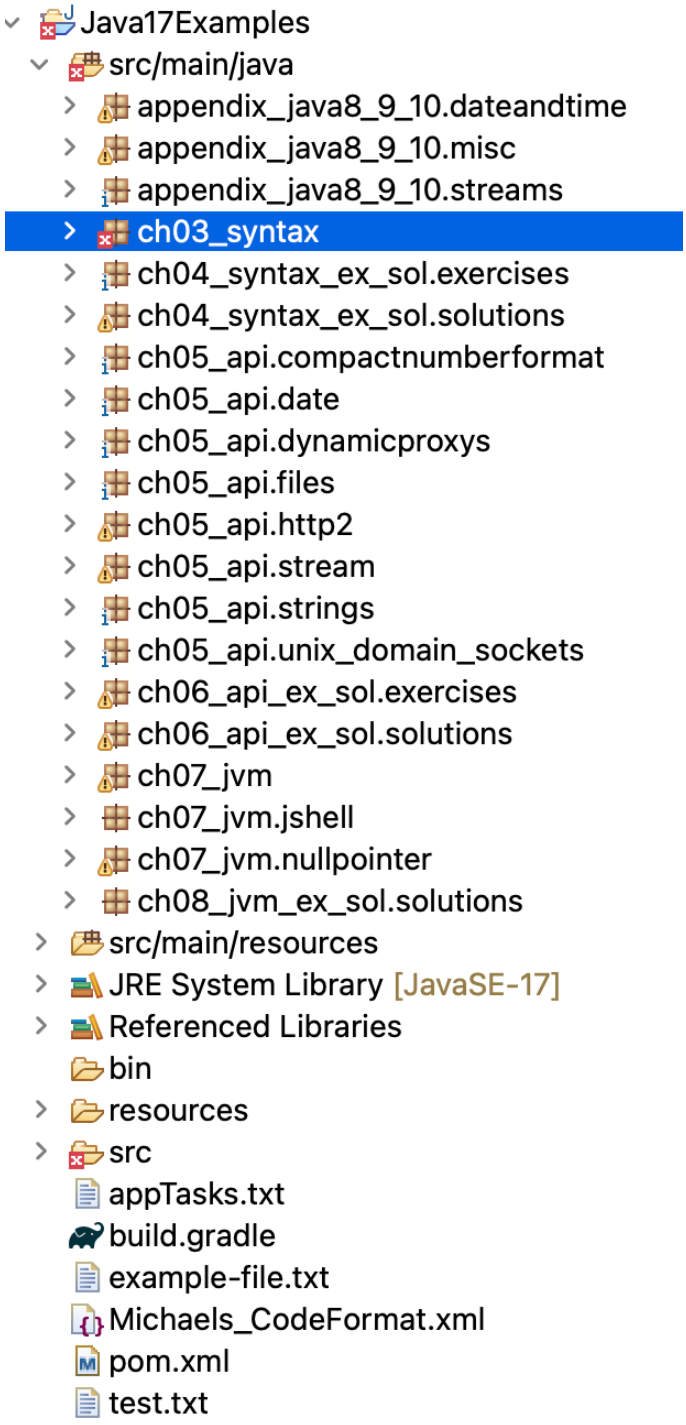
Eventuell erscheint folgende Meldung, die Sie einfach bestätigen:



Zudem wird vermutlich auf den Einsatz von experimentellen Features hingewiesen — auch das ist in Ordnung. Bitte scrollen Sie bis Sie den blauen Text Accept sehen und anklicken.

Java 17

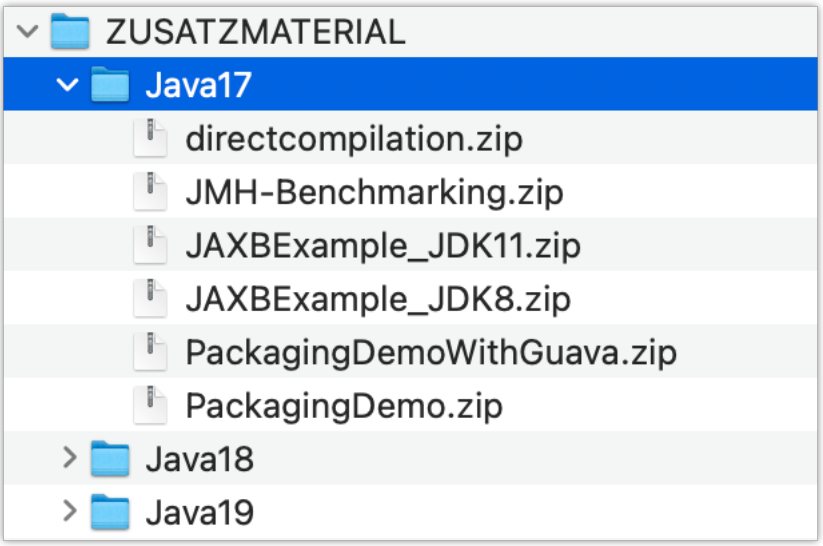
Importieren Sie das Projekt **`Java17Examples`** aus dem Verzeichnis **`Java17`** am einfachsten als Maven-Projekt in Eclipse oder IntelliJ.



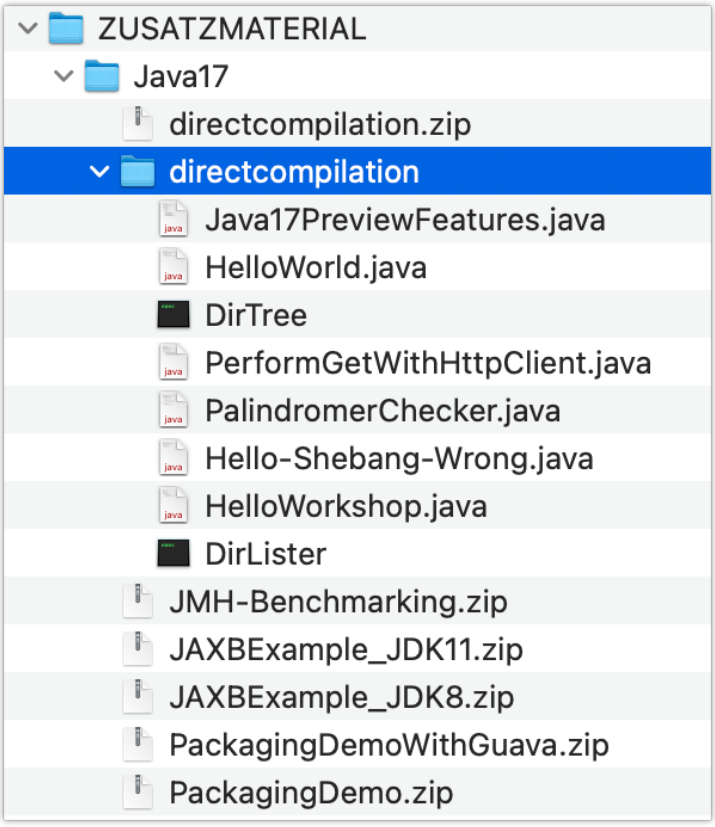
HINWEIS:
Aktuellere Versionen von Eclipse erlauben es nur für die aktuelle Java-Version die Preview-Features zu aktivieren, somit könnte es dann bei einigen wenigen Dateien, die eben diese Preview-Features verwenden, zu Kompilierfehlern kommen. Ein Build mit Maven oder Gradle läuft aber problemlos.

Ergänzende Projekte

Entzippen Sie die jeweiligen ZIP-Dateien im Ordner **`Java17`**, um die korrespondierenden Beispielprojekte oder Dateien bereitzustellen.



Wechseln Sie dann in das jeweilige Verzeichnis, etwa `cd directompilation` und folgen den Anweisungen im Buch.



Bei diversen Beispielen empfiehlt sich ein Maven-Import in Eclipse oder IntelliJ.

Direct Compilation (Kapitel 7.3)

Führen Sie die Aufrufe auf der Konsole aus, wie im Buch beschrieben.

JMH-Benchmarking (Kapitel 7.4)

Nach dem Projekt-Import können Sie verschiedene Benchmark-Klassen aus dem Verzeichnis `benchmark-templates` in den `src`-Ordner verschieben und dann die jeweiligen Benchmarks starten.

JPackage (Kapitel 7.5.2)

Nach dem Projekt-Import können Sie die Sourcen mit Maven oder Gradle bauen und dann `jpackage` auf der Konsole wie im Buch gezeigt ausführen.

JPackage mit Guava (Kapitel 7.5.3)

Nach dem Projekt-Import können Sie die Sourcen mit Maven oder Gradle bauen und dann `jpackage` auf der Konsole wie im Buch gezeigt ausführen.

JAXB Java 8 (Kapitel 7.5.5)

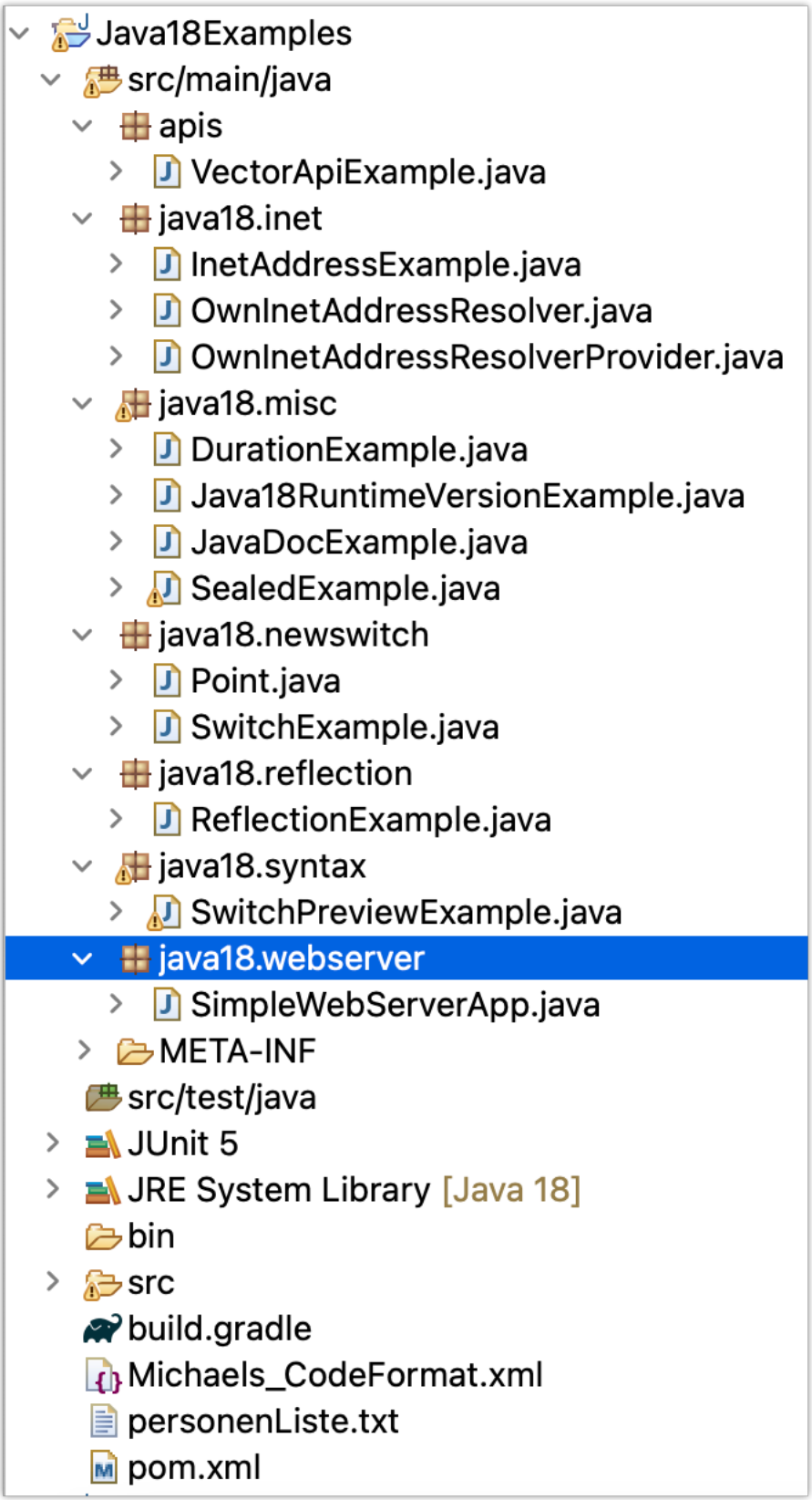
Dieses Projekt benötigt Java 8! Nach dem Projekt-Import in Ihre IDE können Sie die Klassen normal starten.

JAXB Java 11 (Kapitel 7.5.5)

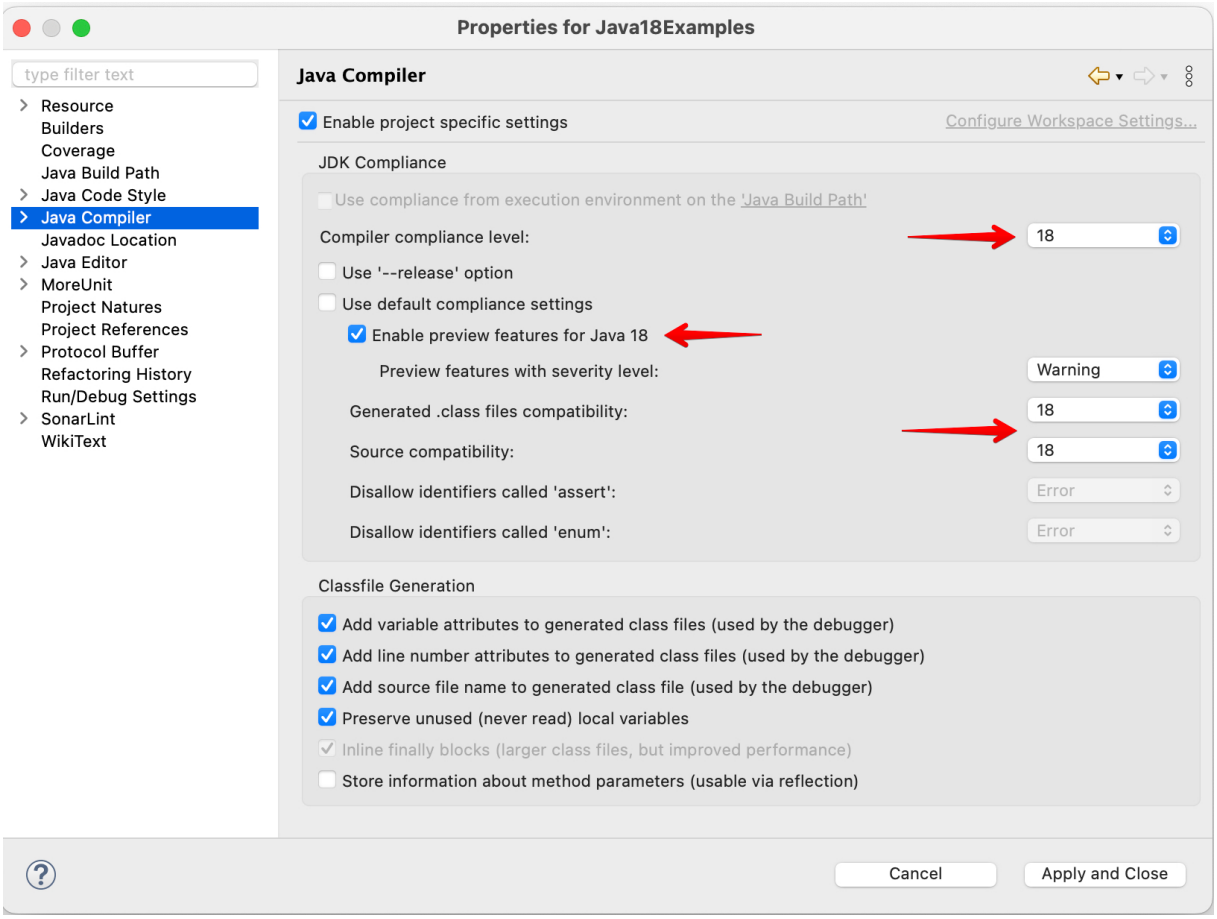
Für dieses Projekt werden externe Abhängigkeiten benötigt. Daher können Sie wieder einen Maven-Import nutzen und dann die Sourcen bauen. Danach sollte sich das Programm starten lassen.

Java 18

Importieren Sie das Projekt **`Java18Examples`** aus dem Verzeichnis **`Java18`** am einfachsten als Maven-Projekt in Eclipse oder IntelliJ:



Damit alles sauber kompiliert aktivieren Sie bitte auch die Preview-Features sowie die korrekte Java-Version und führen dann die entsprechenden Programme aus.



Besonderheiten beim Ausführen Vector-API-Beispiel

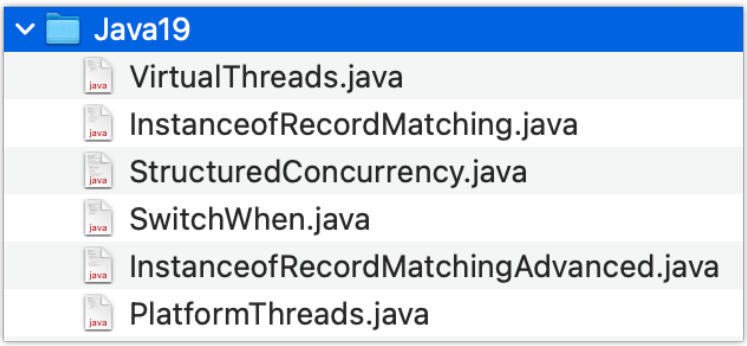
Das Beispielprogramm `VectorApiExample.java` ist im Projekt auskommentiert, da es momentan noch Problem im Maven und Gradle-Build mit dem Einbinden von Incubator gibt.

Folgende Schritte zum Ausführen sind nötig:

1. Einkommentieren der Klasse
2. Wechsel ins Verzeichnis:
   ```
   $ cd /src/main/java/apis
   ```
3. Starten mit Direct Compilation
   ```
   $ java --add-modules jdk.incubator.vector VectorApiExample.java
   WARNING: Using incubator modules: jdk.incubator.vector
   warning: using incubating module(s): jdk.incubator.vector
   1 warning
   [6, 8, 10, 12]
   ```

Java 19

Die einzelnen Beispieldateien finden Sie im Verzeichnis **`Java19`**. Öffnen Sie eine Konsole und führen Sie die Kommandos wie im Buch beschrieben aus.



Weitere Informationen

Im Buch finden Sie weitere Informationen zu den Programmen. Falls doch noch Fragen offen sein sollten, erreichen Sie mich unter: michael_inden@hotmail.com