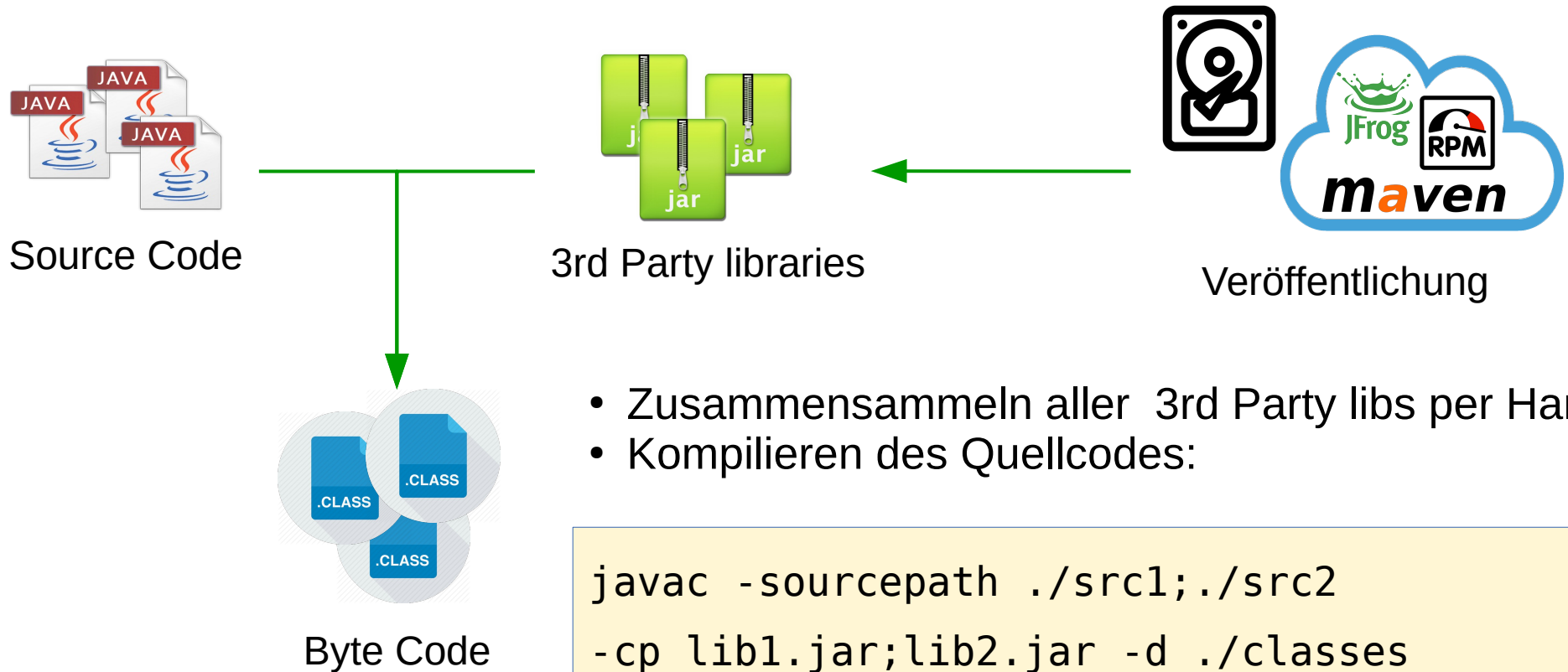




gradle
Build Management Tool

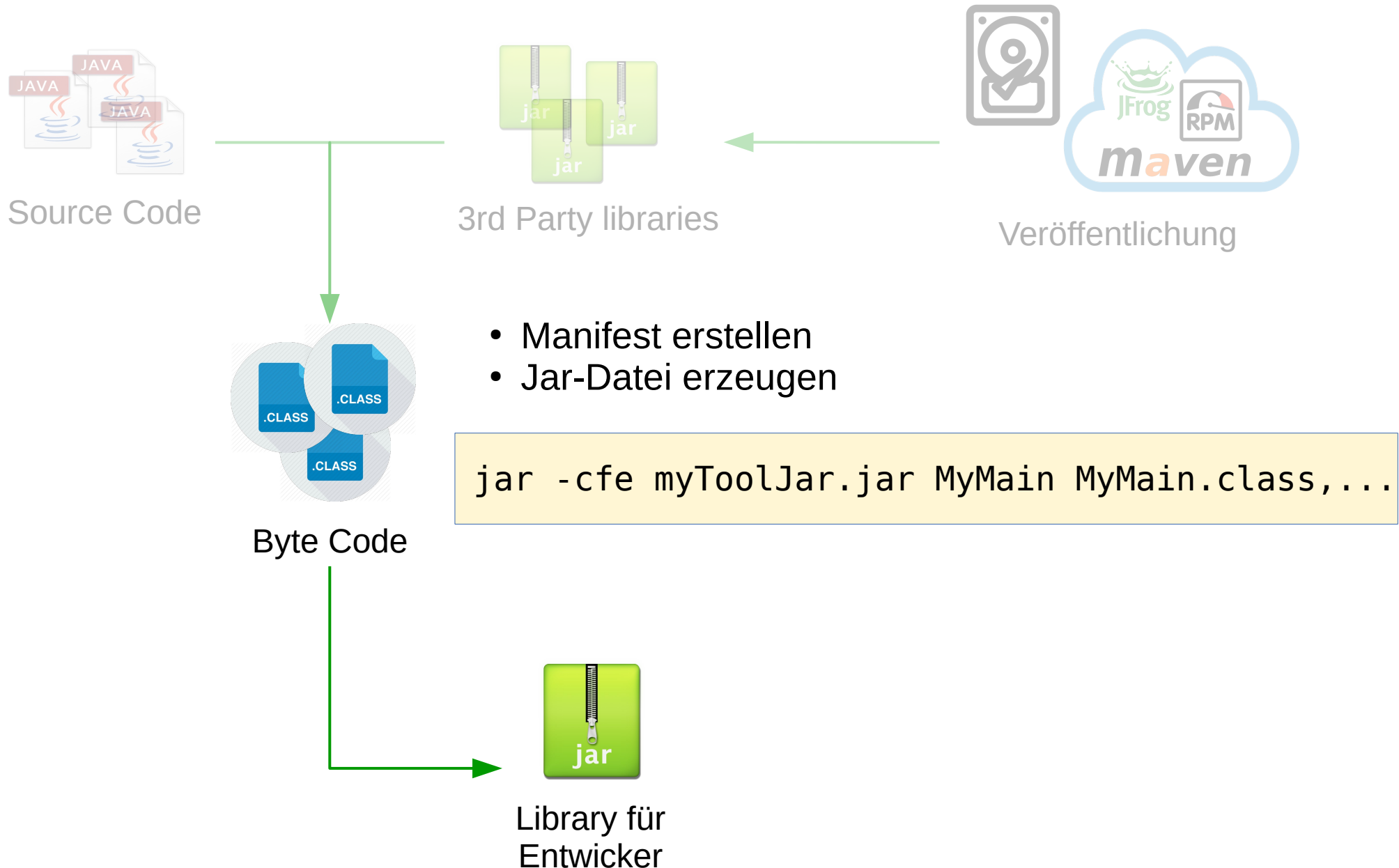
Build-Prozess - Kompilieren + Dependencies



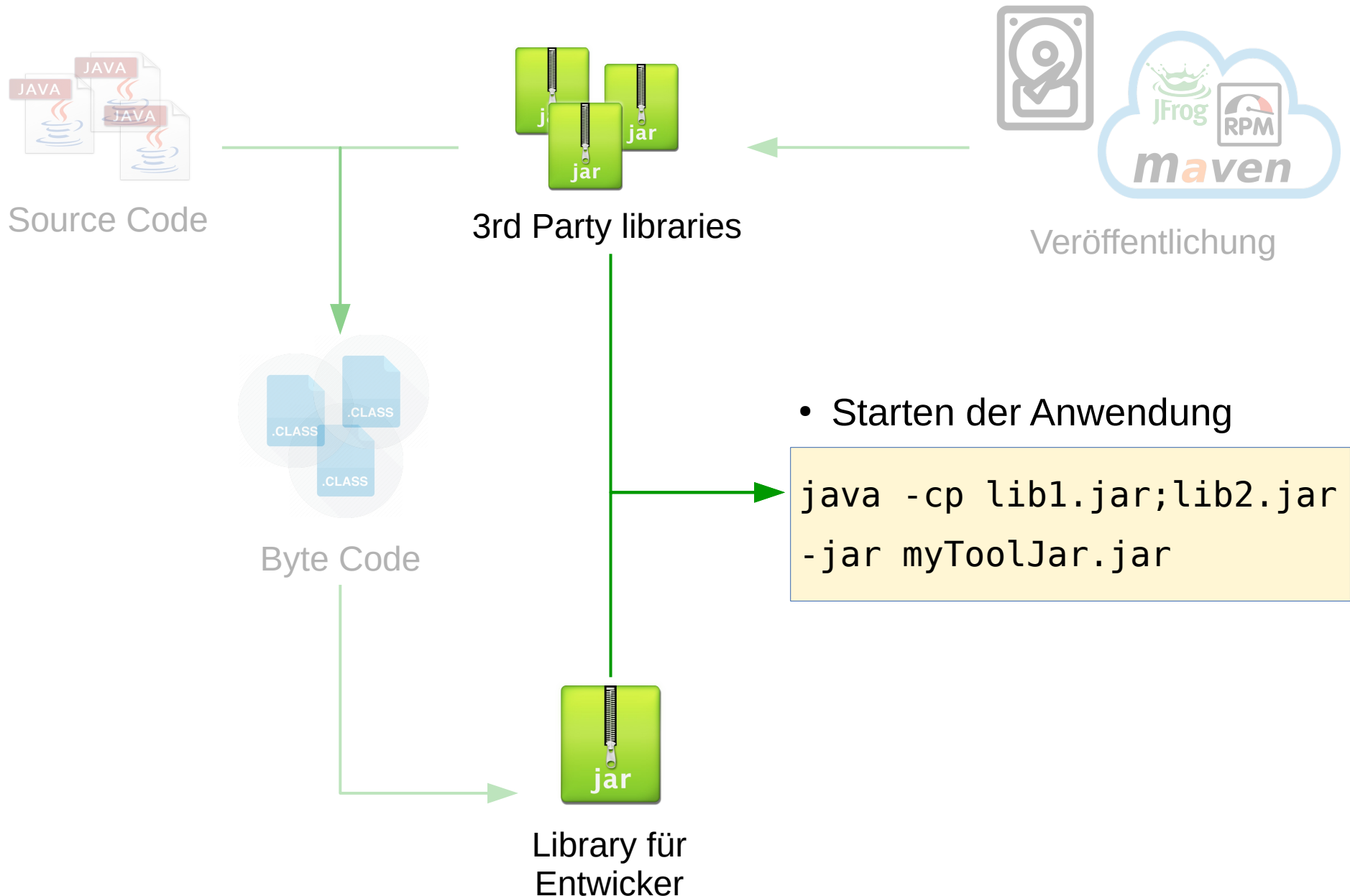
- Zusammensammeln aller 3rd Party libs per Hand
- Kompilieren des Quellcodes:

```
javac -sourcepath ./src1;./src2  
-cp lib1.jar;lib2.jar -d ./classes  
MyMain.java
```

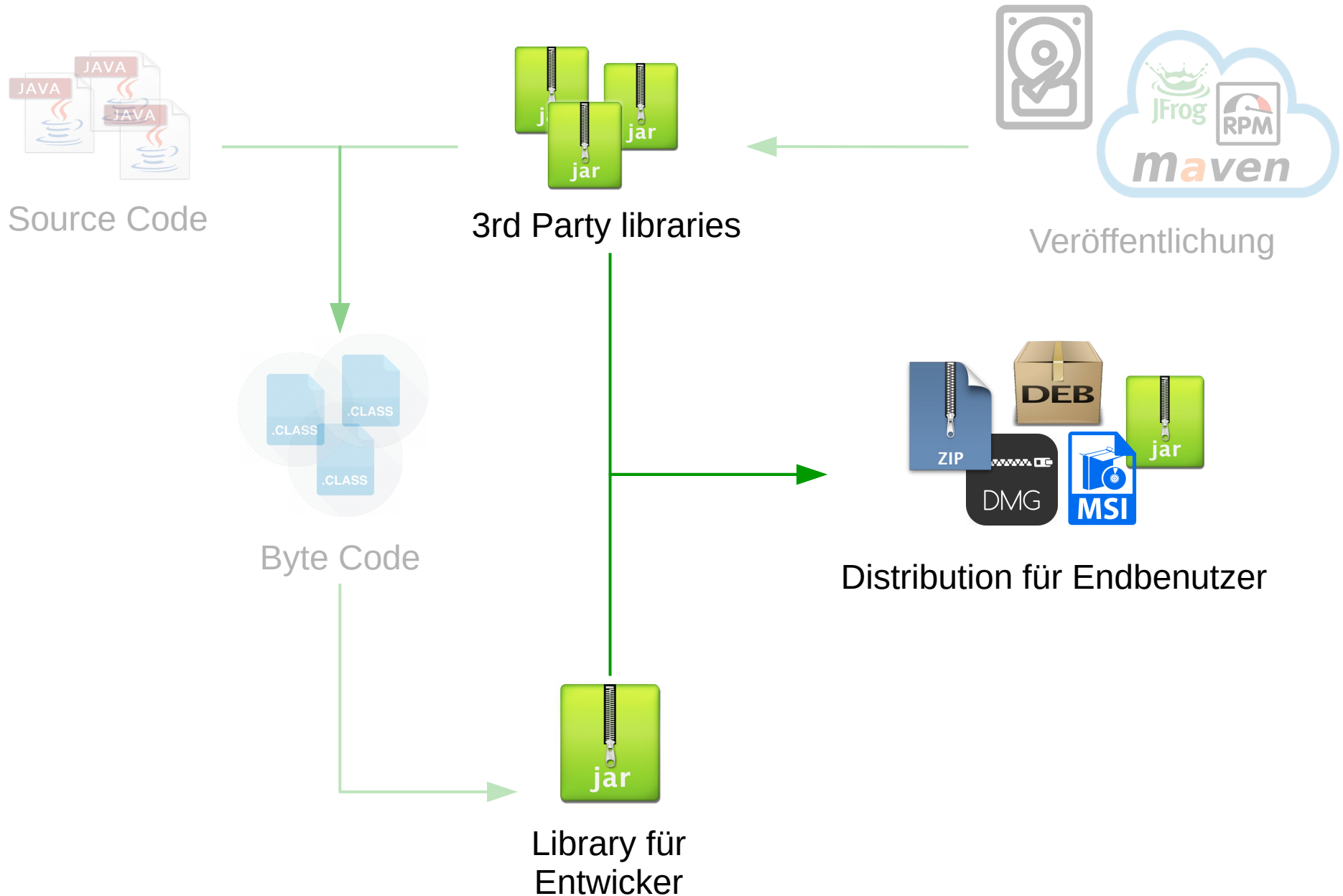
Build-Prozess - ausführbares Programm?



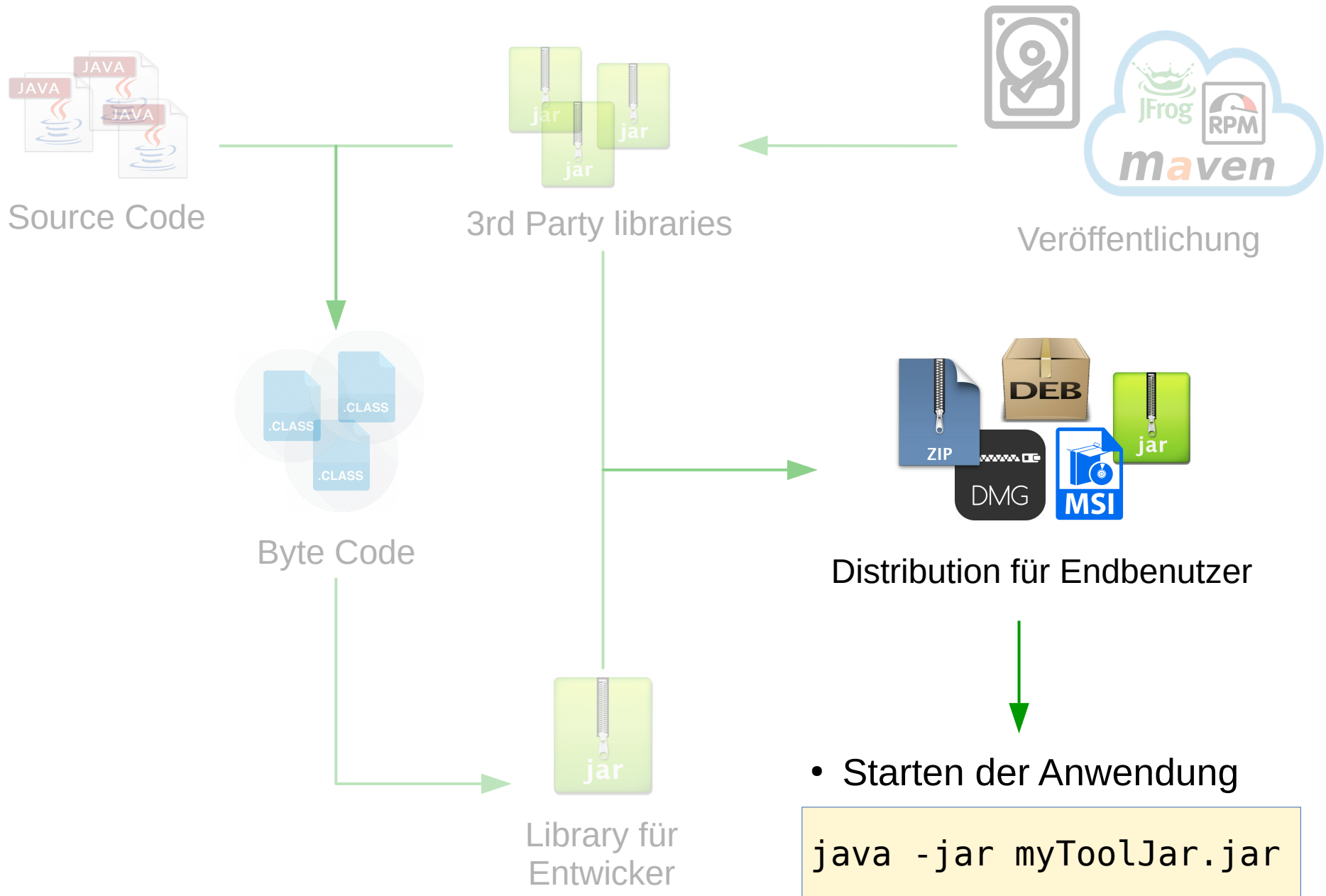
Build-Prozess - Programm ausführen (User!)



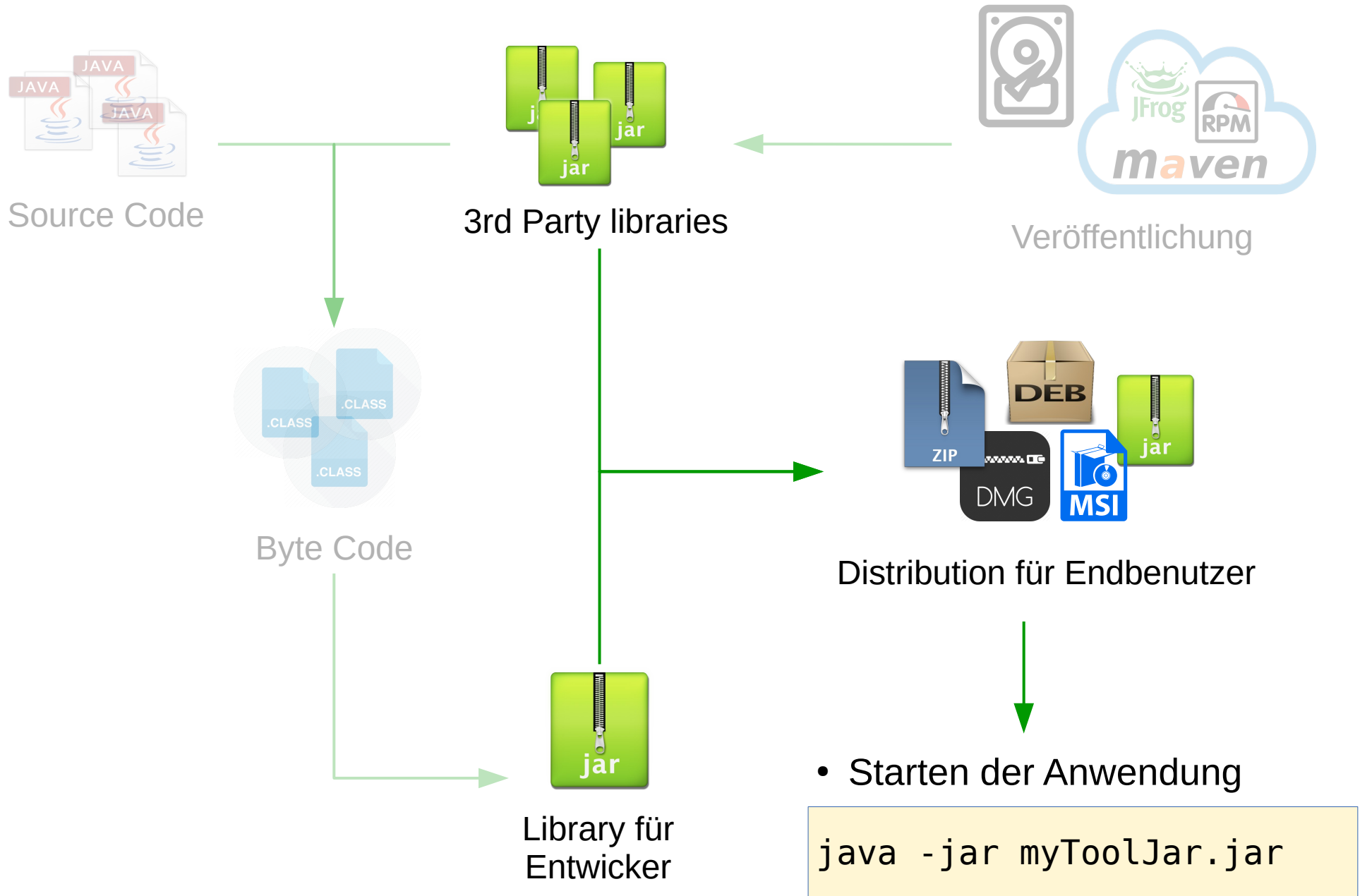
Build-Prozess - Starter / Installer



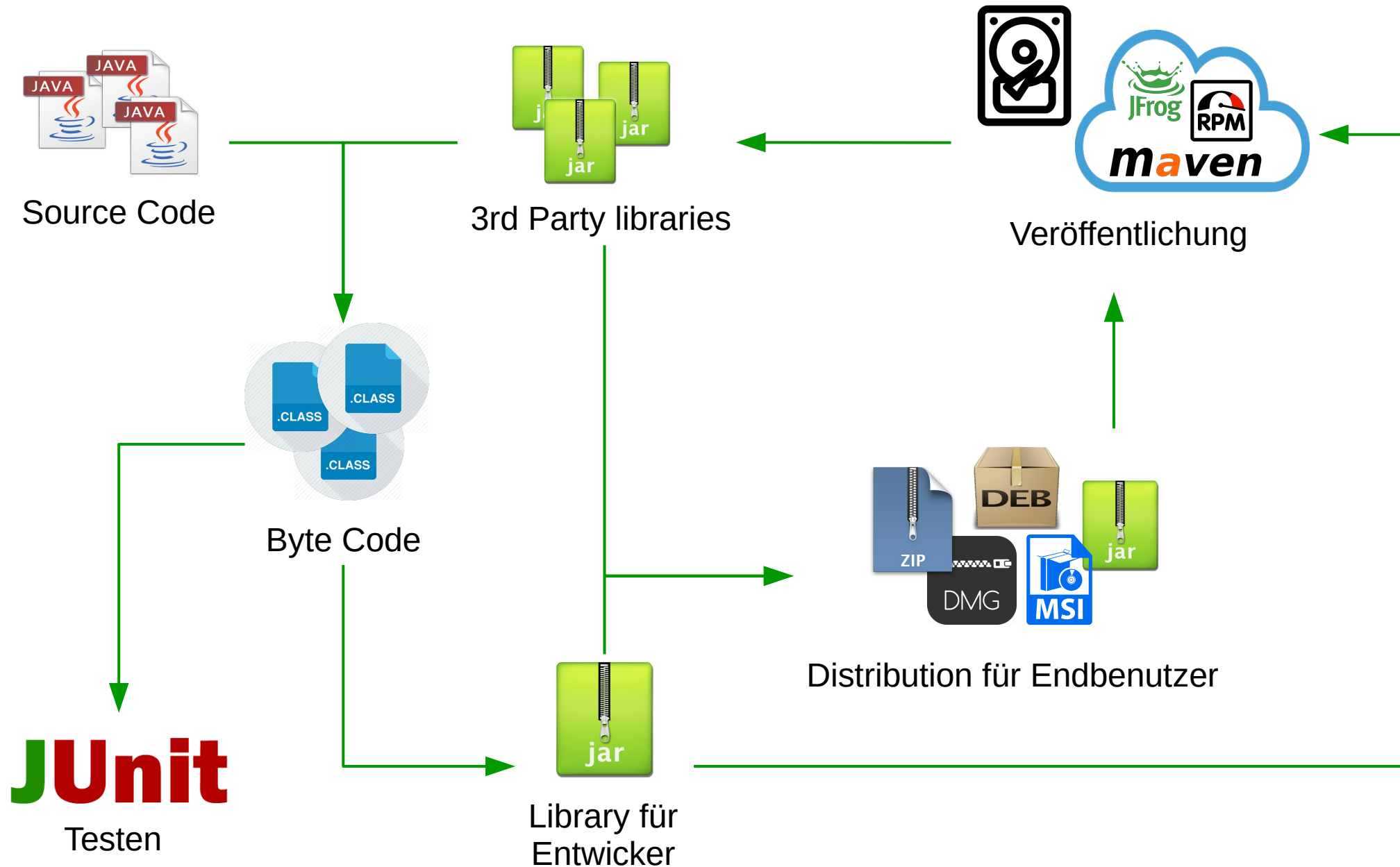
Build-Prozess - Programm ausführen (User!)



Build-Prozess - Verteilen/Veröffentlichen?



Build-Prozess - Lebenszyklus



Begriffe:

Repository – Source Code

- Zentraler Ort (Server) der Sourcecode (mit Versionskontrolle) bereitstellt. (z.B. Gitlab, Github etc.)

Repository / Artifactory – Byte Code / Binaries

- Zentraler Ort (Server) der Libraries (jars) bereitstellt. (z.B. Artifactory, MavenCentral, .m2)

Artifact (deploy, publish)

- Einheit/Archiv/Library/Jar die (anderen Entwicklern) als dependency zur Verfügung steht.

Distribution (deploy, publish)

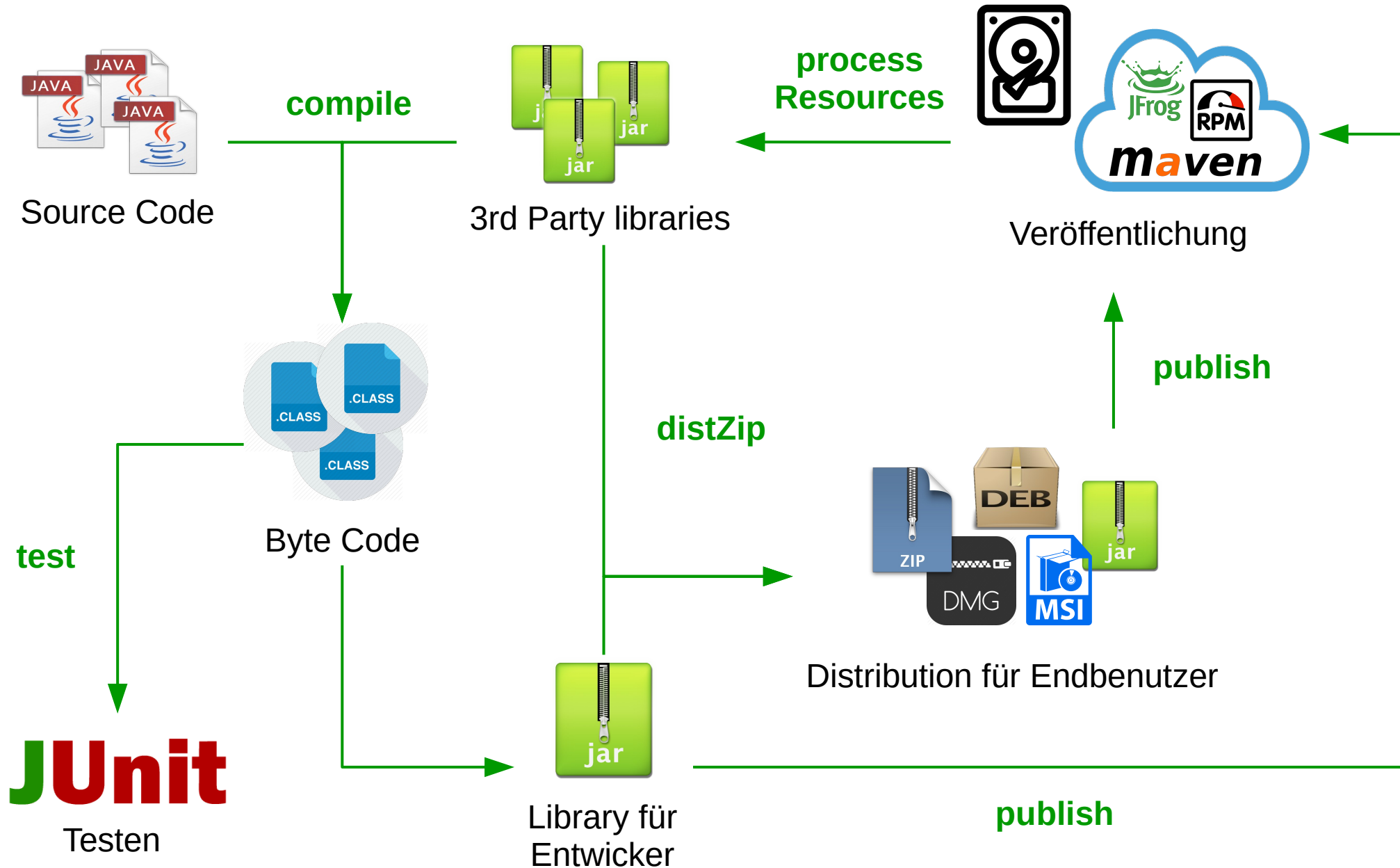
- Paket das an User ausgeliefert wird
- Starter, Manual, Libraries, Installer, Lizenzen

Build Management Tool?

Aufgaben

- Dependency Management
 - Verwaltet und lädt 3rd-Party libraries
- Kompiliert den Sourcecode
- Führt automatisierte Tests aus
- Installiert Software oder Lädt sie in Repositories
- Erstellt eine Dokumentation
- Erstellt Software Pakete (Releases/Distributionen)

Build-Prozess - Automatisieren

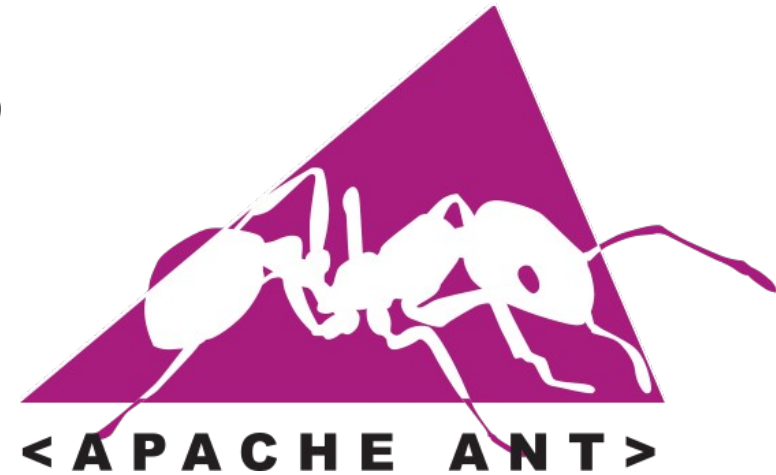


Build Management Tool?

Vertreter

- Make

maven

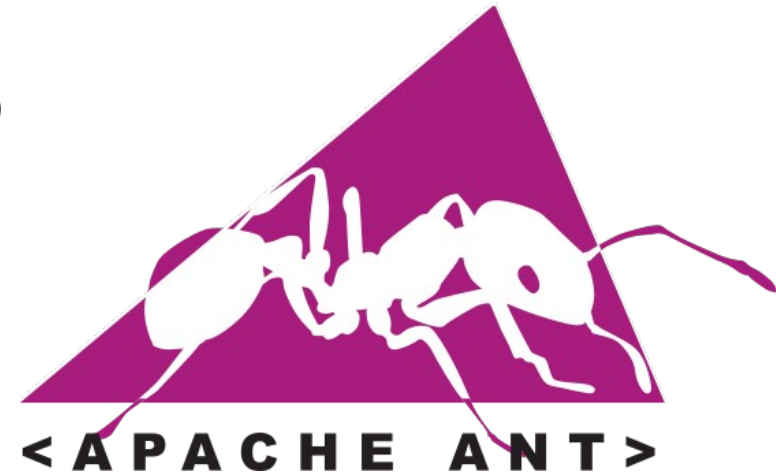


Build Management Tool?

Vertreter

- Make

maven



Und die IDE?

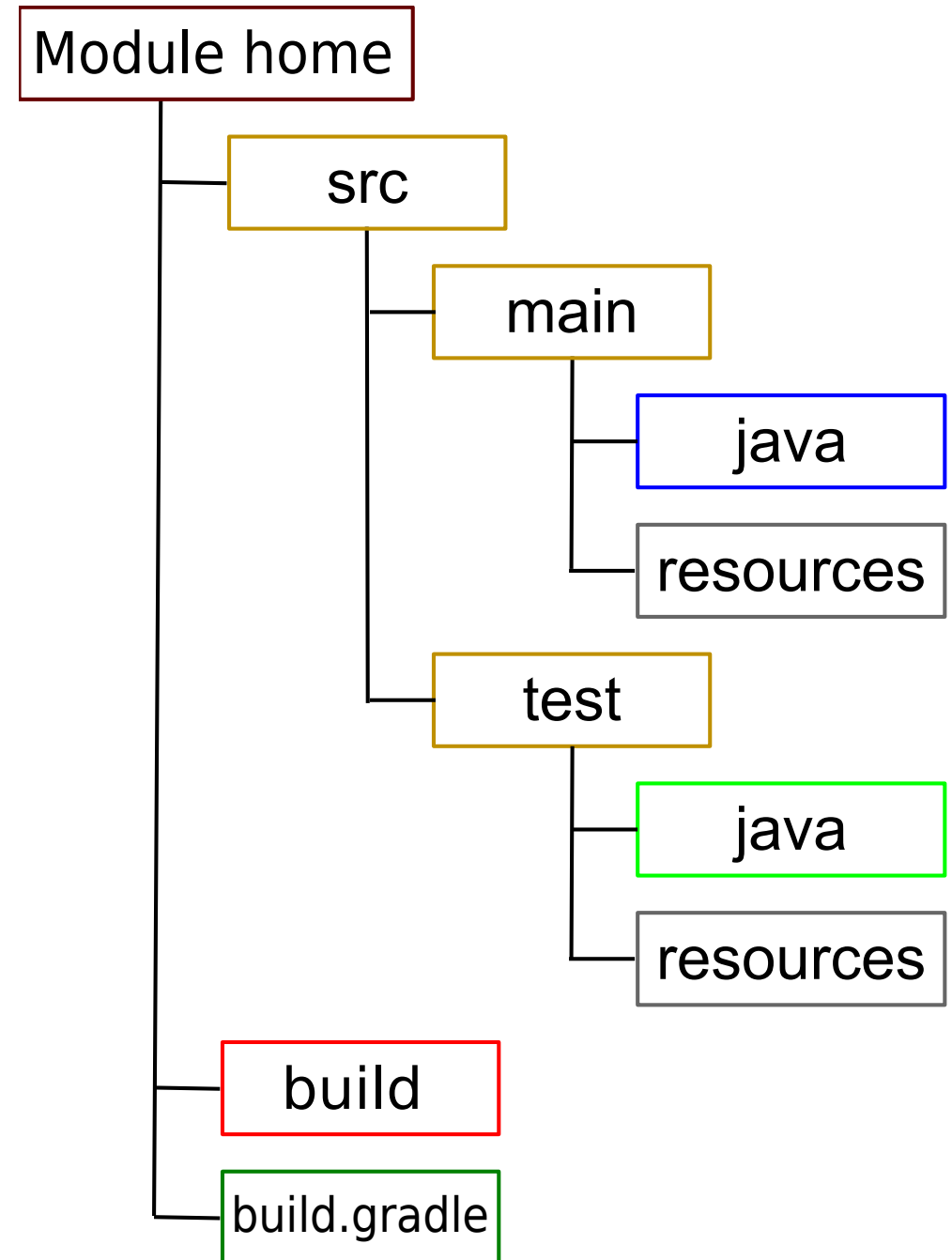
- Nicht in Versionskontrolle → Build Skripte schon
- Steht jedem Entwickler zur Verfügung
- **Gut Automatisierbar mit Build Servern**
- **Trend: IDEs verwenden Build-Tools für Projekt Organisation**



- Build Management Tool / Build System
- DSL (Domain-specific language) die auf Groovy Basiert
- Man schreibt **scripte** und **keine** Config files (Maven → xml)

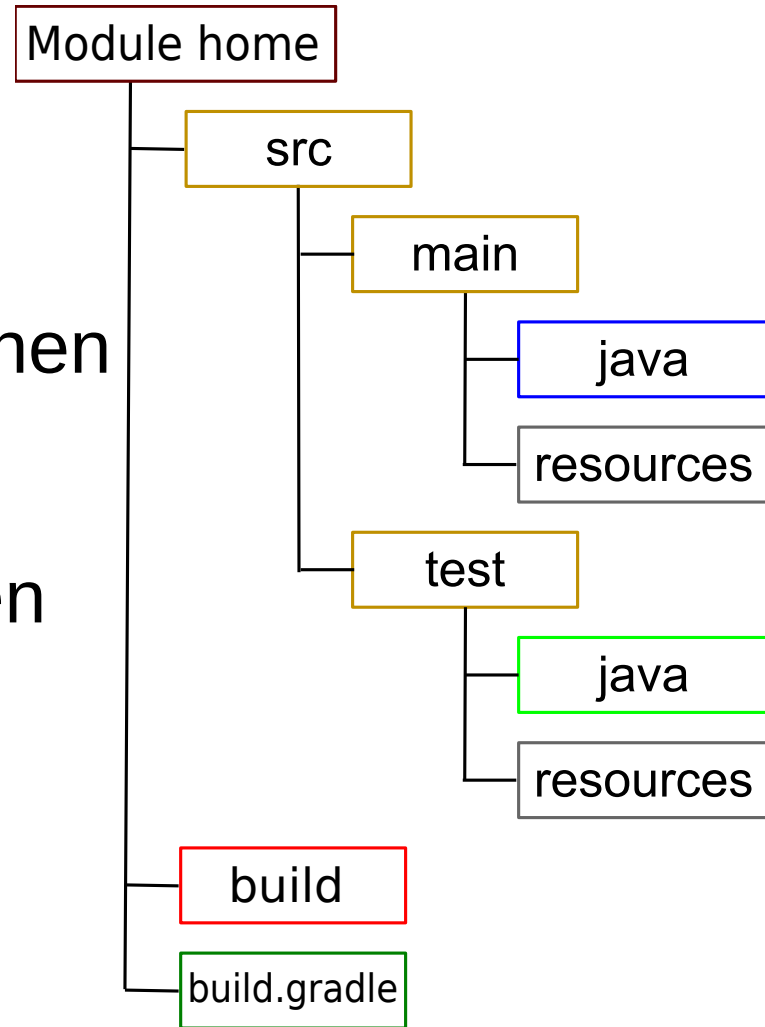
Gradle Project Structure

- Im wesentlichen von **maven** entliehen
- **src**: source files
 - **main**: java Klassen mit resources
 - **test**: test java Klassen mit resources
- **build**: compilierte files
 - NICHT in die Versionskontrolle
 - Wird automatisch erstellt wenn benötigt



Gradle

- **build.gradle** (Projekt-/Modul-Root)
 - Sind die eigentlichen build Skripte
 - Beinhalten Tasks und Konfigurationen
 - Erben Tasks und Configs von Obermodulen
 - Hier kann gradle ausgeführt werden
- **setting.gradle** (Projekt-Root)
 - Definiert Module die von gradle gebaut werden sollen
 - Einige globale Optionen



Java Modules vs Packages

Package (Compiler):

- Dient der Einordnung von Java Klassen in Namespaces um **Rechte/Sichtbarkeit** zu organisieren

Module/Jars (Build-Tool):

- Einheit die als eigenständiges **JAR** zur Verfügung stehen soll
- Einteilung in use cases
- Welche Teile meiner Software/Library möchte ich separat bereitstellen
 - Bsp.: API ↔ konkrete Implementierungen
- Der interessante Organisationsmechanismus für das Build-Tool

Minimales Projekt

build.gradle

```
group 'de.unijena.bioinf.dm.20[year].grp[number] '  
version '1.0-SNAPSHOT'
```

settings.gradle (Optional: sonst wird Verzeichnisname verwendet)

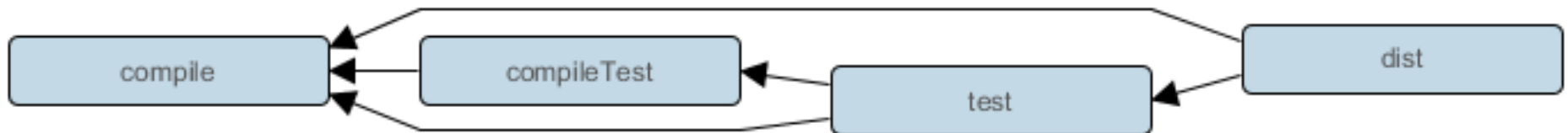
```
rootProject.name = 'exactSearch'
```

Tasks

Tasks (gradle -q tasks)

- Aufgaben die Gradle erledigen kann
- Verschiedene Typen von Tasks
- Können Abhängigkeiten haben
- Gradle bringt grundlegende Tasks mit (**Sprachenunabhängig**)
- Weitere Tasks
 - Über Plugins (zB. java, maven)
 - Selbst definieren

Beispiel: gradle test



Java Plugin - Java spezifische Tasks

Was muss ich tun?

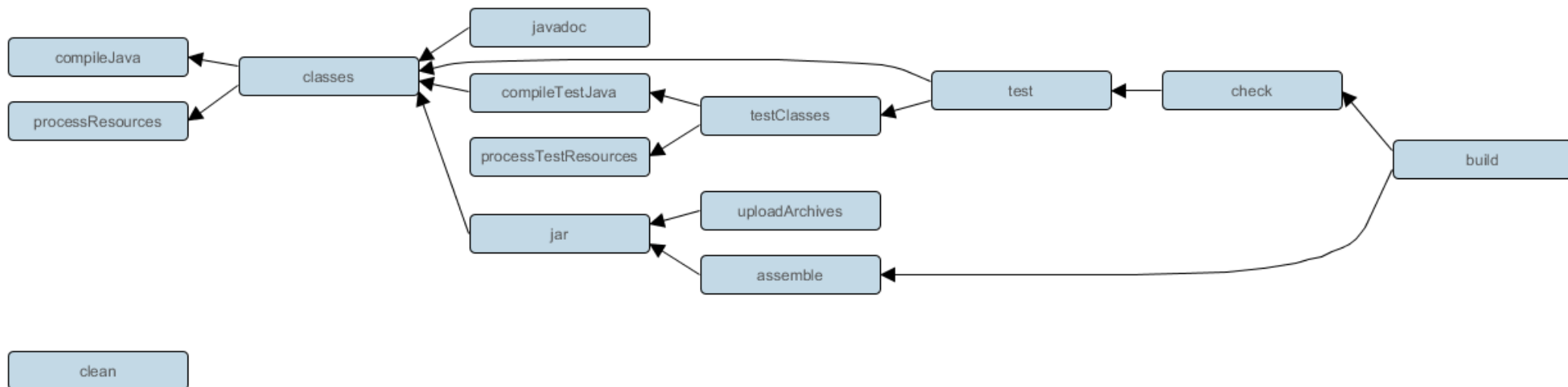
build.gradle

```
apply plugin: 'java'
```

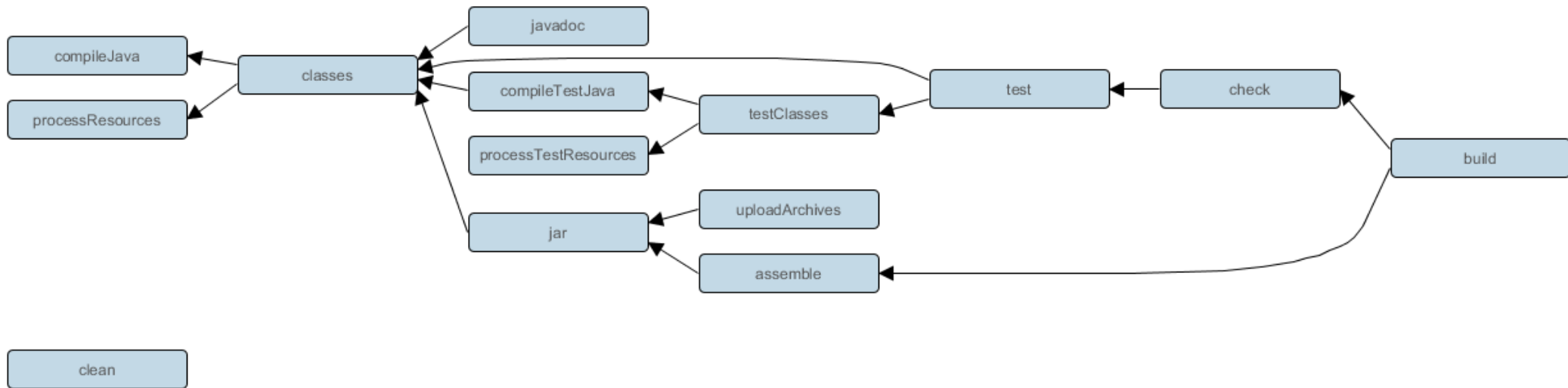
Konsole

```
gradle build
```

Was bekomme ich?



Java Plugin - Java spezifische Tasks



- Das Projekt wird kompiliert
- Javadoc wird erzeugt
- Die Test werden ausgeführt
- Jar wird erzeugt und im Ordner moduleRoot/build gespeichert

Javadoc

Konsole

```
gradle javadoc
```

Clean

Konsole

```
gradle clean
```

Welche Taks gibt es?

Konsole

```
gradle tasks
```

Tasks Konfigurieren - build.gradle

build.gradle

```
dependencies {
    compile group: 'args4j', name: 'args4j', version: '2.33'
    testCompile group: 'junit', name: 'junit', version: '4.12'
}

jar {
    baseName = 'exactSearchTool'
    manifest {
        attributes 'Main-Class': 'package.path.to.your.MainClass'
    }
}
```

- Konfiguriere dependencies
- erzeugt das Jar mit dem Namen **exactSearchTool-<VERSION>.jar**
- erzeugt ausführbares Jar

Repositories - build.gradle

build.gradle

```
repositories{
    mavenCentral()
    mavenLocal()
    maven {
        url "http://repo.xyz.de/maven2"
    }
}
```

Projekt ins lokale Repository packen

build.gradle

```
apply plugin: 'maven-publish'

publishing {
    publications {
        mavenJava(MavenPublication) {
            from components.java
        }
    }
}
```

Konsole **> gradle publishToMavenLocal**

Eigene Projekte einbinden

Exact Search

build.gradle

```
group 'de.unijena.bioinf.teaching.20[year].grp[number]'  
version '1.0-SNAPSHOT'
```

settings.gradle (Optional: sonst wird Verzeichnisname verwendet)

```
rootProject.name = 'exactSearch'
```

Konsole `> gradle publishToMavenLocal`

Eigene Projekte einbinden

Alignment

build.gradle

```
group 'de.unijena.bioinf.teaching.2018.grp01'  
version '1.0-SNAPSHOT'  
  
Dependencies {  
    compile group: 'de.unijena.bioinf.teaching.2018.grp01',  
        name: 'exactSearch', version: '1.0-SNAPSHOT'  
    ...  
}
```

settings.gradle (Optional: sonst wird Verzeichnisname verwendet)

```
rootProject.name = 'Alignment'
```

Eine Distribution erzeugen

build.gradle

```
apply plugin: 'application'
Application {
    mainClassName = „de.fsu.example.Test“
    applicationDefaultJvmArgs = [„-Xmx4G“]
}
```

/src/main/java/de/fsu/Example.java

```
package de.fsu.example;
public class Test {
    public static void main(String[] args){
        System.out.println("Programm gestartet");}
}
```

Konsole

```
> gradle distZip

Executing task 'distZip'...
> Task :compileJava
> Task :processResources
> Task :classes
> Task :jar
> Task :startScripts
> Task :distZip

BUILD SUCCESSFUL
```

- Ihr erhalten ein zip mit einem kompletten Programm für den User
 - Eure libs + dependencies und startscripten für Mac, Win, Linux
- „InstallDist“ → ohne zip, „run“ → führt Programm aus

Installationen - eventuell schon installiert

Was brauchen wir:

- **Java Installation (JDK)**
 - Test: `java -version`
- **Gradle installation**
 - Test: `gradle -v`