

# Inhaltsverzeichnis

1. Welcome .....	1
2. Fachlicher Schwerpunkt des Kurses (Zugdisposition) .....	1
2.1. Kontext .....	1
2.2. Klassenmodell .....	2
3. IDE Setup .....	4
3.1. Lokales Arbeitsverzeichnis: Der <b>Workspace</b> .....	4
3.2. Agile Software Development .....	5
3.3. Programmiersprache <b>Java</b> .....	5
3.4. IDE <b>IntelliJ</b> .....	6
3.5. Quellcodeverwaltung: <b>Git</b> .....	7
3.6. Herunterladen der Kursinhalte (Git mit IntelliJ) .....	8

## 1. Welcome

### Über mich



**Thorsten Eckstein**

<b>Konzern</b>	Deutsche Bahn, ca. 335.000 Mitarbeiter DB Systel GmbH, Frankfurt a.M., ca. 7.000 Mitarbeiter, IT-Dienstleister der Deutschen Bahn
<b>Qualifikationen</b>	Dipl. Geo-Informatik, zertifizierter Softwareentwickler, zertifizierter Softwarearchitekt

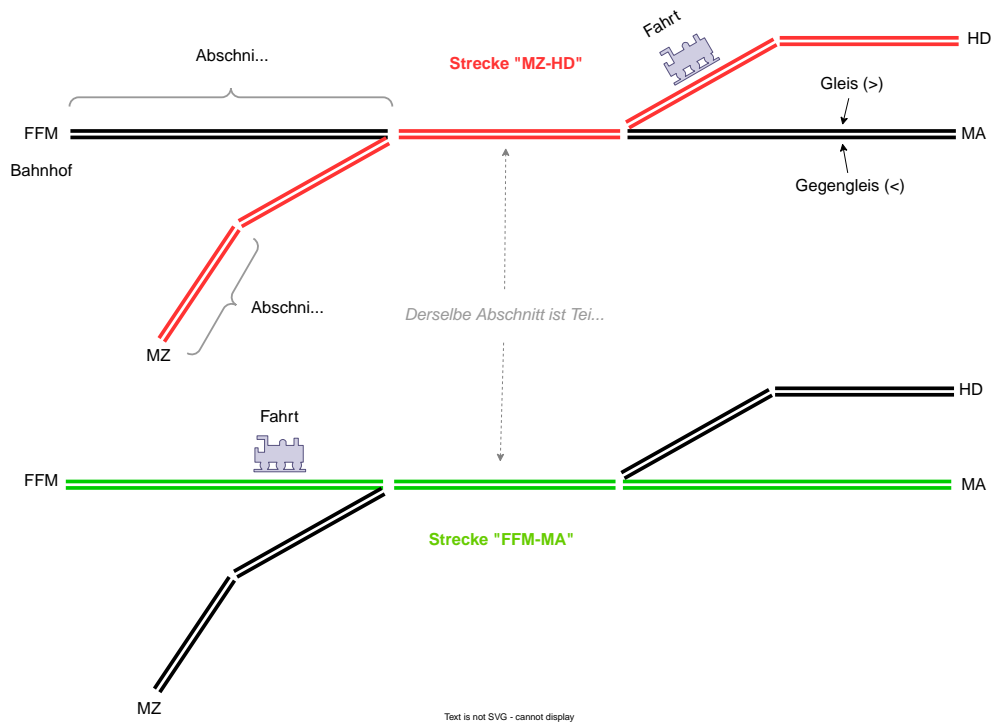
## 2. Fachlicher Schwerpunkt des Kurses (Zugdisposition)

### 2.1. Kontext

Der **Betrieb** von Zügen kann grundsätzlich unterteilt werden in:

1. **infrastrukturelle** Sicht (*Stammdaten*)
2. **planerische** Sicht (*Bewegungsdaten, SOLL*)
3. **operative** Sicht (*Bewegungsdaten, IST*)

## Ein paar wichtige Begriffe:



## 2.2. Klassenmodell

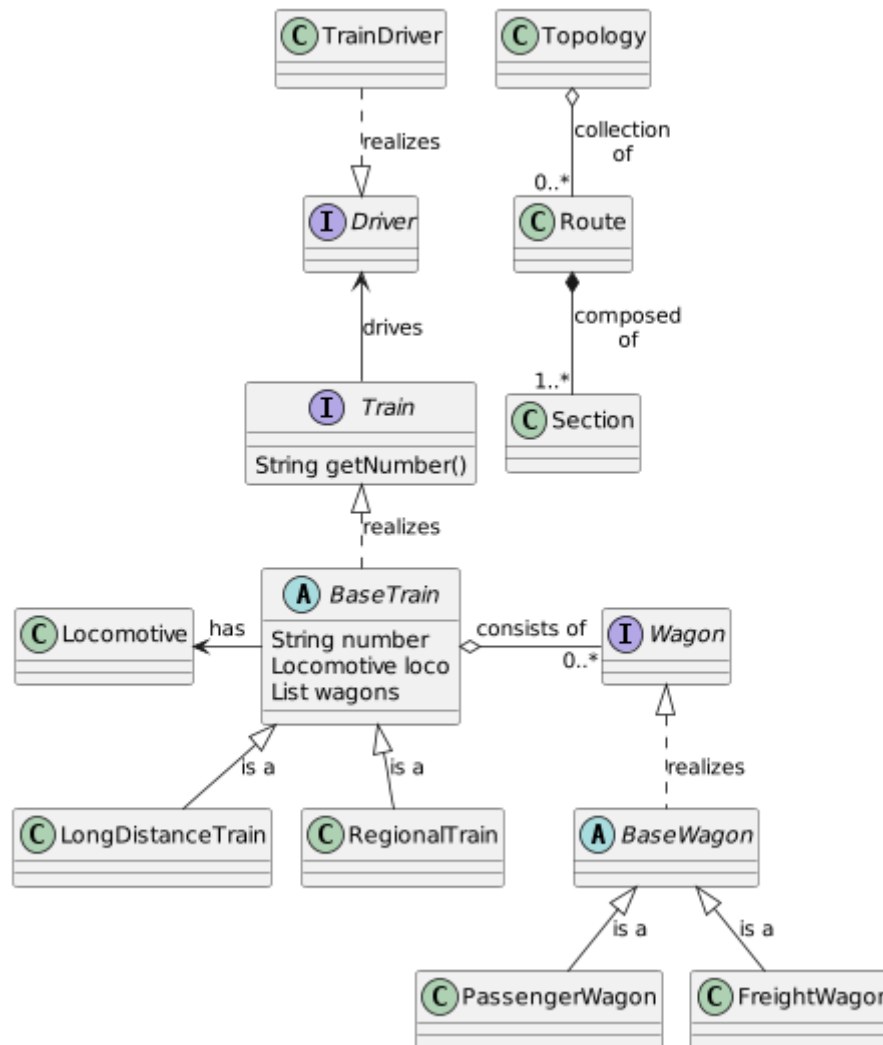
Eine solche Fachlichkeit (s.o.) muss, um Applikationen zu erstellen, in (Fach-) **Klassenmodellen** abgebildet werden, z.B. mit folgenden Objekten:

- + Zug (Zugarten), Wagon (Wagonarten)
- + Strecke, Abschnitt, Gleis
- + Ereignisse: Abfahrt, Ankunft
- + Fahrt
- + Fahrplan
- + Fahrgast
- + Dispositionsmaßnahme, Umleitung

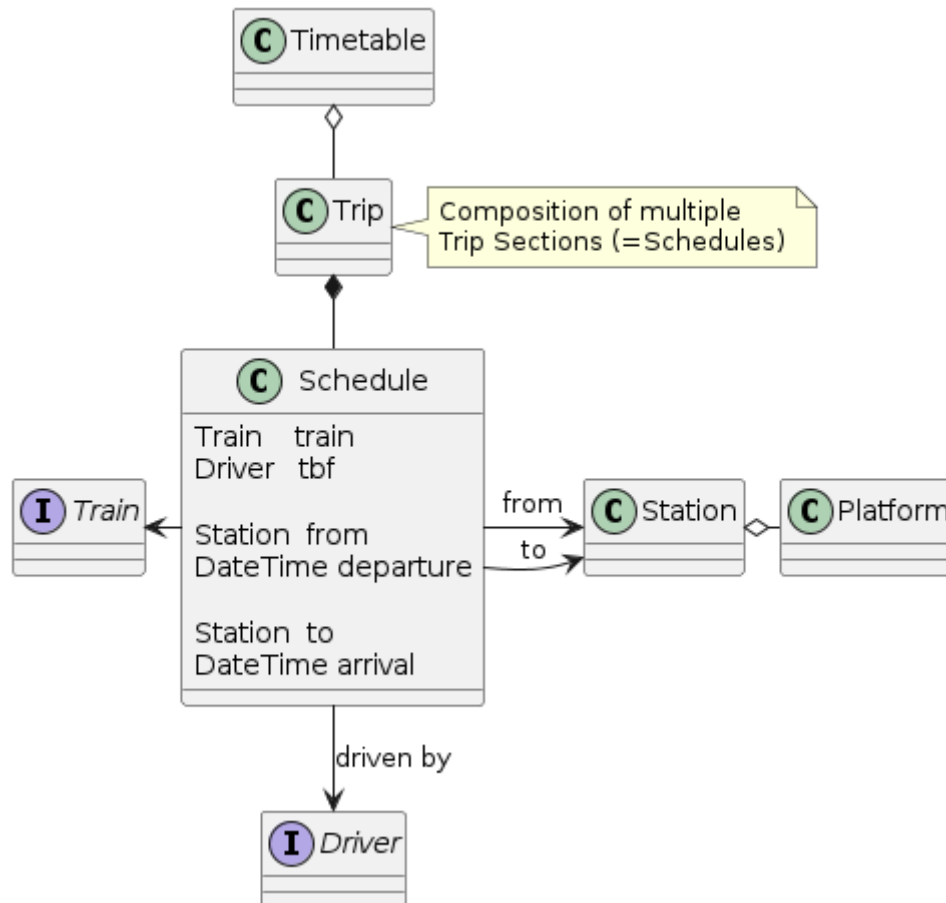
und viele weitere ...

Beispiel für ein grundsätzliches **Fachklassenmodell** mit zwei "Geschmacksrichtungen":

**(1) Infrastrukturelle Sicht (statische Sicht):**



(2) **Planerische** Sicht (dynamische Sicht):



## 3. IDE Setup

### 3.1. Lokales Arbeitsverzeichnis: Der **Workspace**

Zuerst wird ein **lokaler Workspace** eingerichtet, d.h. ein (separates) Wurzelverzeichnis, in das im Anschluss das die **Inhalte des Kurses**, d.h. SourceCode-Repository **gespeichert** werden können.



*Natürlich kann man hier immer den persönlichen Präferenzen für dieses Wurzelverzeichnis folgen, für die Einrichtung des Kurses ist es aber besser, den Empfehlungen zu folgen ...*

Der Workspace - das **lokale Arbeitsverzeichnis**:

Zur Einrichtung eines Workspace unter **Windows** (11) nutzt man am besten einfach den **Explorer** und erzeugt ein neues Verzeichnis, zum Beispiel (wird auch in späteren Beispielen genutzt) das folgende:

D:\Projekte

Wie gesagt, hierin werden die Kursinhalte in den nächsten Schritten dann lokal gespeichert und verwaltet (→ mittels **git**, siehe zugehöriges Kapitel).

## 3.2. Agile Software Development

Der "agile" Entwicklungsprozess als Grafik:

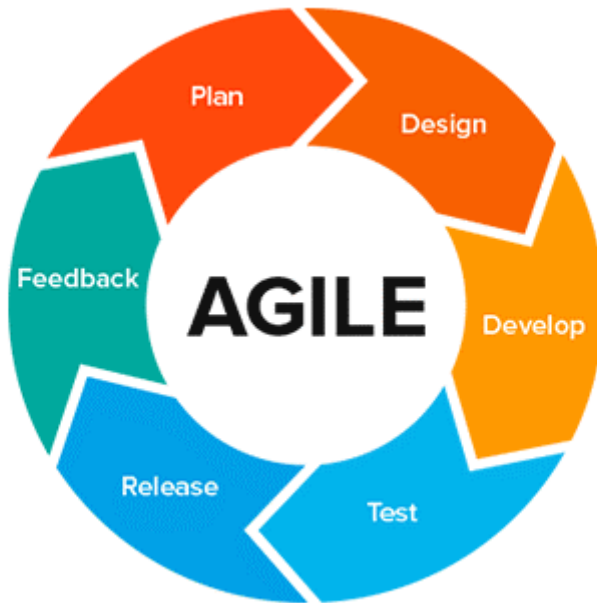


Figure 1. Der agile Entwicklungsprozess

In diesem Kontext werden viele **Tools** eingesetzt, dazu gehören auch diese, die in dieser LV eingesetzt werden, vor allem:

- IntelliJ
- Git

## 3.3. Programmiersprache Java

**Empfehlung:** Im Kurs soll folgende **Java** Version verwendet werden:

19

Ggf. muss diese Version noch installiert werden, wobei die Versionsnummern ("Minor") *hinter* der Hauptversionsnummer 19 ("Major") nicht so wichtig sind.

- In den neuen Workspace, momentan noch leeren Ordner (s.o.), wechseln

```
cd Projekte
```

- Hier ein Terminal/Kommando-Shell öffnen
- Prüfen, ob **java** vorhanden ist, einfach durch Eingabe von

```
java -version
```

Dann sollte eine Java Version angezeigt werden.

## 3.4. IDE IntelliJ

Die empfohlene IDE ist **IntelliJ**. Die LV-Inhalte sollten sowohl in der *Community* als auch in der *Enterprise* Edition funktionieren.

Der Code für den Kurs wurde mit folgender Version erstellt, die installiert sein muss (falls noch nicht vorhanden, bitte installieren):

IntelliJ IDEA 2023.3.2 (Ultimate Edition; Community Ed. sollte auch funktionieren)



Alternativ könnte auch **MS Visual Studio Code** genutzt werden und diese Umgebung sollte auch bei der Einrichtung keine besonderen Probleme verursachen. Aber ... auf eigene Gefahr ;-), die Lehrveranstaltung wurde für IntelliJ vorbereitet!

IntelliJ bietet ein neben dem Standard-Layout auch ein experimentelles, das aktiviert werden kann.

### IntelliJ PlugIns







#### Bundled PlugIns

Mit IntelliJ werden diverse PlugIns **automatisch mitinstalliert**, dazu gehören die folgenden, die für das Seminar erforderlich sind und anfangs mal geprüft werden sollen:

PlugIn	Kommentar
Git & GitHub	<i>Source Code Management, lokaler Git-Client</i>
Maven	<i>Build, Dependency &amp; Project Management</i>

#### Non-bundled PlugIns

Für den Quellcode, Tests und Dokumentation sind weitere PlugIns erforderlich, die *nicht* mit IntelliJ installiert werden, also separat installiert werden müssen:

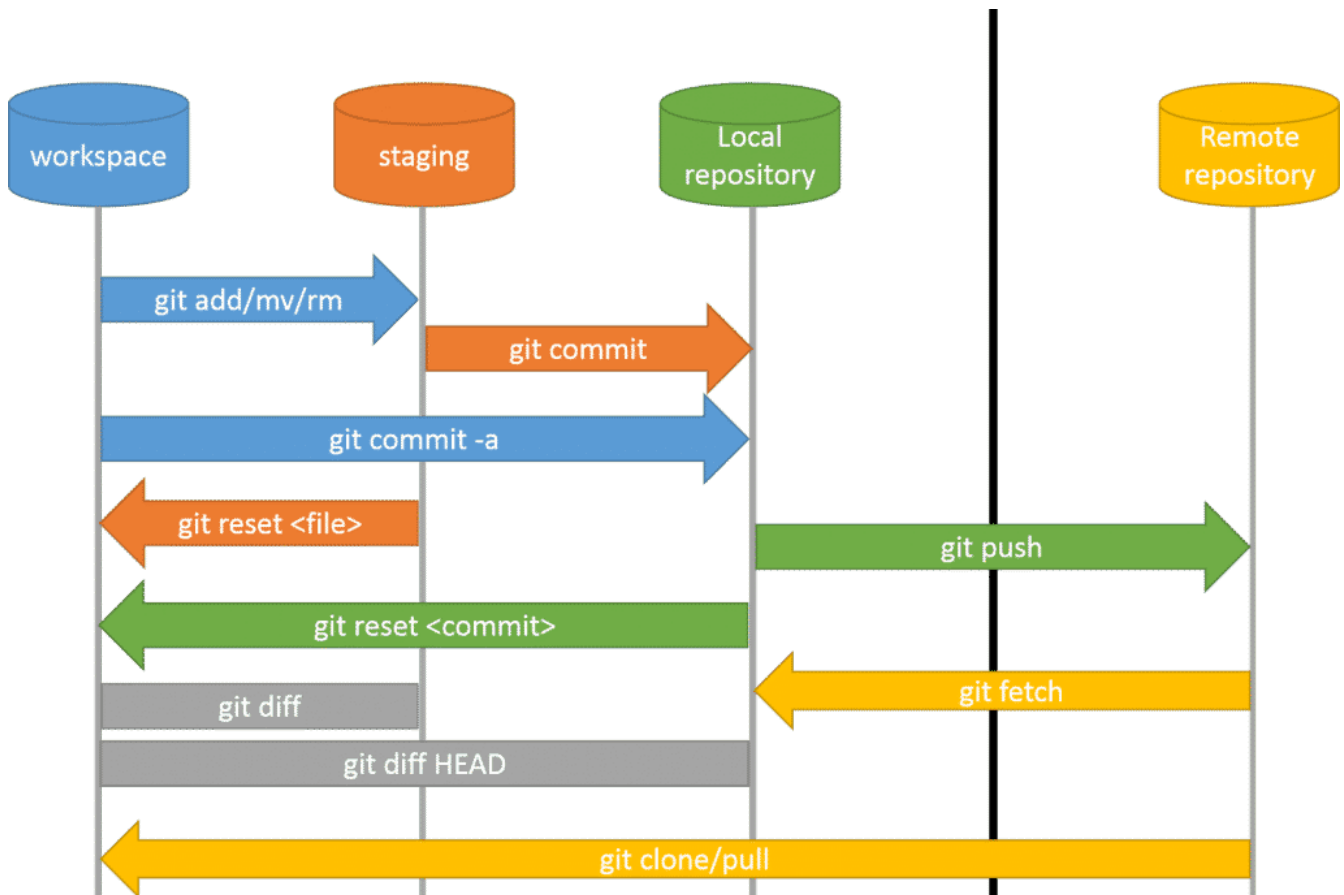
PlugIn	Was ist das?	Version
 <b>AsciiDoc</b>  0.38.9 AsciiDoctor IntelliJ Plugin Project	<i>Einheitliche Erstellung von Dokumentationen</i>	<b>&gt;= 0.41.x</b>
 <b>Diagrams.net Integration</b>  0.1.14 Docs As Code	<i>Erstellung von beliebigen Diagrammtypen</i>	<b>&gt;= 0.2.4</b>
 <b>PlantUML Integration</b>  5.22.0 Eugene Steinberg, Vojtech Krasa	<i>Erstellung von UML-Diagrammen</i>	<b>&gt;= 7.3.0-IJ2023.2</b>

PS: Ich persönlich aktualisiere IntelliJ selbst und auch die PlugIns recht zeitnah.

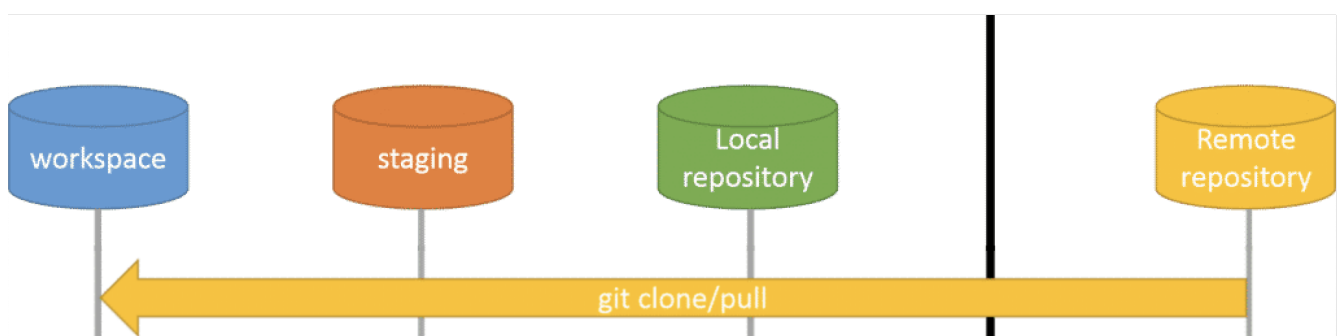
### 3.5. Quellcodeverwaltung: Git

Web: <https://de.wikipedia.org/wiki/Git>

Git ist ein verteiltes Versionsverwaltungssystem für Quellcode und setzt folgenden **Workflow** um:



Allerdings kommt im Kurs nur ein kleiner Teil hiervon zum Einsatz, lediglich das erstmalige **Downloaden** (**clone**) & die **Aktualisierung** (**pull**) der lokalen Dateien wird benötigt:

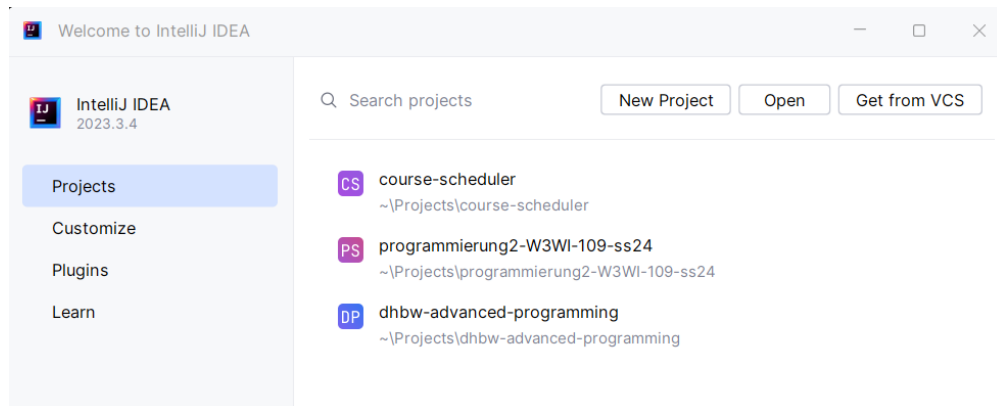


Das lokale Arbeitsverzeichnis wird **workspace** genannt. In diesem werden alle Quellcode-Dateien abgelegt, also z.B. ein Windows-Verzeichnis, in das der Quellcode heruntergeladen wird:

D:/Projekte

## 3.6. Herunterladen der Kursinhalte (Git mit IntelliJ)

1. In das eigene *Workspace*-Verzeichnis wechseln, z.B. **D:\Projekte**
2. IntelliJ starten. Wenn zuvor kein anderes Projekt geladen war, dann kommt ...



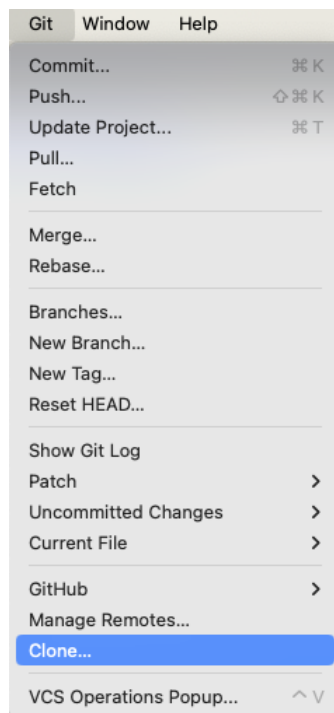
3. Nun soll das Projekt von GitHub heruntergeladen werden, dazu gibt es 2 Varianten:

a.

**Get from VCS** anklicken (VCS = Version Control System, **Git**)

**Get from VCS**

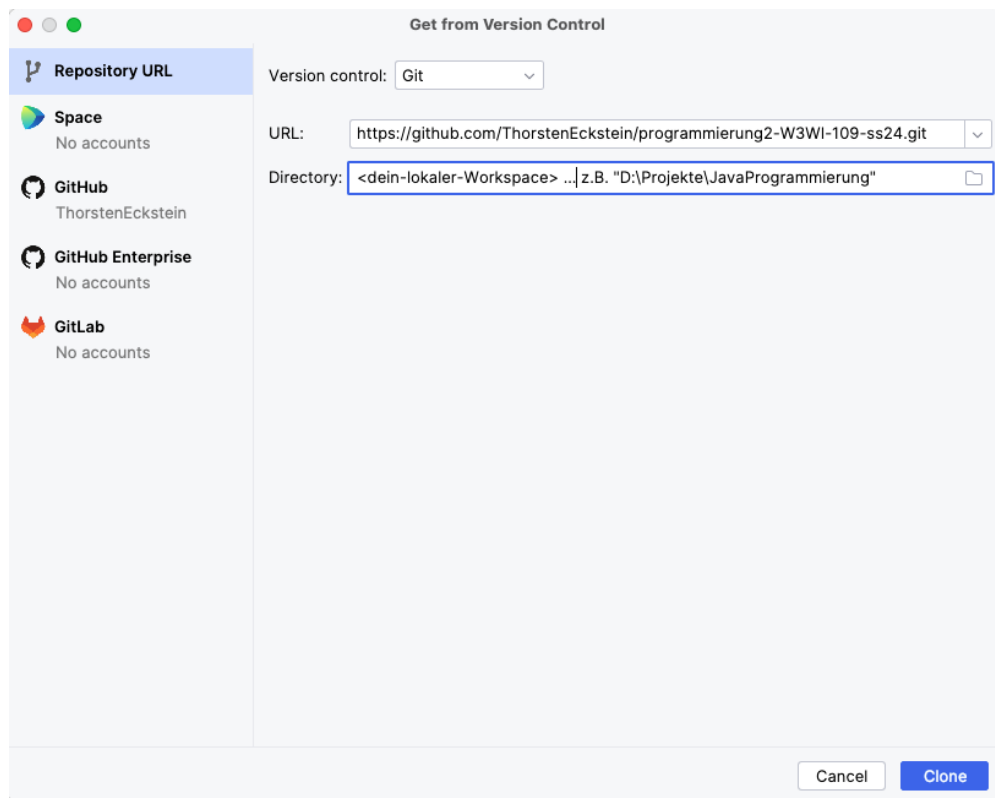
b. oder über die Menüoption **Git > clone ...** aufrufen



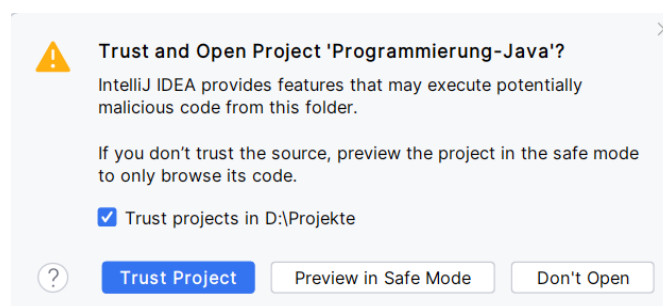
4. In beiden Fälle erscheint der gleiche **clone-Dialog**, in dem die folgenden Angaben eingegeben werden müssen:

**URL** : <https://github.com/ThorstenEckstein/programmierung2-W3WI-109-ss24.git>  
**Workspace** : <dein-lokales-Arbeitsverzeichnis>

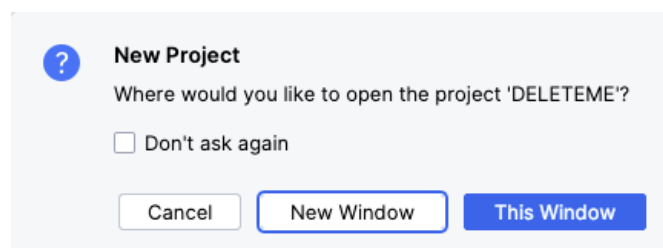




5. Dann kommt (wahrscheinlich) eine Abfrage zur **"Vertrauenswürdigkeit"** → [Trust Project]:



6. Im Falle von "3b" (siehe oben) kann man noch entscheiden, ob das Projekt **in einem separaten Fenster** geöffnet werden soll → [New Window]:



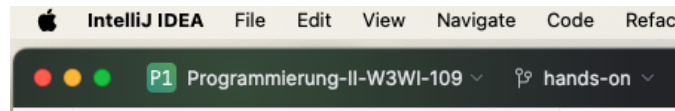
Nun sollte das heruntergeladene Projekt in IntelliJ vorhanden sein.



*Falls irgendetwas am Ende nicht so ist wie gedacht, kann das lokale Arbeitsverzeichnis einfach lokal gelöscht werden. Dann führt man die Schritte einfach nochmal durch.*

Zur Überprüfung einmal (in der IDE) schauen, ob der richtige **Branch** namens

heruntergeladen wurde. Dies ist sozusagen eine spezielle Variante (oder auch Version), die im Semester genutzt wird.



Es kann auch die *Kommandozeile* genutzt werden, um das Repository zu klonen, dazu gibt es ein paar Tipps im folgenden Abschnitt, kann aber für den Moment ignoriert werden, es sei denn, das der Weg über IntelliJ nicht funktionieren sollte.

#### ▼ Aufklappen für Git mit der Kommandozeile ...

- Zuerst prüfen, ob **git** installiert ist. In der Kommandozeile geht dies durch

```
git --version
```

- was zum Beispiel folgende Ausgabe ergibt: **git version 2.39.1**
- Dann in das lokale Arbeitsverzeichnis **wechseln** (oder neu anlegen), d.h. in den Workspace, z.B. nach ...

```
cd D:/Projekte
```

- Innerhalb dieses Verzeichnisses wird das Repository **geklont**, d.h. heruntergeladen. Das Repository lautet:

```
https://github.com/ThorstenEckstein/programmierung2-W3WI-109-ss24
```

Dieses Repository wird geklont und damit zu einer lokalen Kopie im **Workspace** mit:

```
git clone https://github.com/ThorstenEckstein/programmierung2-W3WI-109-ss24.git -b  
hands-on ①
```

① Der Branch-Name **hands-on** ist hier wichtig, das ist der exakte "Ort" in der Online-Quellcodeverwaltung (Repository), von dem aus heruntergeladen wird!

- Zur Prüfung, auf welchem Branch man sich aktuell befindet, gibt man ein:

```
git branch
```

- Hier sollte jetzt etwas Ähnliches erscheinen. Das Sternchen **\*** zeigt an, welcher Branch gerade

benutzt wird, das *soll* hier auch so sein:

\* hands-on



... es soll kein Code in das *remote* Repository **gepushed** werden, da es sonst zu Konflikten bei möglicherweise erforderlichen Updates (durch *pullen*) kommen kann!