Vanderlande IntelliJ Workshop

Warum BlueJ nicht die Lösung sein kann

Einführung in aktuelle IDEs am Beispiel von IntelliJ

Sabrina Friesenborg, Pascal Michallik, Marvin Polinowski, Nico Redick

VANDERLANDE

Contents

Begrüßung

Grundeinführung IntelliJ

Grundeinführung Git

Git in IntelliJ

Hands on

Mittagspause

Debugger und Generatoren

Hands on

Abschluss

Feedback

Ablauf

Vormittags (10-12 Uhr)	Nachmittags (13-16 Uhr)
Vorstellung	Teil 2: debugging, Generatoren, Doku
Teil 1: git basics, Code style, renaming	Hands on Teil 2
Hands on	Abschlussrunde
Wie wird Software im Unternehmen entwickelt?	Fragen- und Feedbackrunde

VANDERLANDE

Nico Redick

- Javaentwicklung
- Seiteneinsteiger ins Studium

Marvin Polinowski

- Verschd. Sprachen
- Software Engineer allgemeiner Service

Pascal Michallik

- Javaentwicklung mit Maven (Alternative zu gradle)

Sabrina Friesenborg

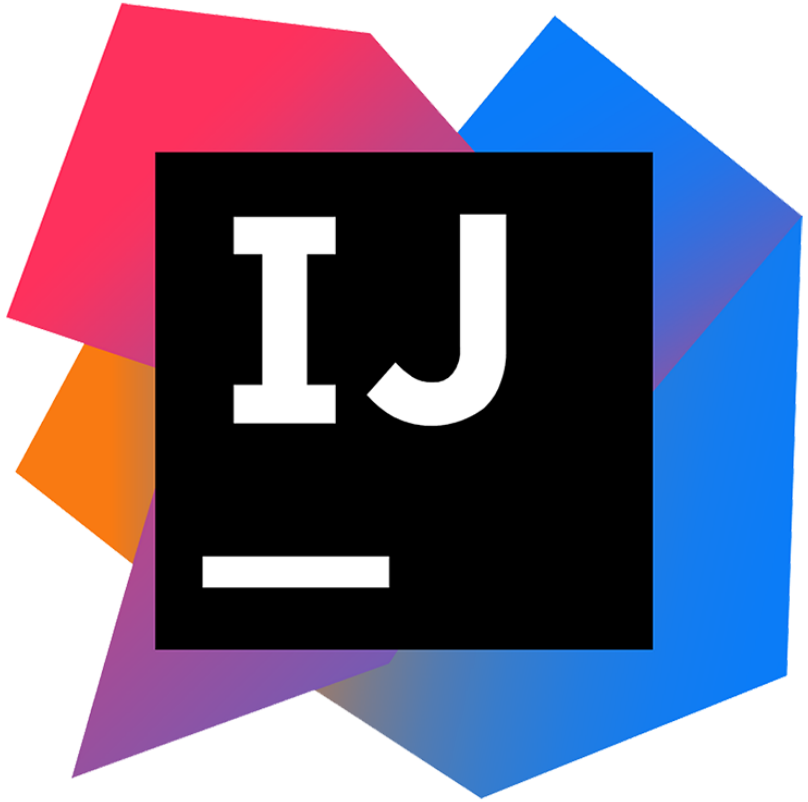
- Verschd. Sprachen
- Software Engineer spezieller Service

VANDERLANDE

Grundeinführung IntelliJ

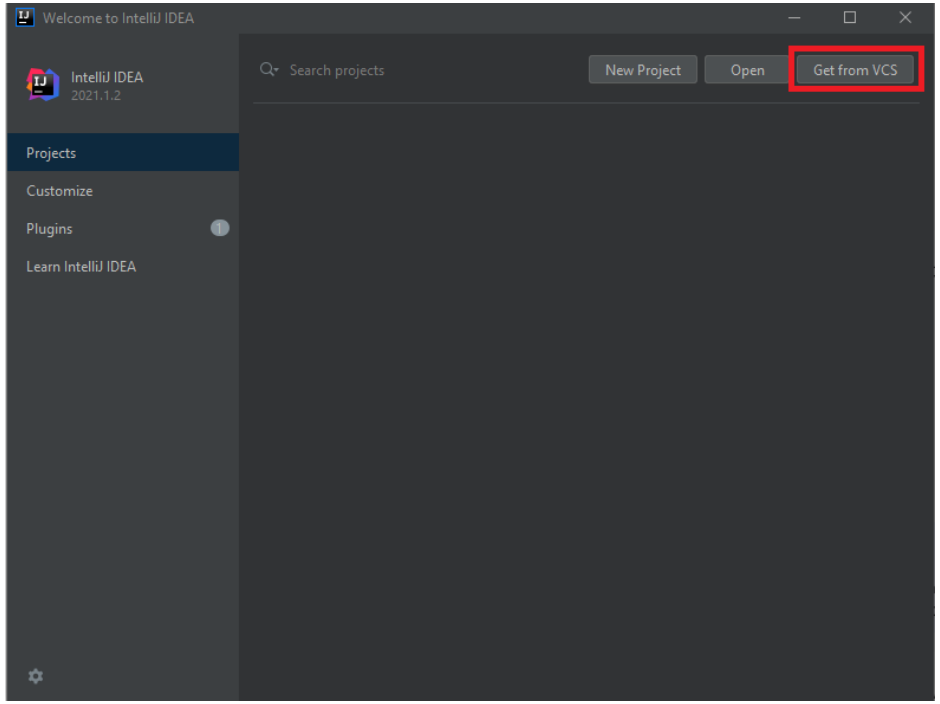
01 Import eines Projektes

02 Grundlagen UI



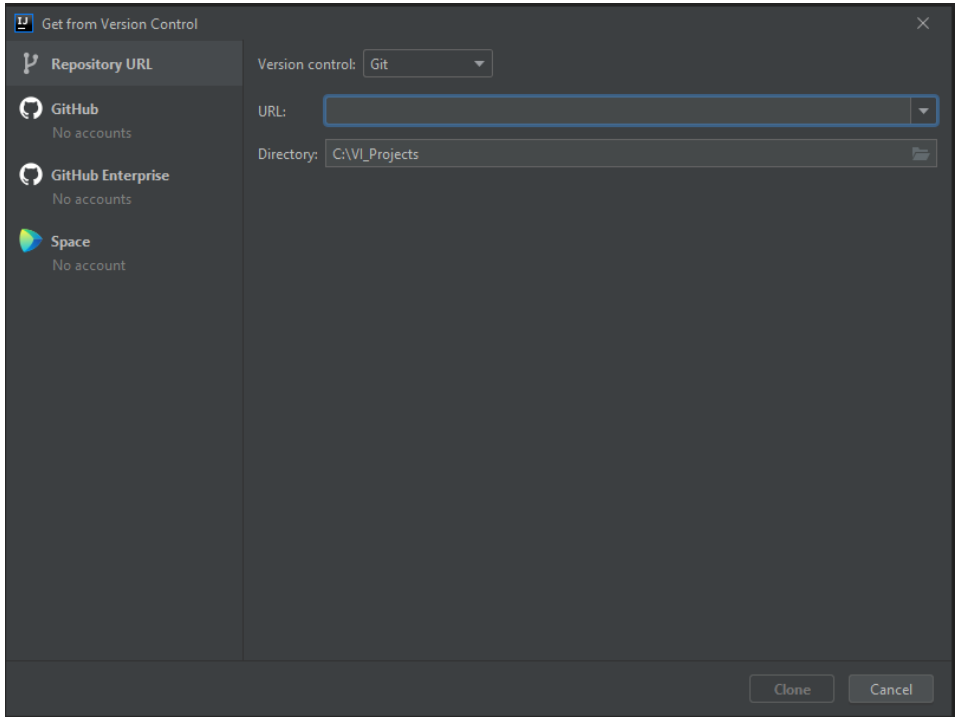
GIT Integration in IntelliJ - Cloning

- Startbildschirm von IntelliJ
- Über “Get from VCS” beginnen wir ein Projekt mit Daten aus einem Version Control System (z.b. Git)



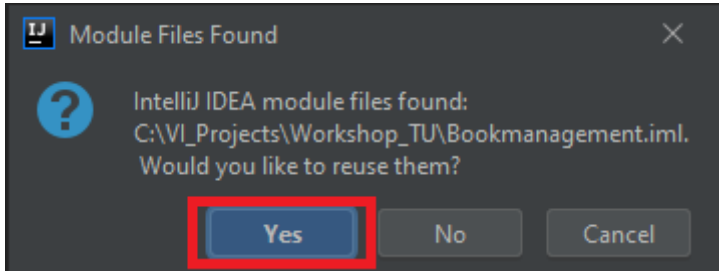
GIT Integration in IntelliJ - Cloning

- **Unter URL wird der Repository Link eingetragen**
  - Für diesen Workshop ist das: https://gitlab.com/marvinpolinowski/buchverwaltung.git
- **Unter Directory wird das Projekt Speicherort angegeben**



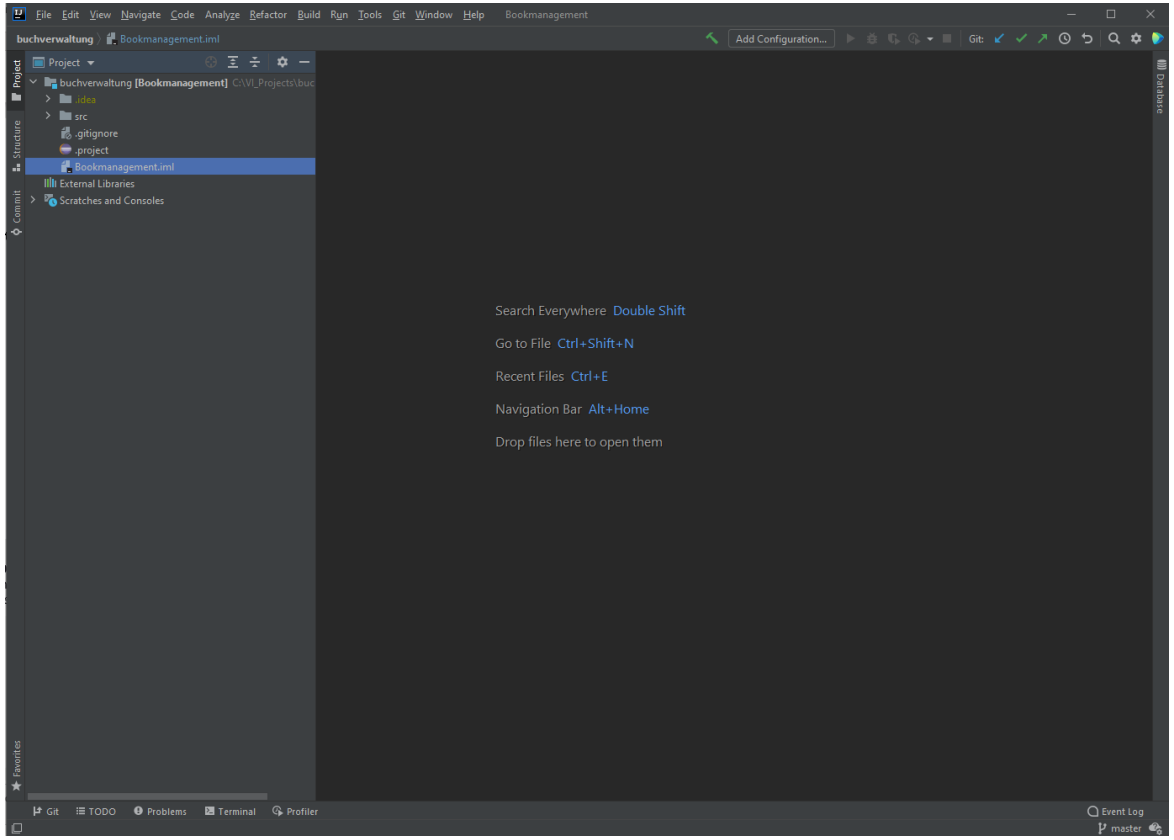
GIT Integration in IntelliJ - Cloning

› **Den Dialog mit “Yes” bestätigen**



Einführung IntelliJ IDEA

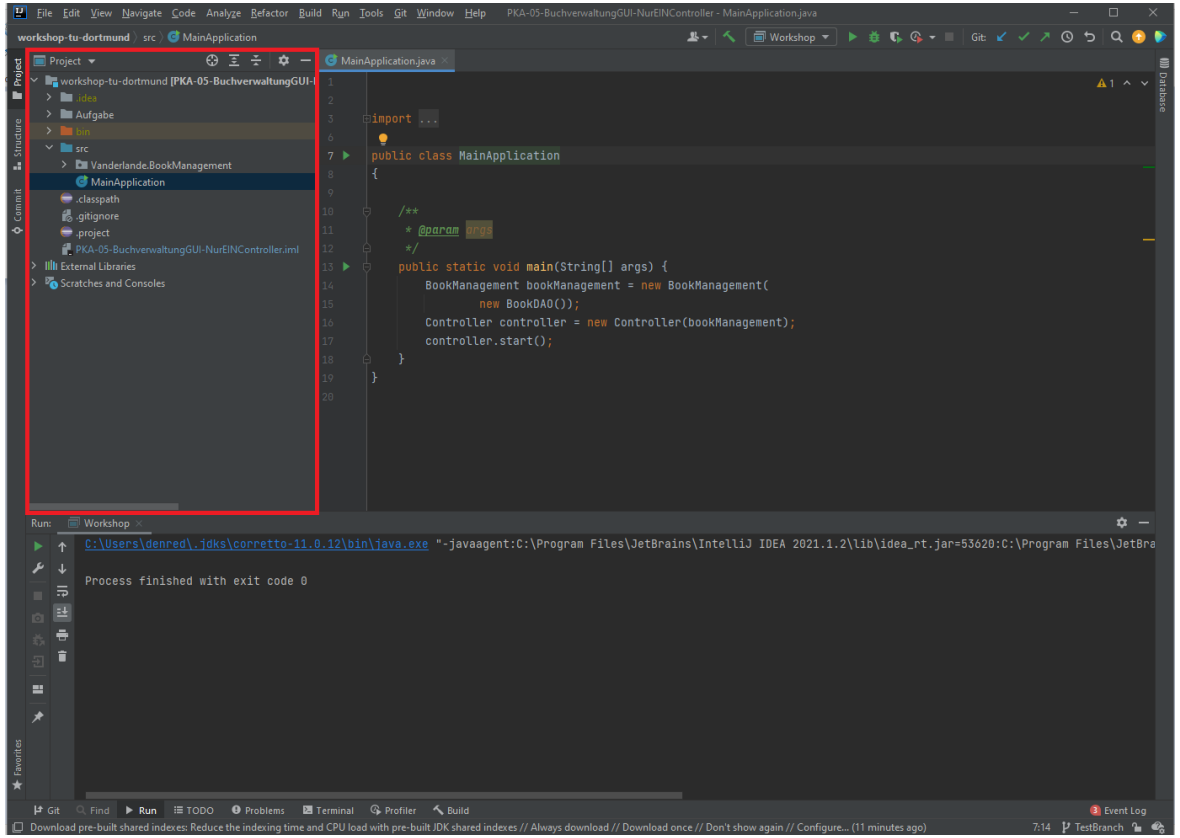
› **Nach dem clonen sollte die GUI ungefähr so aussehen:**



VANDERLANDE

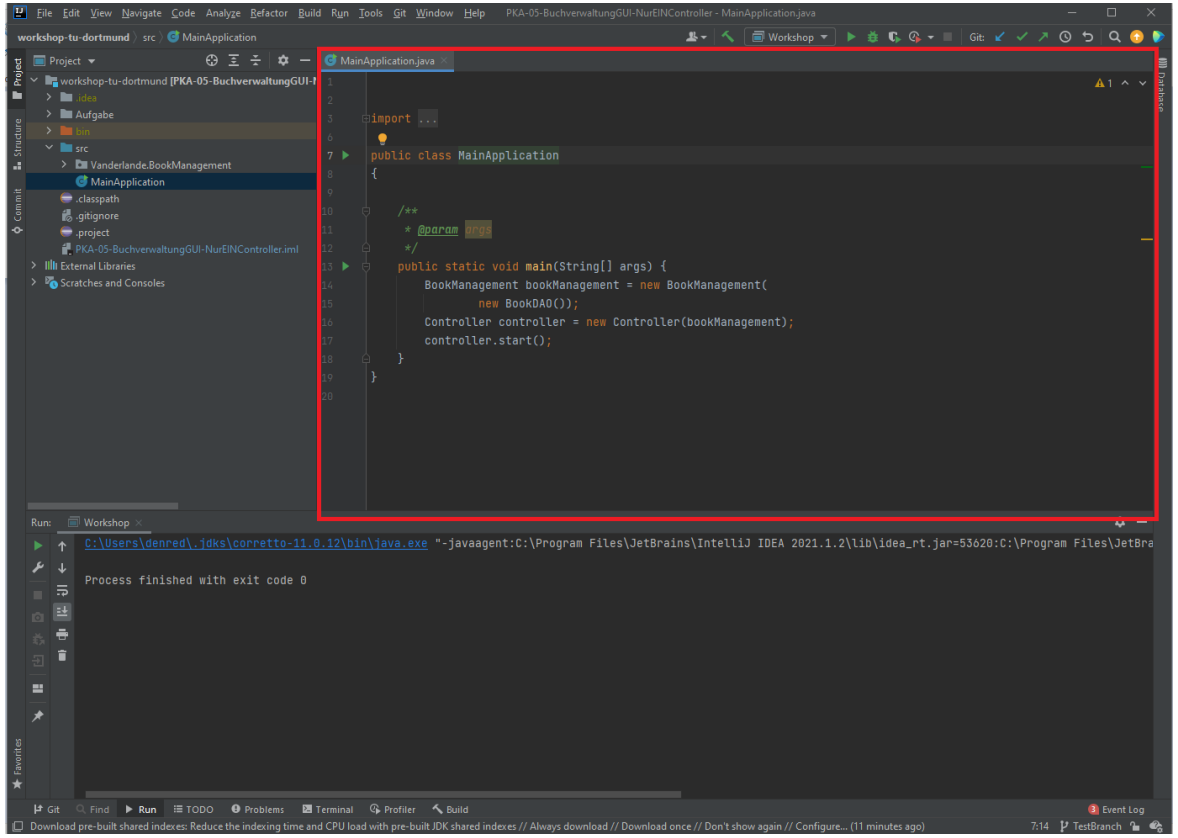
Einführung IntelliJ IDEA - Komponenten

› **Projekt Browser**



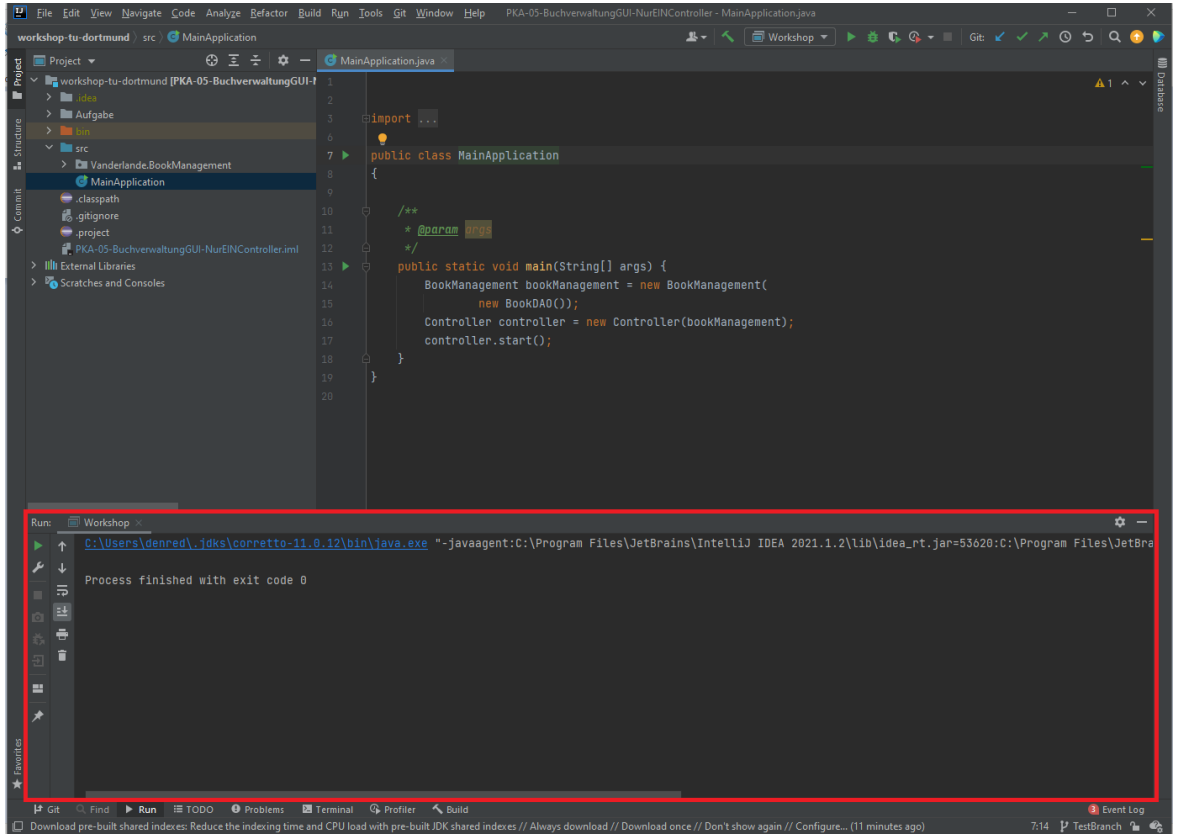
Einführung IntelliJ IDEA - Komponenten

- **Text Editor**



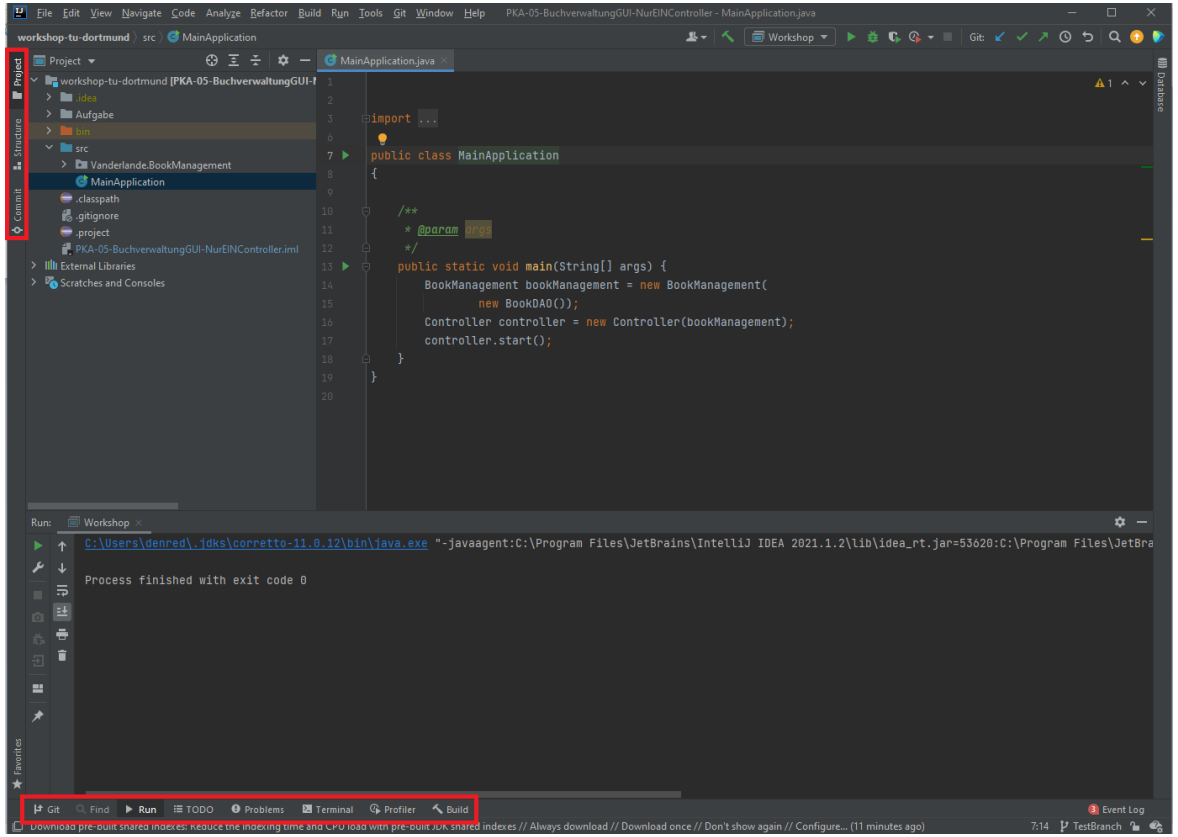
Einführung IntelliJ IDEA - Komponenten

- Java Konsole



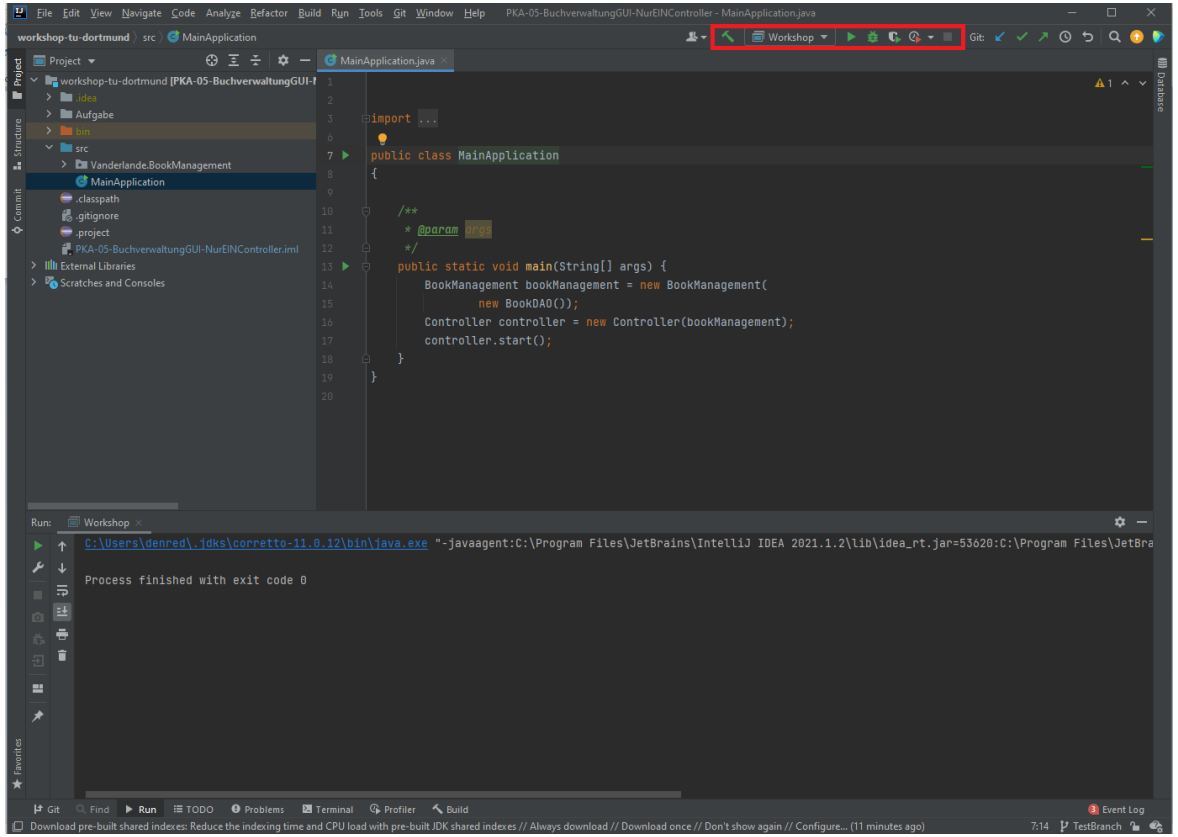
Einführung IntelliJ IDEA - Komponenten

- Navigations Register
- Links ändern das Projekt Browser Fenster, unten das Java Konsolen Fenster



Einführung IntelliJ IDEA - Komponenten

- Shortcuts für Debugging und Ausführen des Programms
- Knöpfe: Build, Konfigurationen, Run, Debug Run, Run with Coverage, Run with Flight Recorder



VANDERLANDE

Grundeinführung Git

01 Projekt Erstellung

02 Staging

03 Updating

04 Branching

Einführung GIT

Grundlegendes:

Quelle: https://xkcd.com/1597/

VANDERLANDE

Einführung GIT – Ein Projekt beginnen

init	> Erstellt ein GIT Repository in einem bestehenden Ordner
clone [url]	> Erstellt ein bestehendes GIT Repository von der URL im angegebenen Ordner

Einführung GIT – Staging

add [file]	> Fügt eine Datei in ihrem aktuellen Zustand dem Commit hinzu (Die Datei ist damit “staged”)
reset [file]	> Entfernt eine Datei aus dem Commit (“unstaging”)
diff	> Zeigt an welche Dateien Verändert sind im vergleich zum aktuellen Branch, und noch nicht staged sind.
commit –m “[message]”	> Fasst alle staged files in einem Commit mit der angegebenen Message zusammen

Einführung GIT – Updating

push [branch]	> Fügt die Änderungen die im Commit festgelegt sind dem Branch hinzu.
pull	> Holt die Änderungen vom aktuellen Branch und versucht sie in das aktuelle Arbeitsverzeichnis einzufügen

VANDERLANDE

Einführung GIT – Branching

branch [branch-name]	> Erstellt einen neuen Branch mit dem angegebenen Namen
checkout [branch-name]	> Fügt den branch den lokalen branches hinzu
merge [branch-name]	> Fügt den angegebenen mit dem aktuellen Branch zusammen
diff branchA…branchB	> Zeigt welche Änderungen in BranchA Vorhanden sind aber nicht in BranchB

Einführung GIT – Branching

VANDERLANDE

Git in IntelliJ

01 Projekt Erstellung

02 Staging

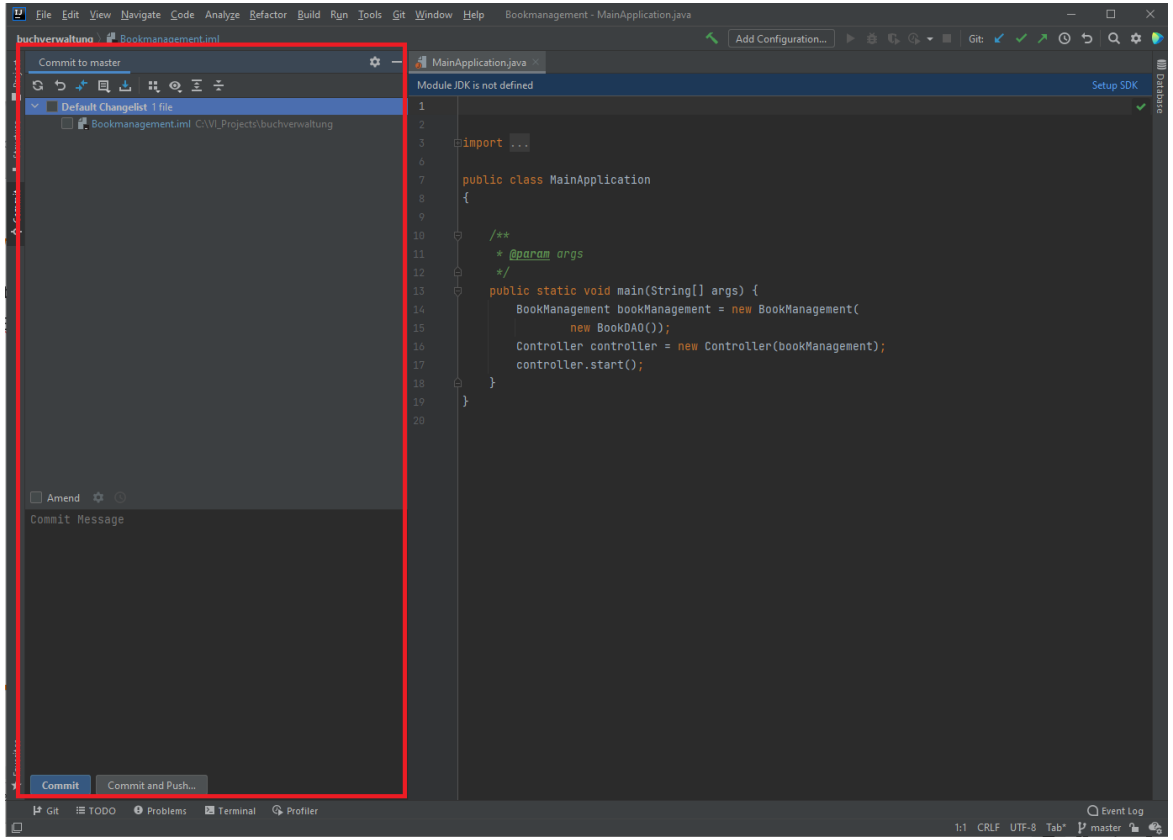
03 Updating

04 Branching

GIT Integration in IntelliJ – Staging

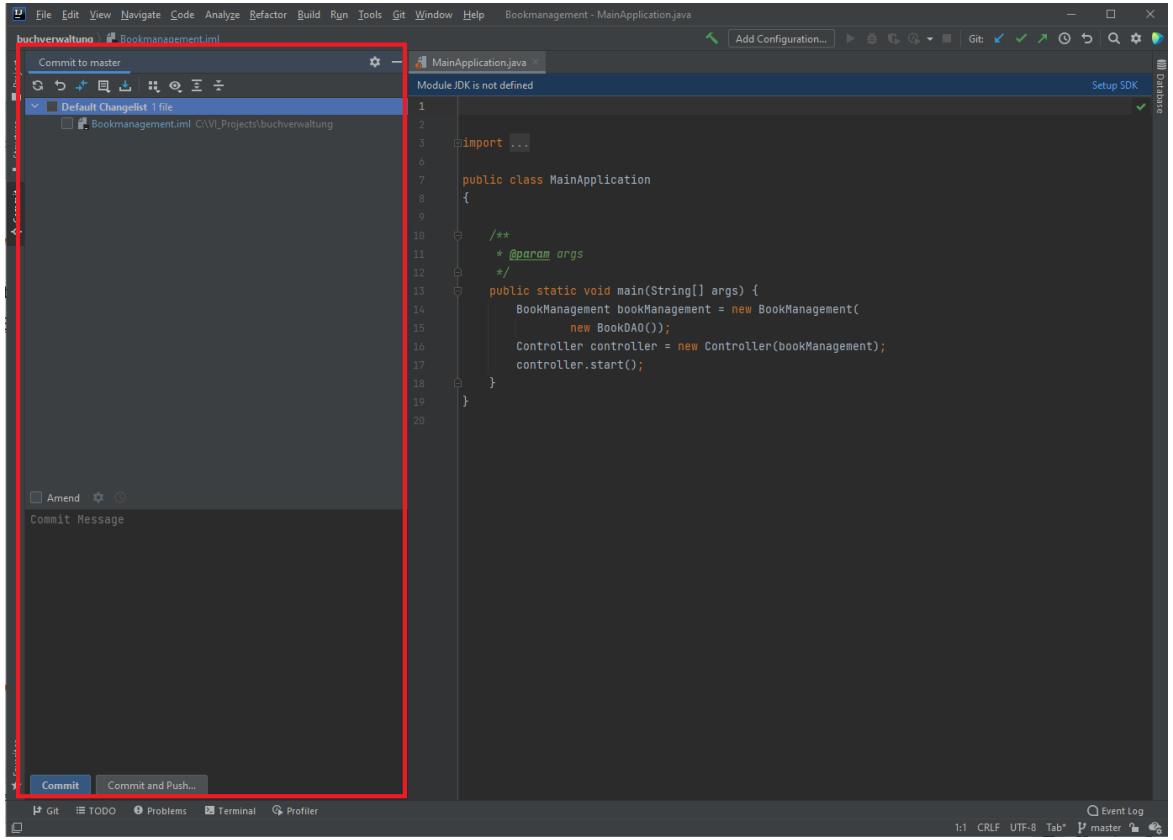
› Git Commit Browser

- Erreichbar über das obere linke Navigationsregister



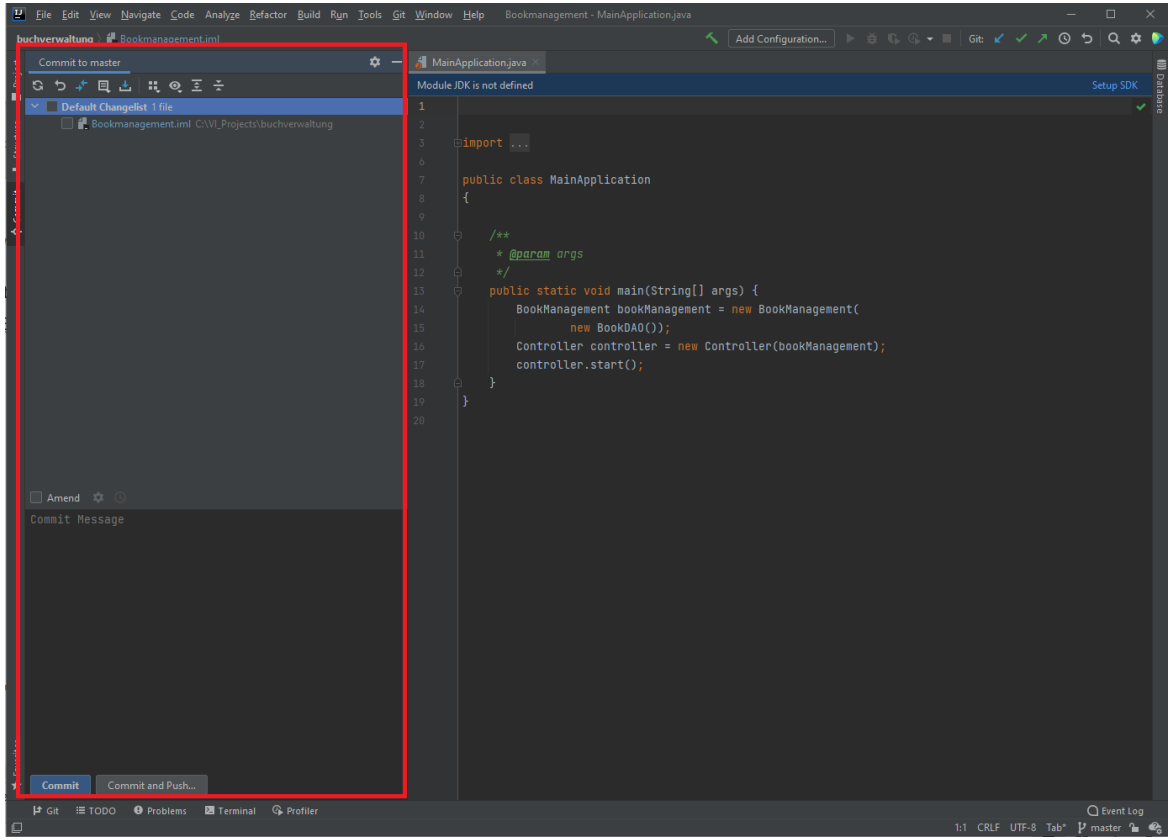
GIT Integration in IntelliJ – Staging

- Dateien die sich gegenüber dem lokalen Branch verändert haben werden hier aufgelistet
- Dateien die mit einem Haken versehen sind, werden in den nächsten Commit eingefügt



GIT Integration in IntelliJ – Staging

› **In die Textbox wird die Commit Message eingetragen**



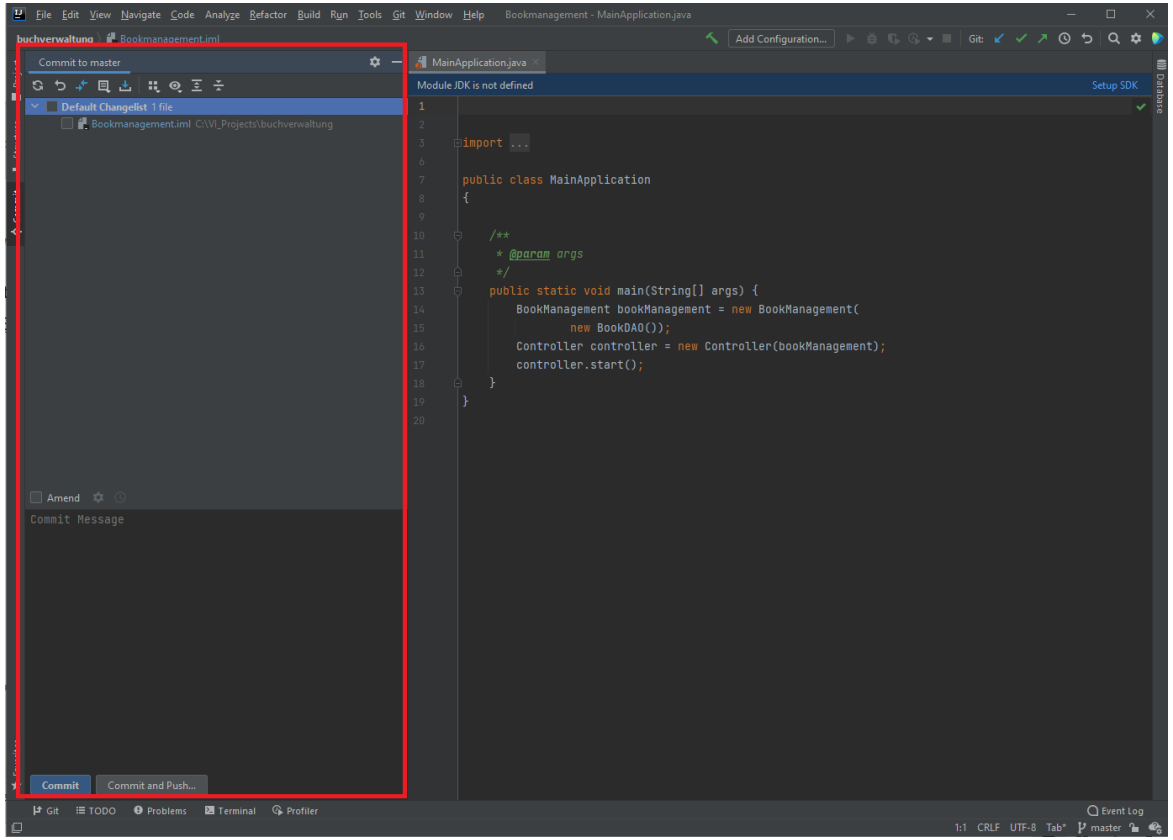
GIT Integration in IntelliJ – Staging

› In die Textbox wird die Commit Message eingetragen

Quelle: https://xkcd.com/1296/

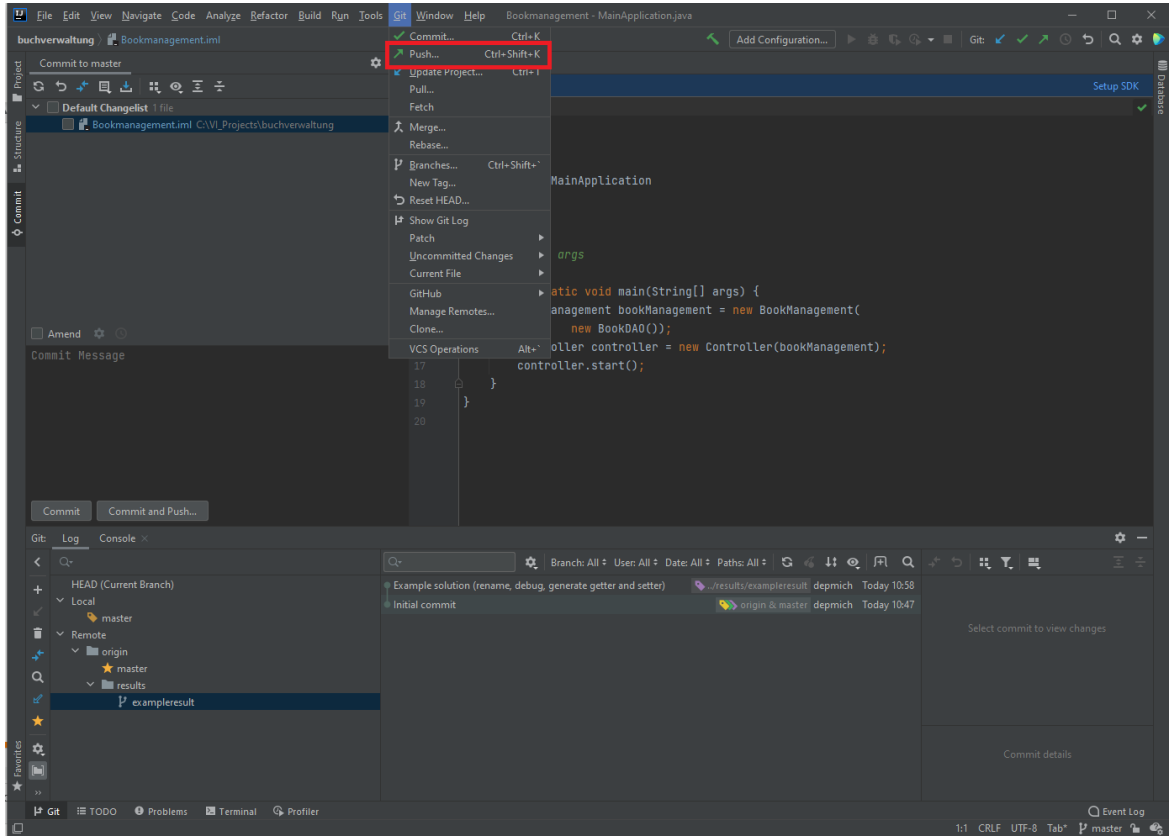
GIT Integration in IntelliJ – Staging

› **Buttons: Es kann nur Commit ausgeführt werden (links), oder Commit und Push zusammen (rechts)**



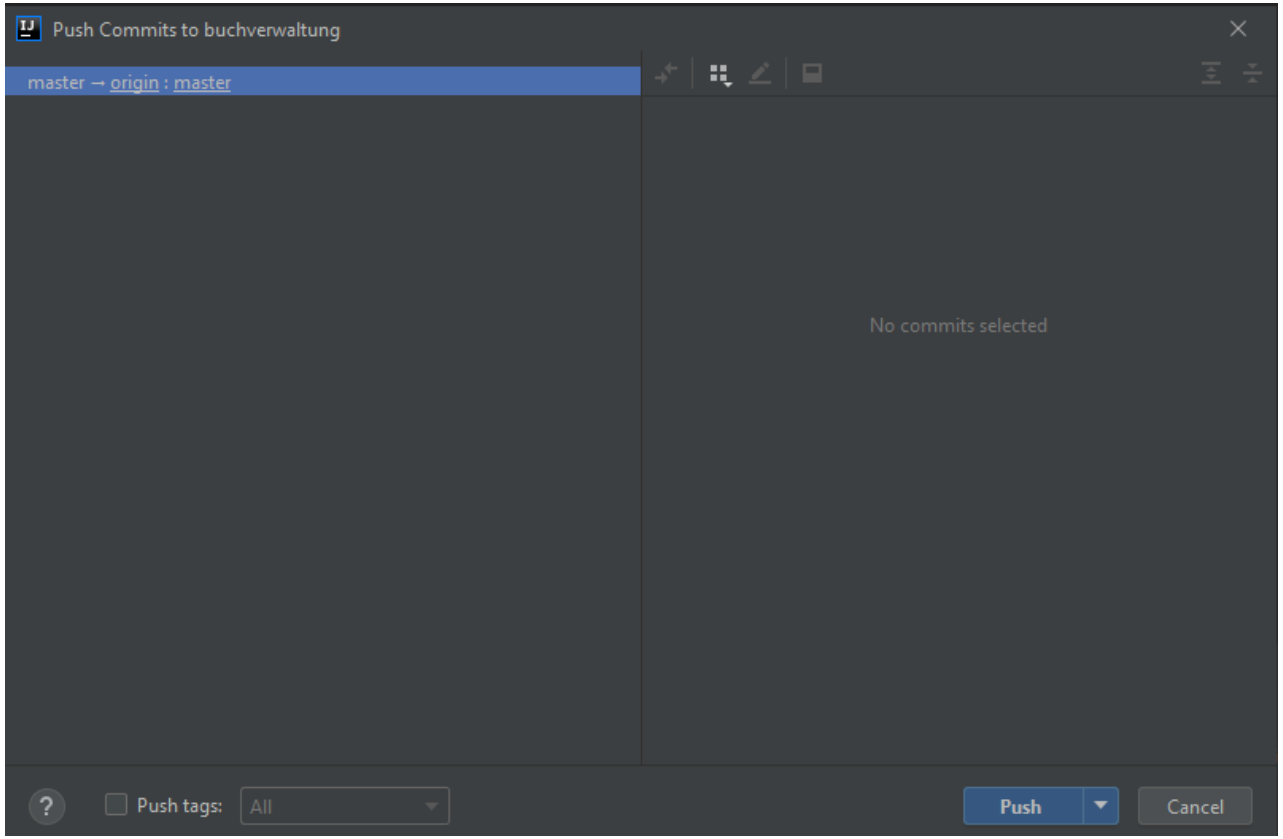
GIT Integration in IntelliJ – Updating

› **Git Push kann über die obere Navigationsleiste ausgeführt werden.**



GIT Integration in IntelliJ – Updating

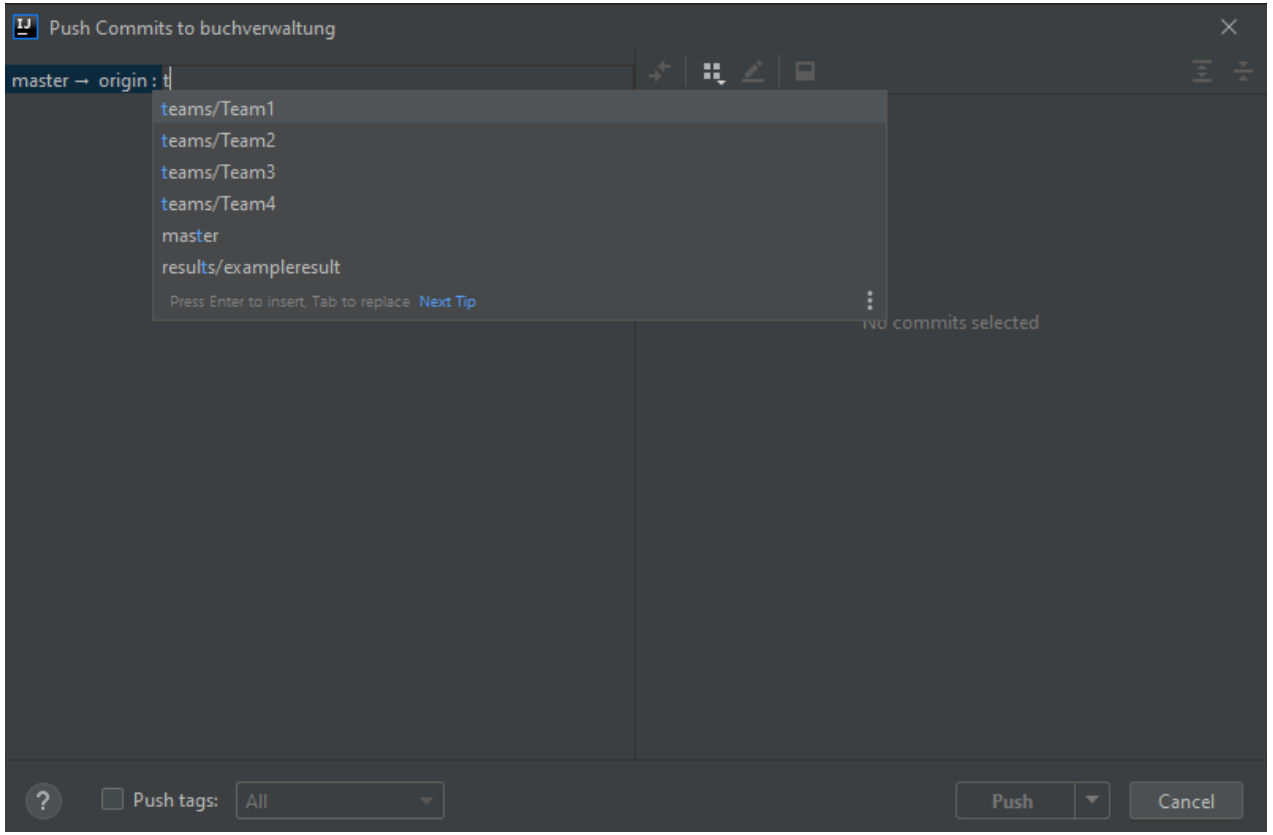
› **Es muss ausgewählt werden in welchen Branch der Push erfolgen soll**



GIT Integration in IntelliJ – Updating

› **Per Autovervollständigung werden existierende Branches vorgeschlagen**

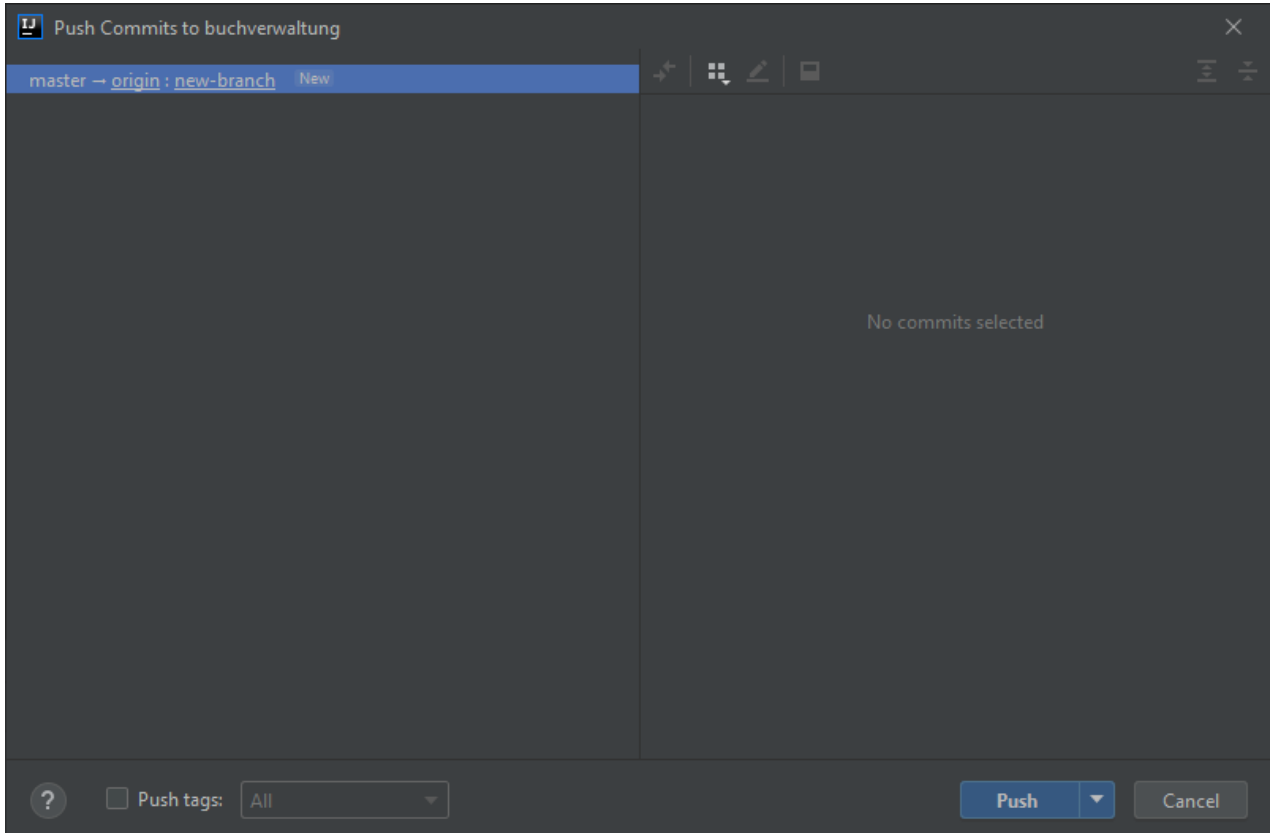
- Ordnernamen können vor den Branchnamen geschrieben und mit “/” getrennt werden



GIT Integration in IntelliJ – Updating

› **Es kann aber auch ein komplett neuer Branch angelegt werden.**

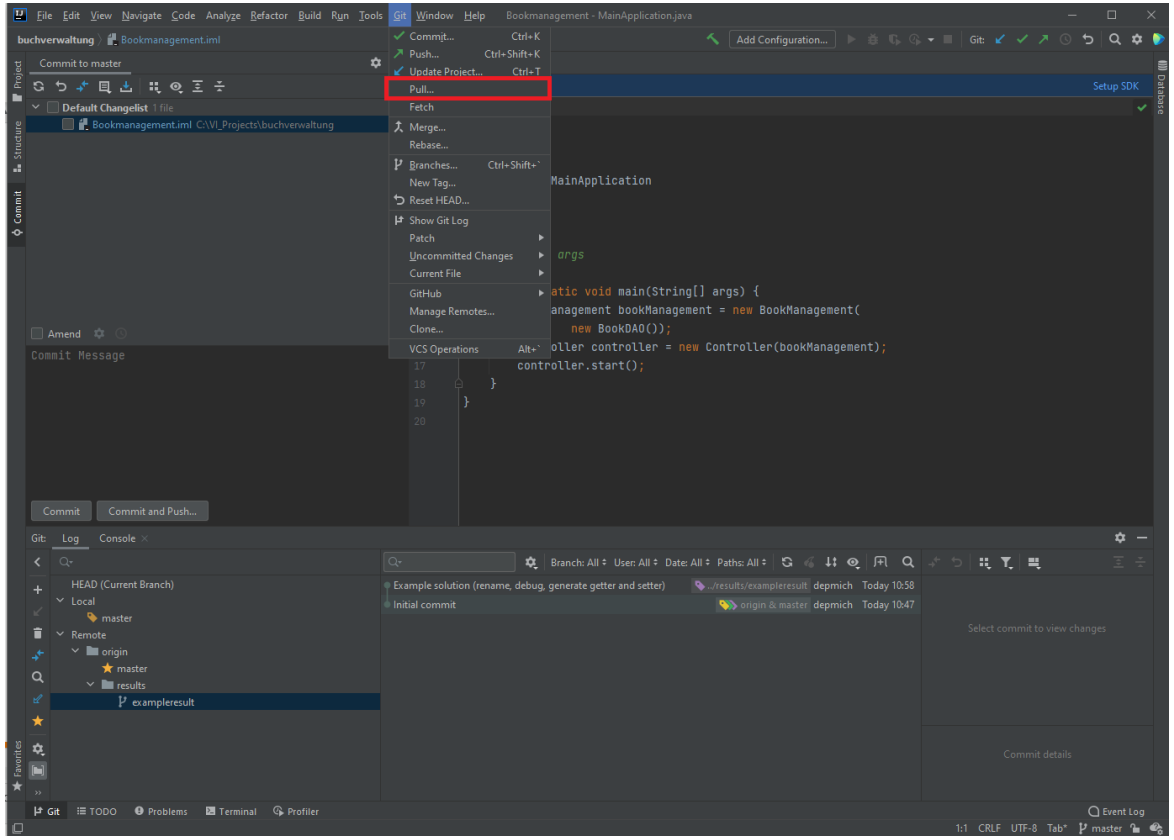
- Der Name des lokalen Branch, und eines remote Branches hängen nicht zusammen!



GIT Integration in IntelliJ – Updating

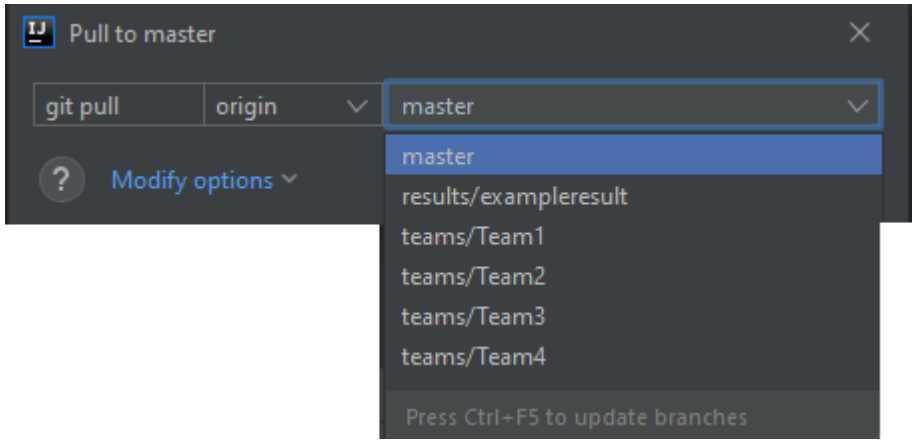
› **Git Pull kann auch über die obere Navigationsleiste ausgeführt werden.**

›



GIT Integration in IntelliJ – Updating

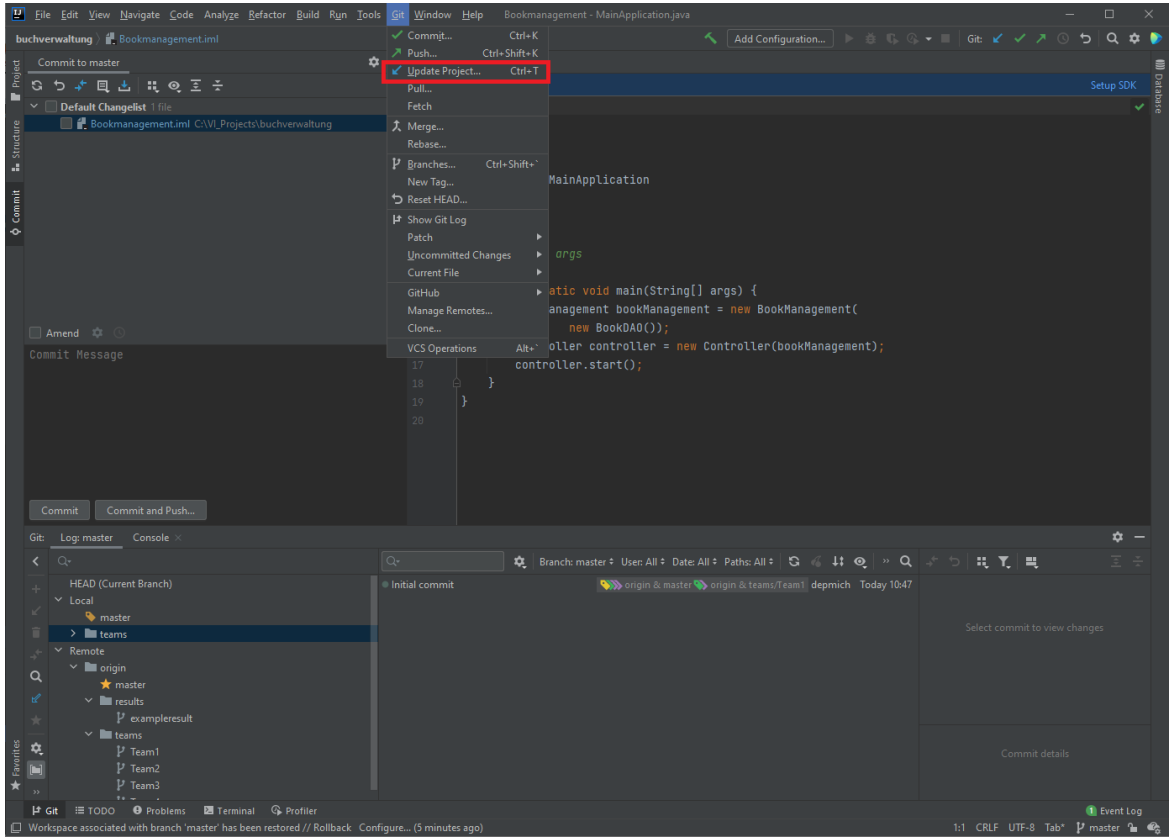
› Unter dem “Pull…” Befehl kann ausgewählt werden, welcher Branch in den aktuellen gepullt werden soll.



VANDERLANDE

GIT Integration in IntelliJ – Updating

› **Update Project, ist “Pull” vom tracked Branch in den Lokalen.**

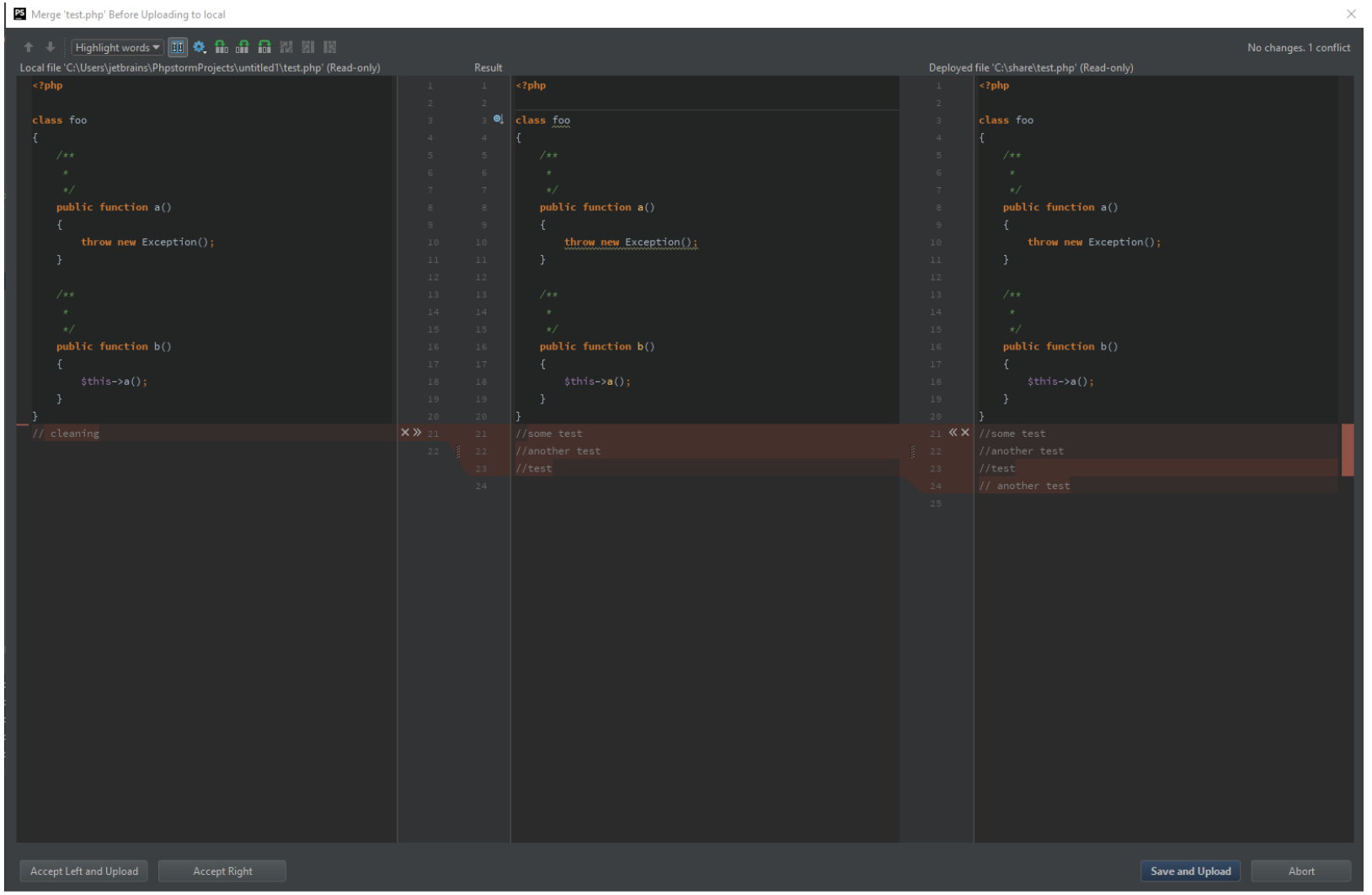


GIT Integration in IntelliJ – Merge

GIT Integration in IntelliJ – Merge

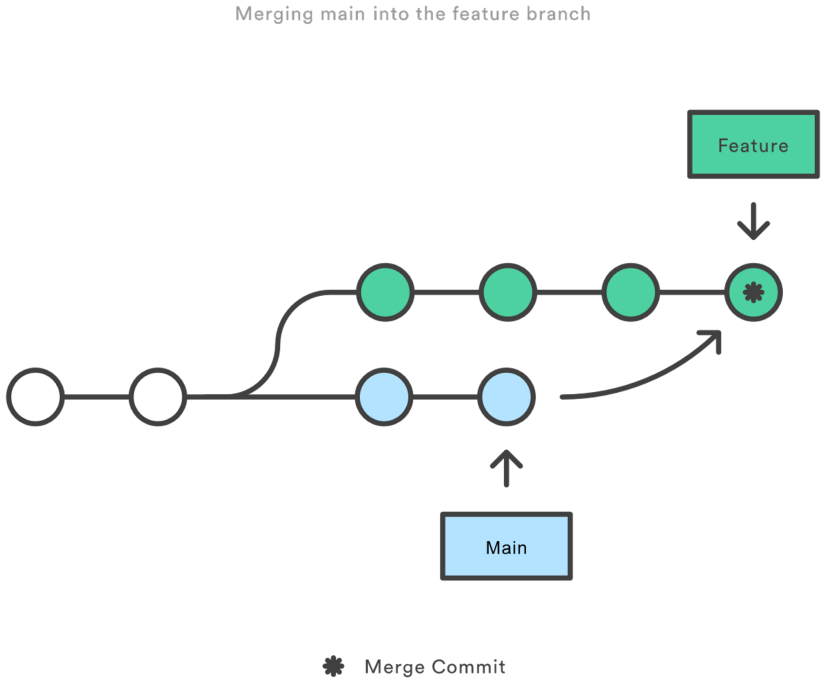
VANDERLANDE

GIT Integration in IntelliJ – Merge



GIT Integration in IntelliJ – Merge vs Rebase

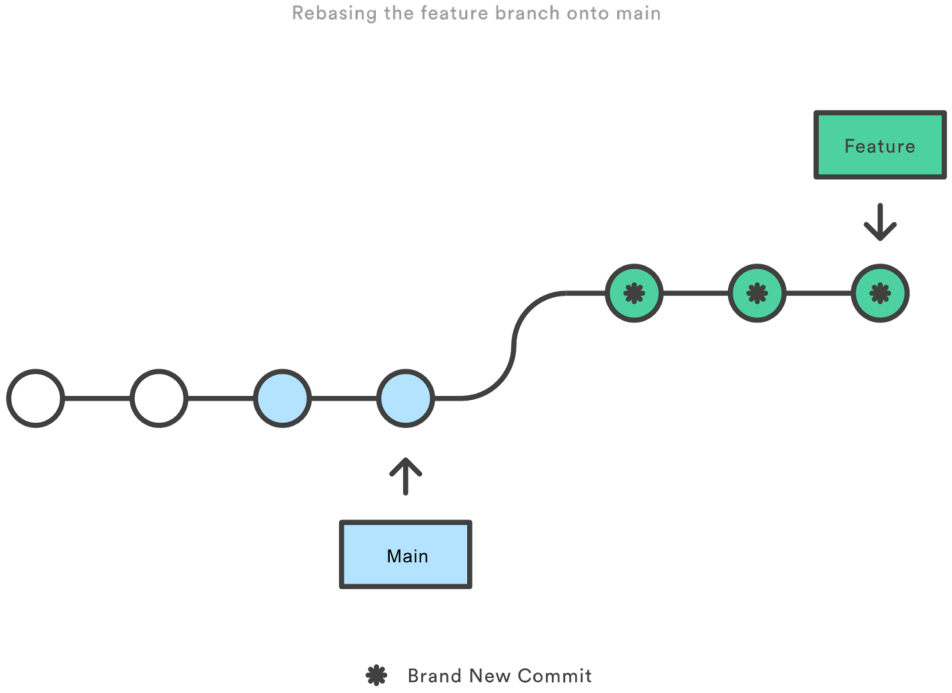
- **Merge und Rebase erfüllen die selbe Aufgabe, änderungen in den Lokalen Branch übernehmen**
  - Merge: Änderungen werden in einem Merge-Commit zusammen gefasst
  - Viele Merge-Commits können unübersichtlich werden, in Bezug auf Code Änderungen



Quelle: Atlassian

GIT Integration in IntelliJ – Merge vs Rebase

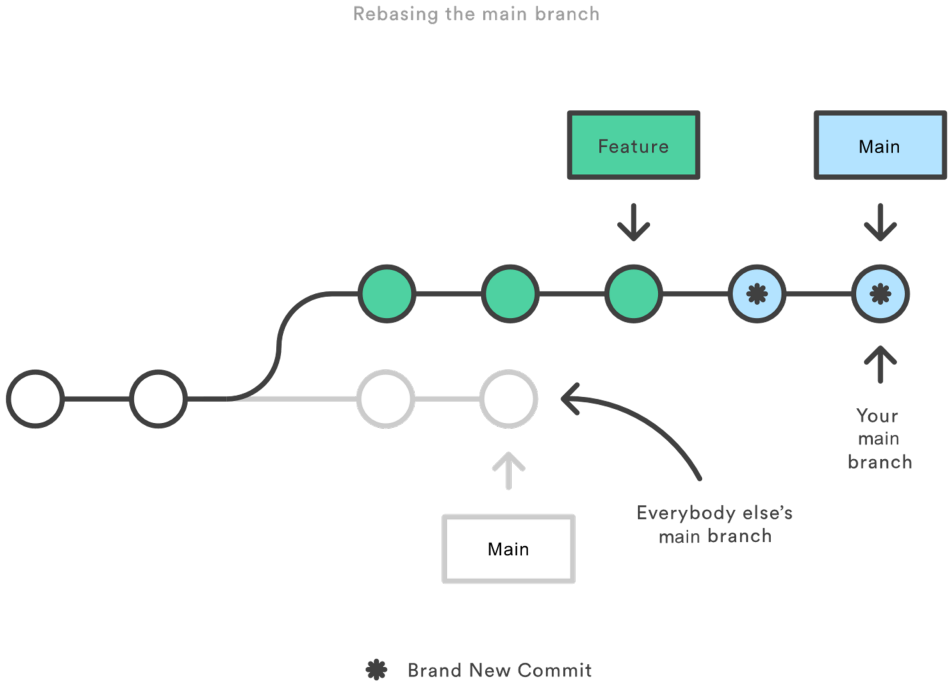
- **Merge und Rebase erfüllen die selbe Aufgabe, änderungen in den Lokalen Branch übernehmen**
  - Rebase: Alle Commits bisher werden an den aktuellen neuen Main Branch angehangen.
  - Dies erzeugt neue lineare Commits, ohne unötige Commits



Quelle: Atlassian

GIT Integration in IntelliJ – Merge vs Rebase

- **Die Goldene Rebase Regel: Warum Rebase gefährlich sein kann**
  - Es darf nie(!) ein öffentlicher Branch, wo andere auch dran arbeiten, in den eigenen rebased werden
  - Dadurch entsteht ein neuer privater Mainbranch, der nicht mit dem der anderen übereinstimmt



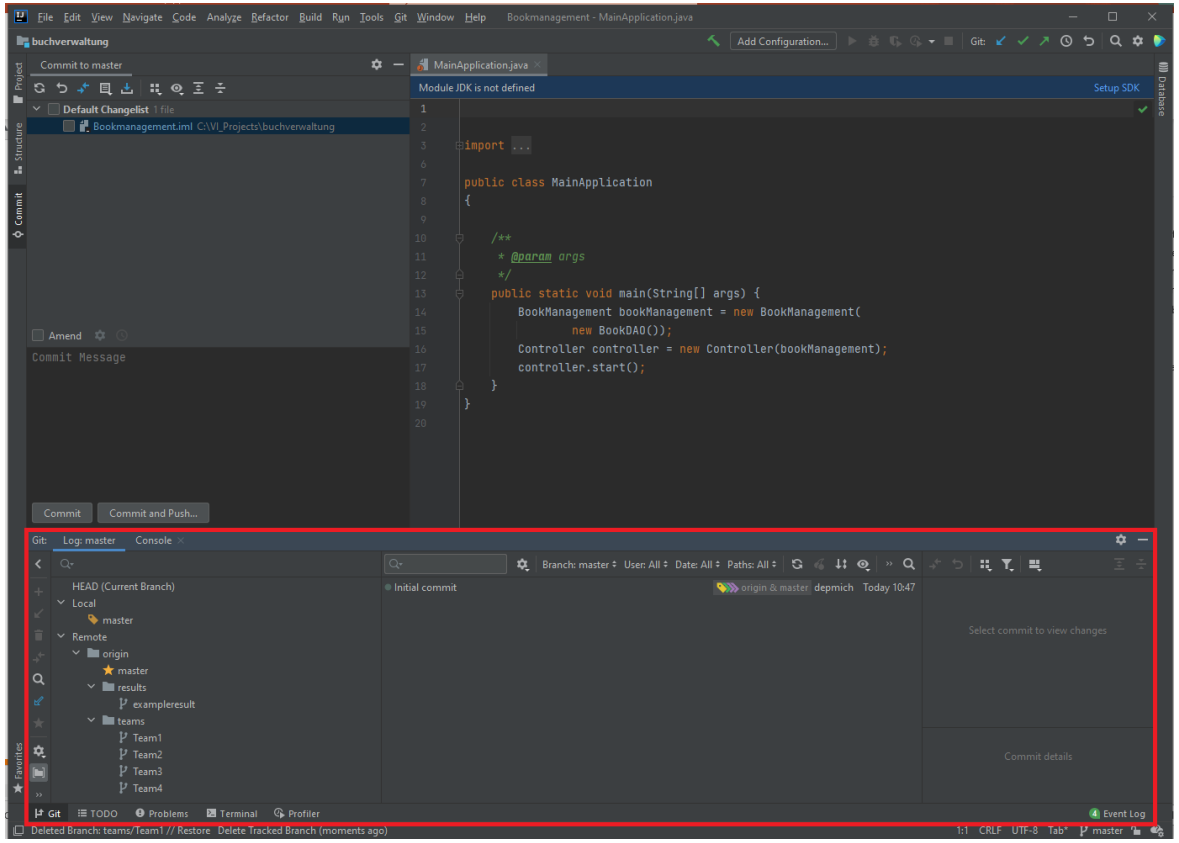
Quelle: Atlassian

GIT Integration in IntelliJ – Merge vs Rebase

- **Wenn ihr euch unsicher seit, macht lieber einen Merge, und löst mögliche Konflikte Gewissenhaft**
  - Wenn ihr euch unsicherseit redet mit dem der den anderen Commit gemacht hat, und sucht gemeinsam eine Lösung.
- **Wenn Ihr ein Feature abgeschlossen habt , macht einen Merge in den Mainbranch**
  - So können alle anderen ab da auf eure änderungen zugreifen, und der Mainbranch ist aktuell

GIT Integration in IntelliJ – Branches

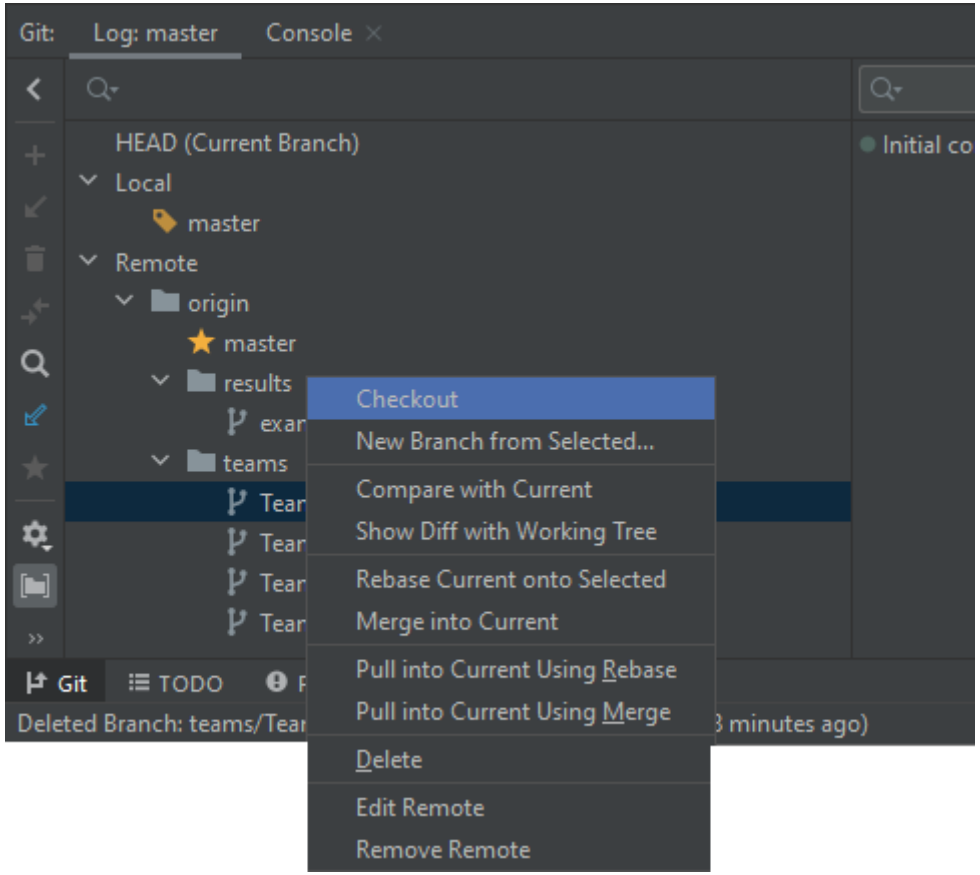
- **Git Browser**
  - Erreichbar über das untere Navigationsregister



GIT Integration in IntelliJ – Branches

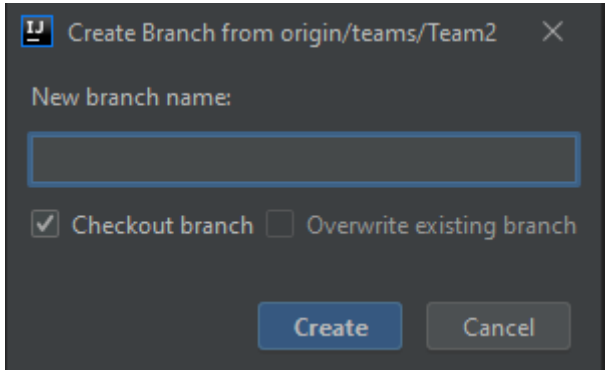
› **Checkout Branch**

- Remote oder Lokalen Branch auswählen, und per Rechtsklick das Context Menü öffnen -> Checkout



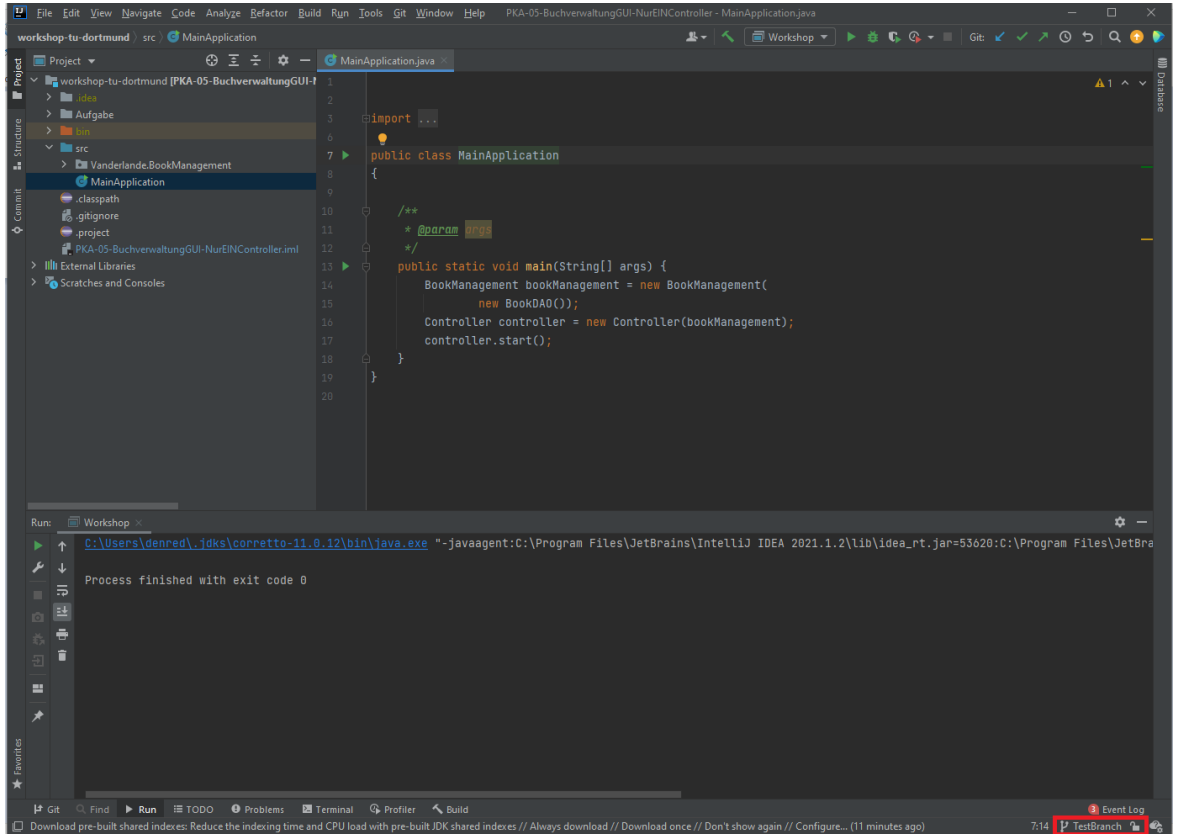
GIT Integration in IntelliJ – Branches

- **Neuen Branch erstellen**
  - Remote oder Lokalen Branch auswählen, und per Rechtsklick das Context Menü öffnen -> New Branch
  - Branchnamen angeben und wählen ob man in den Branch wechseln will



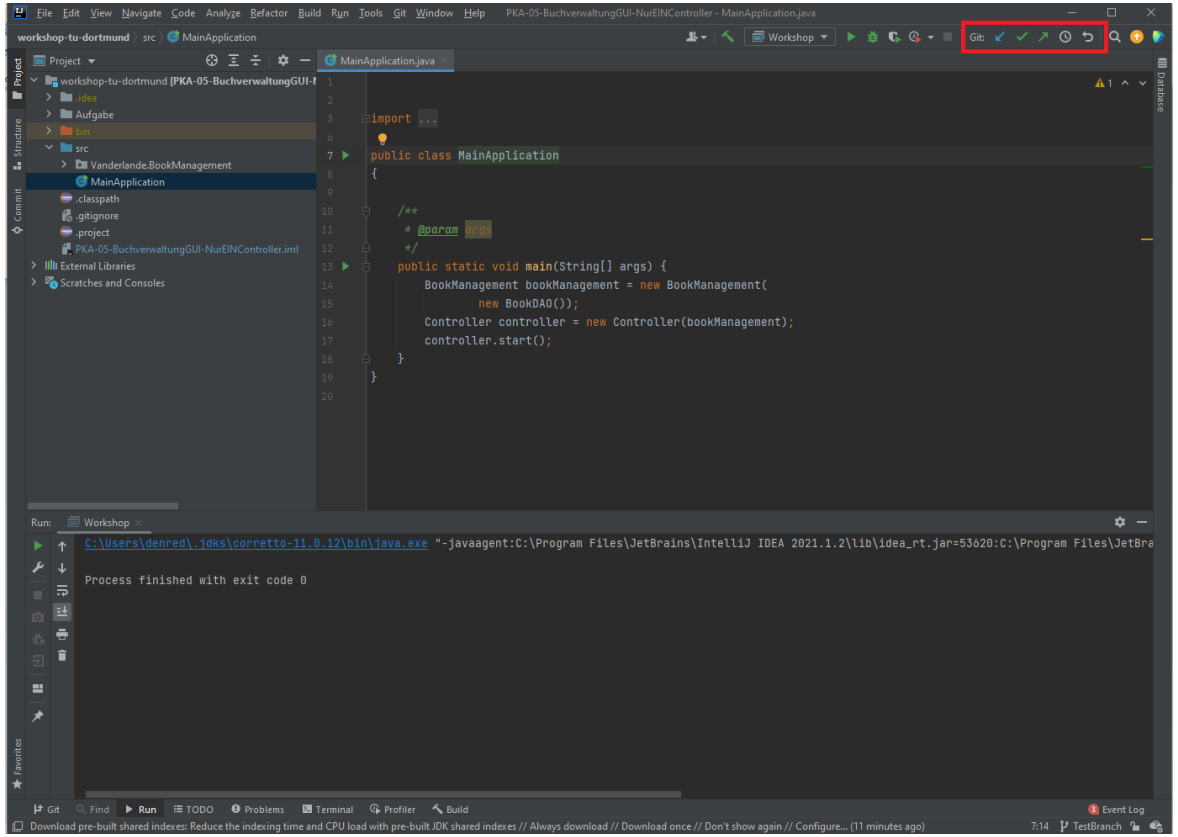
GIT Integration in IntelliJ - Zusätzliche Komponenten

- **Aktiver Git Branch**



GIT Integration in IntelliJ - Zusätzliche Komponenten

- Git Schnellzugriff
- Buttons: Pull/Merge , Commit, Push, History, Rollback



Einführung IntelliJ IDEA : Aufgaben

Aufgabe 1:

- **Wechselt in das Git Browser Fenster und wählt den Branch eurer Gruppe**
- **Dann erstellt ihr einen neuen Branch mit eurem Namen nach folgenden Schema:**
  - teamX-Vorname-Nachname
  - Der Branch soll in einem Unterordner mit dem Namen “teams” liegen
- **Wenn noch nicht beim erstellen gemacht, wechselt per checkout in euren Branch**
- **Unten Rechts im IntelliJ UI sollte nun euer Branch als aktiver Branch angezeigt werden.**
- **Verwendet Push auf euren lokalen Branch und fügt ihn unterhalb von “teams” im remote ein**

Einführung IntelliJ IDEA : Aufgaben

Aufgabe 2:

- **Öffnet die Datei src>MainApplication**
- **Fügt in der aller ersten Zeile folgendes hinzu:**
  - //Editor: Vorname Nachname
- **Staged eure Änderungen, und führt commit und push (auf euren Branch) aus**

Entwickeln im Unternehmen

Wie entwickeln wir im Unternehmen

Was unterscheidet sich zum privaten Entwickeln

- Nicht nur ihr arbeitet mit eurem Code
- Arbeiten mit Fremdcode
- Anwendung von Source-Control
- Bugtracking/Ticketsysteme
- Coding Standards

Was gilt zu beachten wenn ich im Unternehmen entwickele

- Sprechende Funktionsnamen
- Sprechende Variablennamen
- Verständliche Kommentare
- Logische Klassen/Ordnerstruktur
- Logische Commit-Texte

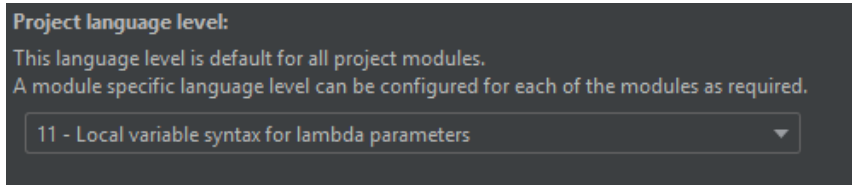
MOVING YOUR BUSINESS FORWARD

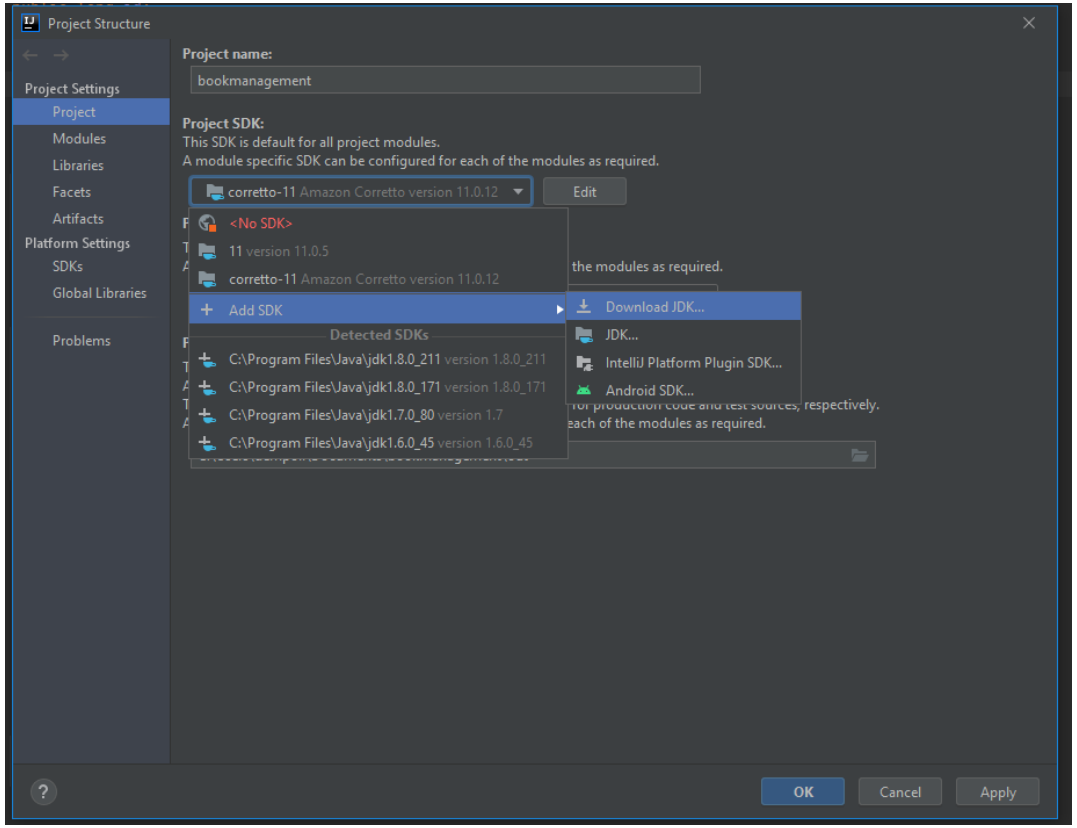
Projekt kompilieren

Auswahl der richtigen SDK

Projekt kompilierbar machen

1. Datei -> Projektstruktur
2. Projekteinstellungen -> Projekt
3. Projekt SDK -> Dropdown
    1. SDK hinzufügen
    2. Download JDK
    3. Version 11, Amazon Corretto
4. Projekt Sprachlevel
    1. Dropdown -> 11

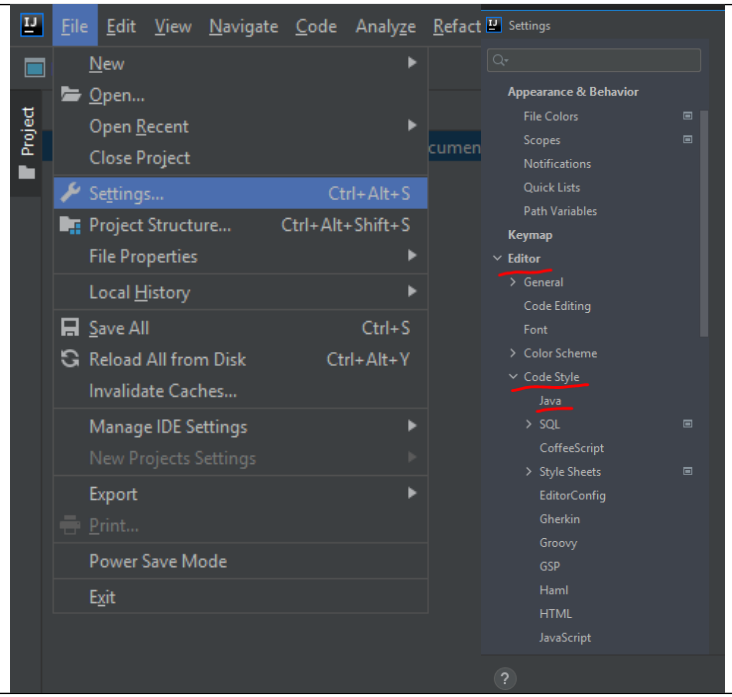




Coding Styles anpassen

Wie passt man in IntelliJ die Code Styles an

1. Geht auf File -> Settings
2. Dort auf Editor -> Code Style -> Java



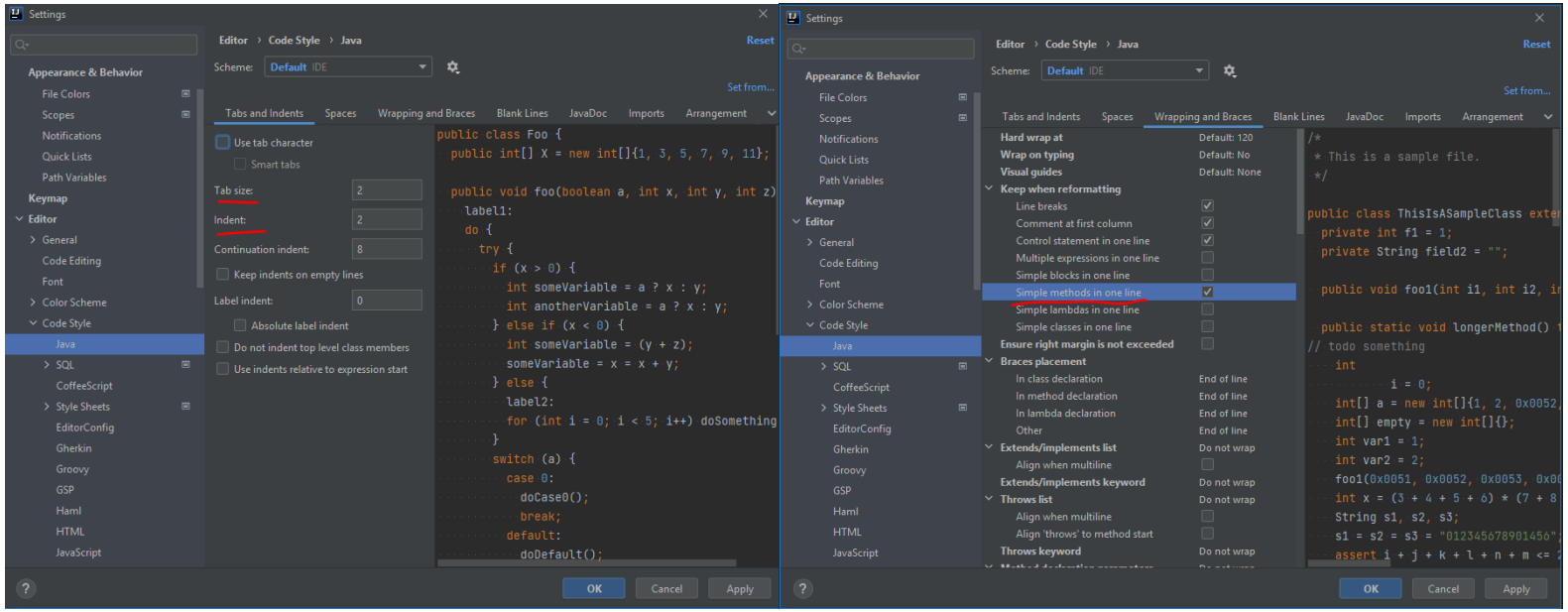
Coding Style : Aufgabe

Aufgabe:

- Ändert die Anzahl der Tabs auf 2
- Ändert die Einrückung auf 2
- Einfache Methoden sollen in einer Zeile angezeigt werden

Coding Style : Aufgabe - Lösung

Lösung:



Generatoren in IntelliJ

Wozu werden diese verwendet

Wozu verwende ich Generatoren

- Schnelle Erstellung von Gettern und Settern
- Constructorgenerierung
- Implementieren von Interfaces und Vererbung, Overrides

Wie verwende ich einen Generator (Beispiel)

1. Gehe in die Klasse Book
2. Rechtsklicke in die Klasse (Alt+Einfg)
3. Getter und Setter auswählen
4. Generiere einen Getter und Setter für den Titel und den Preis

Renaming in IntelliJ

Wozu werden diese verwendet

Warum werden Dinge renamed?

- Nach der Objekterstellung kann es schon mal vorkommen, dass ein anderer Name logisch wäre
- Korrektur von Rechtschreibfehlern

Renamefunktion aufrufen:

1. markiere das Objekt im Code
2. Kontextmenü -> refactor
3. rename auswählen
4. Neuen Namen eingeben, Einstellungen anpassen
5. success

Debugging

Was ist ein Debugger und wozu wird dieser verwendet

Was ist ein Debugger

- Anwendung zum diagnostizieren und auffinden von Fehlern

Wozu verwende ich einen Debugger

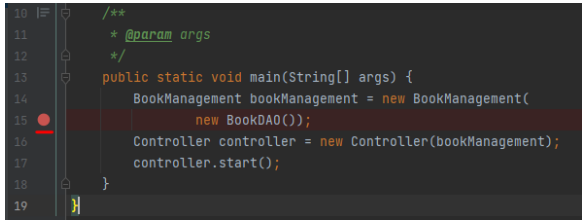
- Steuerung des Programmablaufes
- Auslesen von Daten in Variablen
- Auslesen/Modifizieren des Heap und Stack Speichers

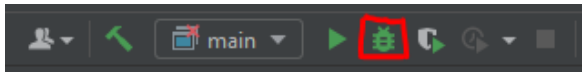
Debugging – Allgemeine Funktion

Debugfunktion in IntelliJ

Wie debugge ich in IntelliJ ?

1. Setzen eines Breakpoints(Strg+F8)
2. Debugging Symbol(Shift+F9) statt Play
3. Debugging-Steuerleiste
    1. Step-Over
    2. Step-Into
    3. Step-Out
    4. Jump-to-cursor
4. Hovern über Variable





Fragen?

VANDERLANDE

Feedback

- Hat der Umfang für euch gepasst?
- Wurden die Themen zu detailiert bzw. zu oberflächlich behandelt?
- Wie war die Schwierigkeit?
- Fühlst du dich gut aufs SoPra vorbereitet?
- Wie hat die Gruppenarbeit funktioniert?
- Würdest du einen Folgetag besuchen?

VANDERLANDE

MOVING YOUR BUSINESS FORWARD