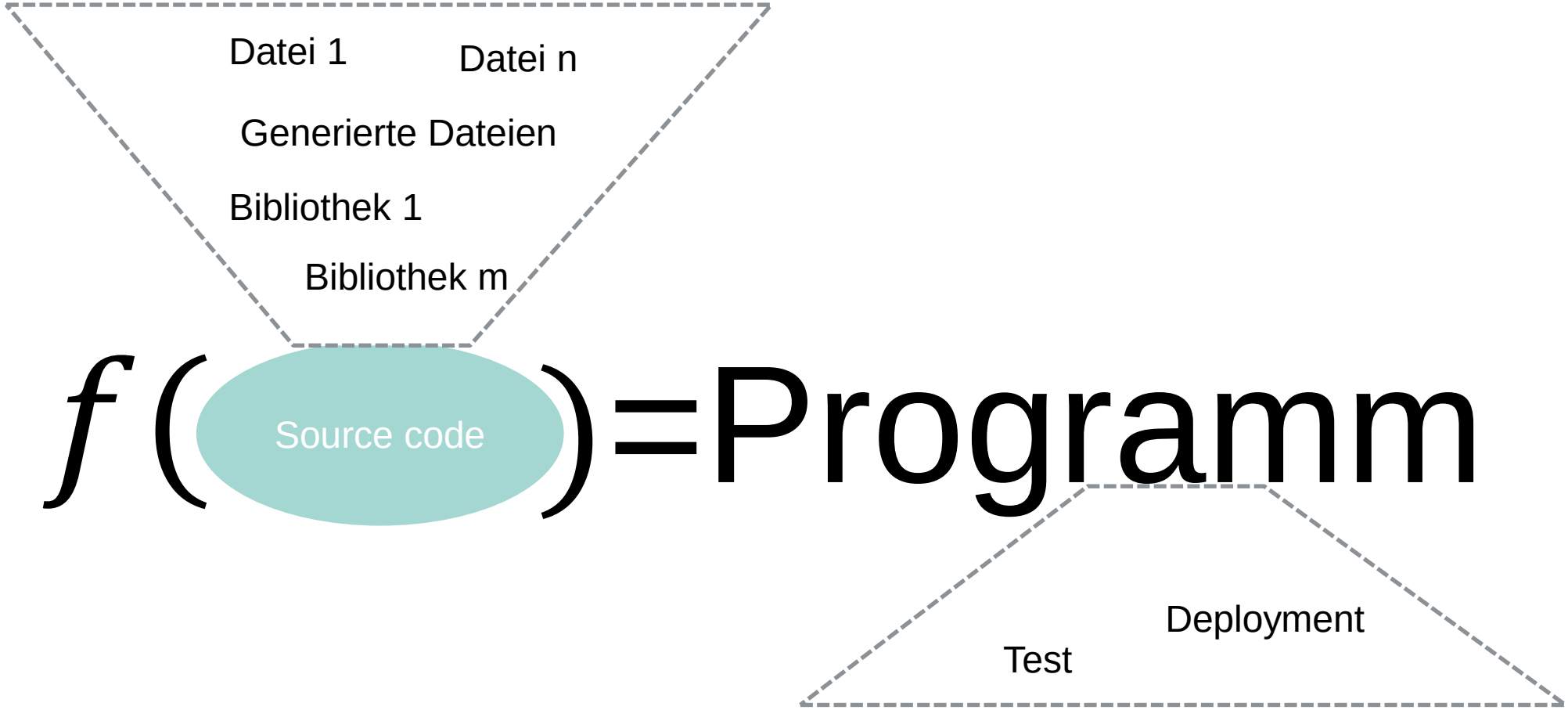
Buildsysteme

Build



Ziel: Erstellen eines Programms aus dem Sourcecode

Ansatz 1

Manuelle Aufrufe

```
> javacc Parser1.cc   // Dateien generieren
> javac -cp libs/junit-4.5.jar:libs/slf4j-2.1.jar *.java  // Kompilieren
> jar cmf my_manifest project *.class // Jar erstellen
> java -cp .:/usr/share/java/junit.jar org.junit.runner.JUnitCore MyClass  // Tests aufrufen
```

- *Mühsam*
- *Fehleranfällig*
- *Lange Buildzeiten, da jede Datei immer kompiliert wird.*
- *Undokumentiert*

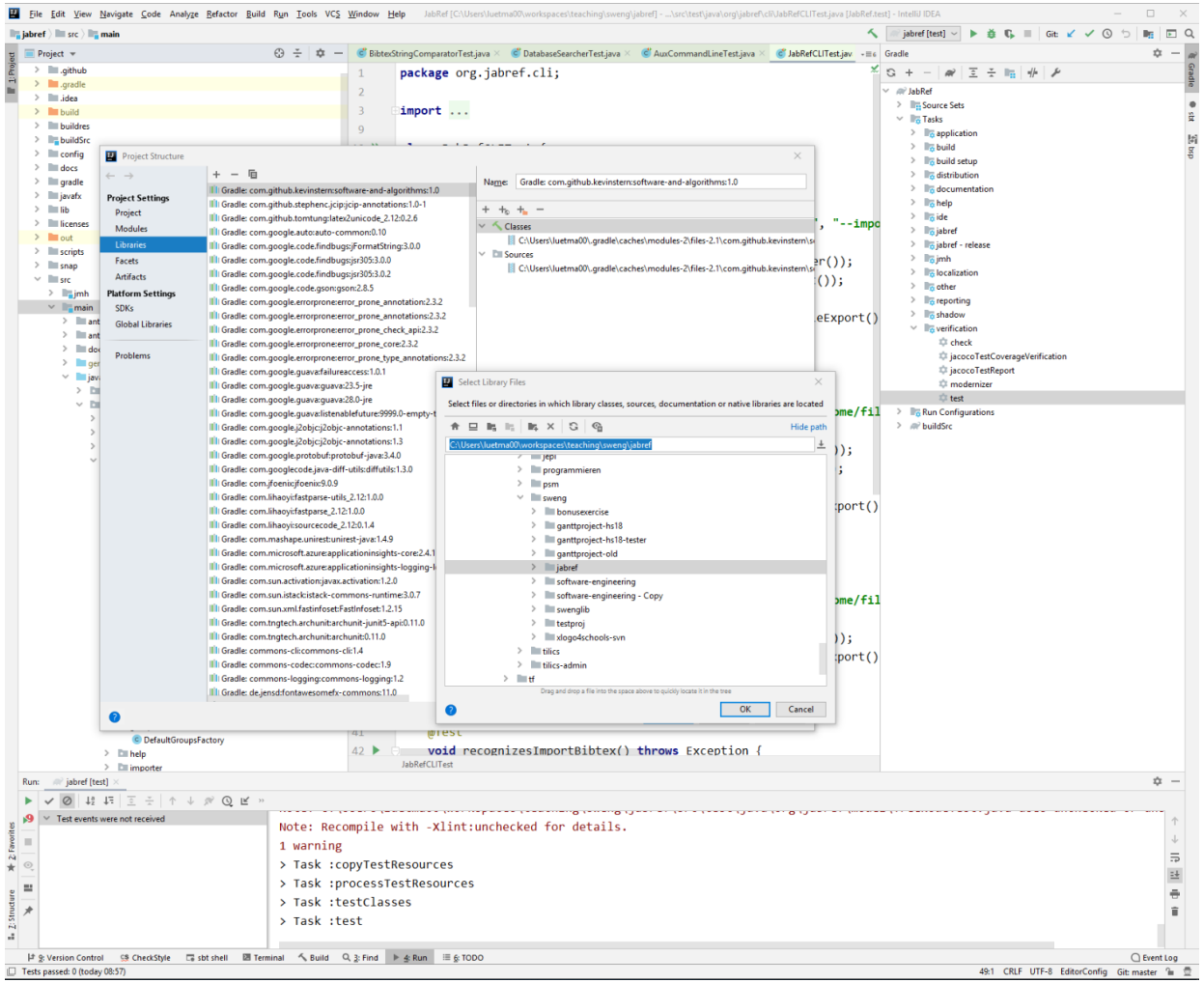
Ansatz 2: Eclipse oder IntelliJ

Moderne IDE's können

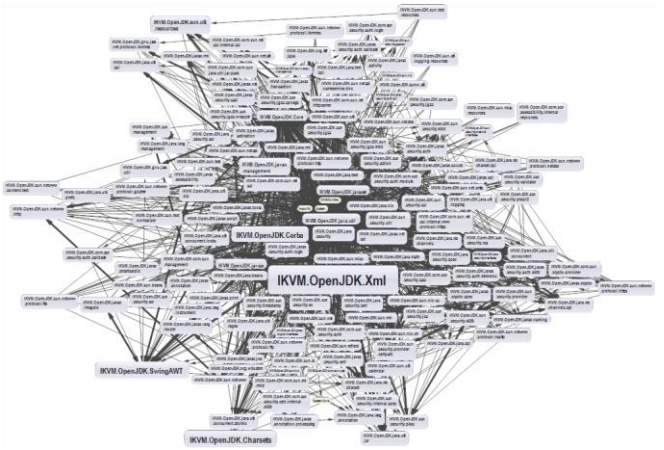
- Feststellen was kompiliert werden muss
- Bibliotheken verwalten
- Jars und Programmer erstellen
- Tests ausführen

Nachteile:

- *Schwerfällig*
- *IDE muss installiert / konfiguriert werden*
- *Schlechte Anpassbarkeit*
- *Kann nicht auf Server ausgeführt werden*



Lösung



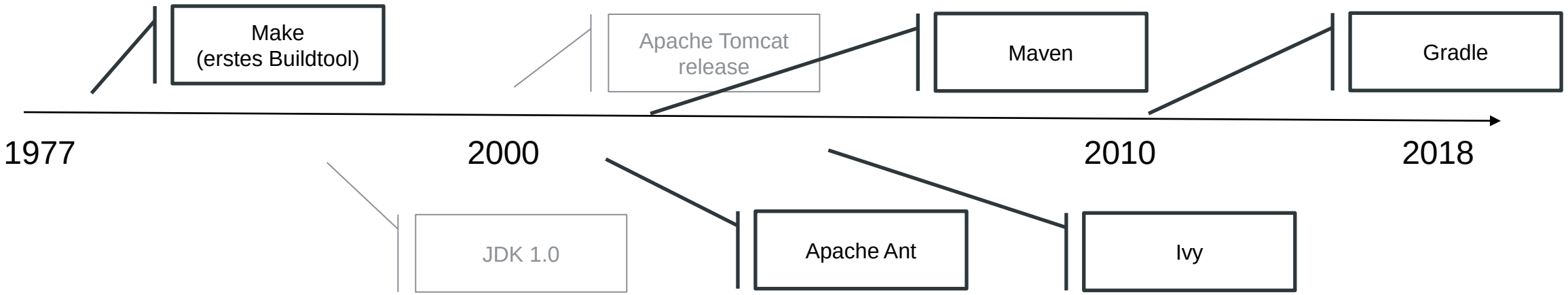
Build automation systems

- Erlaubt das Automatisieren von Build Tasks via Scripts

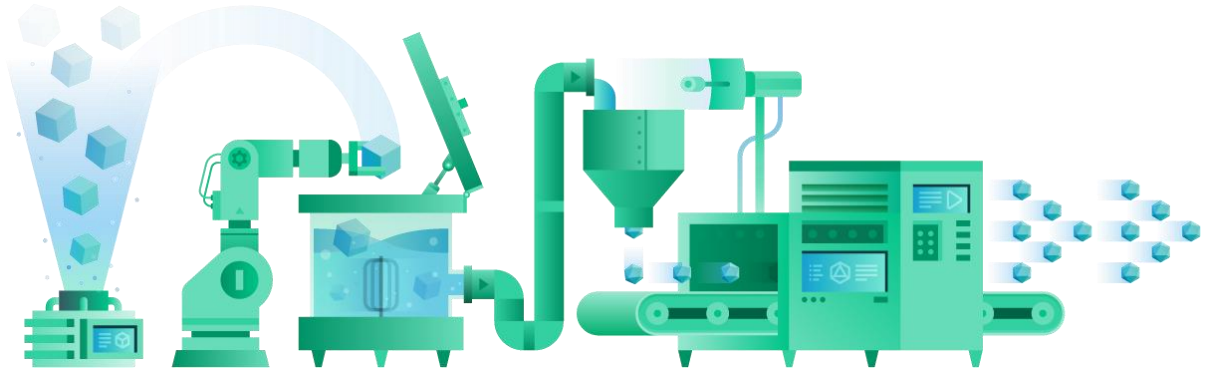
Dependency management systems

- Erledigt das Management von Bibliotheken

Build and dependency management Systeme



Gradle



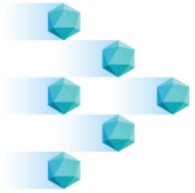
Build Anything

Write in Java, C++, Python or your language of choice. Package for deployment on any platform. Go monorepo or multi-repo. And rely on Gradle's unparalleled versatility to build it all.

Automate Everything

Use Gradle's rich API and mature ecosystem of plugins and integrations to get ambitious about automation. Model, integrate and systematize the delivery of your software from end to end.

Deliver Faster



Scale out development with elegant, blazing-fast builds. From compile avoidance to advanced caching and beyond, we pursue performance relentlessly so your team can deliver continuously.

Quelle: https://gradle.org

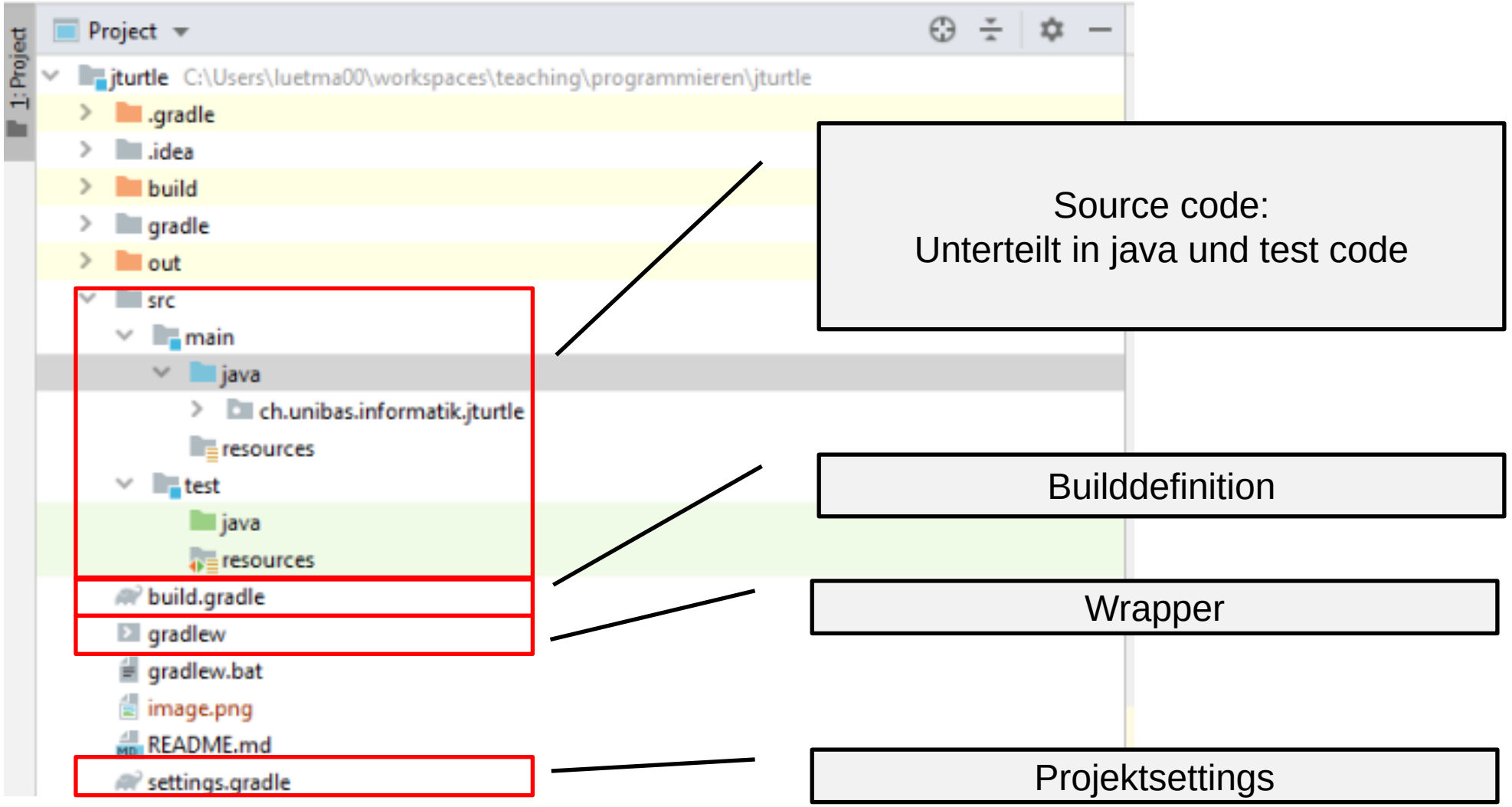
Gradle: Wichtigste Konzepte

Task:
- Einzelne Aufgaben wie kompilieren, Jar erstellen
- Tasks können voneinander abhängen

Projekt:
- Sammlung von Tasks
- Beispiele von Subprojekten: Client, Server, Dokumentation

Plugin:
- Stellt vordefinierte Tasks zur Verfügung.
- Beispiel: Java Plugin für Java-spezifische Tasks

Ordnerstruktur



Build.gradle: Builddefinition

```
plugins {
    id "java"
    id "com.jfrog.bintray" version "1.8.4"
}

apply plugin: 'java‘
```

```
repositories {
jcenter()
}


dependencies {
compile 'org.slf4j:slf4j-api:1.7.12‘
testCompile 'junit:junit:4.12‘
}
```

```
task version {
   doLast {
      println ‘This is version xxx'
   }
}
```

Benutzte Plugins
- Definiert Tasks wie z.B. javaCompile, test, …

Dependency management
- Sucht dependencies sl4fj und junit auf jCenter

Zusätzliche benutzerdefinierte Tasks

Gradle Wrapper

Problem
- Benutzter muss gradle in der richtigen Version installiert haben

Lösung: Gradle Wrapper (gradlew)
- Gradle wird als Teil des Projekts in der richtigen Version mitgeliefert
- Implementation: Batchdatei/Shellscript welches Gradle herunterlädt und im Projekt installiert

Erlaubt automatisierte, reproduzierbare builds auch auf Servern.

Praktische Übung 3

Praktische Übung: Gradle und Code-Reading