FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Einführung in IntelliJ IDEA

Prof. Dr. Bodo Kraft
Benjamin Poniatowski
Jennifer Schwan

Lernziele
Einführung in IntelliJ IDEA

In diesem Kapitel lernen wir …

… wie sie eine integrierte Entwicklungsumgebung nutzen

… wie man damit ein Java-Projekt und Java-Klassen erstellt

… wie man Programme compiliert und ausführt

… wie man den Debugger verwendet

… und Vieles mehr…

Die Folien verwenden IntelliJ IDEA - Ultimate 2020.2

Kernfunktionalität von Entwicklungsumgebungen
Arbeiten mit einer integrierten Entwicklungsumgebung

Software-Entwicklungsumgebungen unterstützen den Entwickler

- Editieren
  - Quellcode erstellen
  - Quellcode formatieren
  - Vervollständigung
  - Annotationen mit API-Dokumentation
- Debuggen
  - Code schrittweise ausführen
  - Fortschritt im Editor steuern, Breakpoints
  - Werte von Variablen abfragen
  - (hot replace)
- Übersetzen
  - Projektabhängigkeiten verwalten
  - Varianten definieren
  - (inkrementell)
- Projekteinstellungen verwalten

Erweiterte Funktionalität von Entwicklungsumgebungen

Arbeiten mit einer integrierten Entwicklungsumgebung

Software-Entwicklungsumgebungen unterstützen den Entwickler

- Konfigurationsmanagement
  - Verwaltung der Revisionen
  - Verwaltung der Varianten
- Codegenerierung
  - Wizards
  - User-Inferface *Editoren*
  - Entwurfspezifikation
- Refactoring - Reorganisation von Sourcecode
  - Umbenennen
  - Verschieben
- weitere …

Features von IntelliJ IDEA
Arbeiten mit einer integrierten Entwicklungsumgebung

Rahmenwerk zur Integration verschiedener Anwendungen

- Java Entwicklungsumgebung inkl. Debugger
- Git, SVN
- Maven
- Android Entwicklungsumgebung
- JavaScript Entwicklungsumgebung (nur Ultimate Edition)
- Datenbanken, SQL (nur Ultimate Edition)

➔ Anwendungsgebiet hier: Java-Entwicklung

Installation / Verschiedene Versionen
Arbeiten mit IntelliJ IDEA

Community Edition (kostenfrei) und Ultimate Edition

- Download unter: https://www.jetbrains.com/de-de/idea/download/
- Ultimate Edition ist für Studierende kostenfrei nach Registrierung mit FH E-Mail unter https://www.jetbrains.com/de-de/community/education/#students

Ein neues Projekt erstellen
Arbeiten mit IntelliJ IDEA

Beim ersten Start von IntelliJ IDEA wird man gefragt, welche Plugins installieren werden sollen. Hier muss man gar nichts und darf man alles auswählen.
Bei Verwendung der Ultimate Version muss noch die Aktivierung per JetBrains-Konto erfolgen.

Danach sollte folgender Dialog angezeigt werden:

Ein neues Projekt erstellen
Arbeiten mit IntelliJ IDEA

Um ein neues Projekt zu erstellen sind verschiedene Schritte nötig:

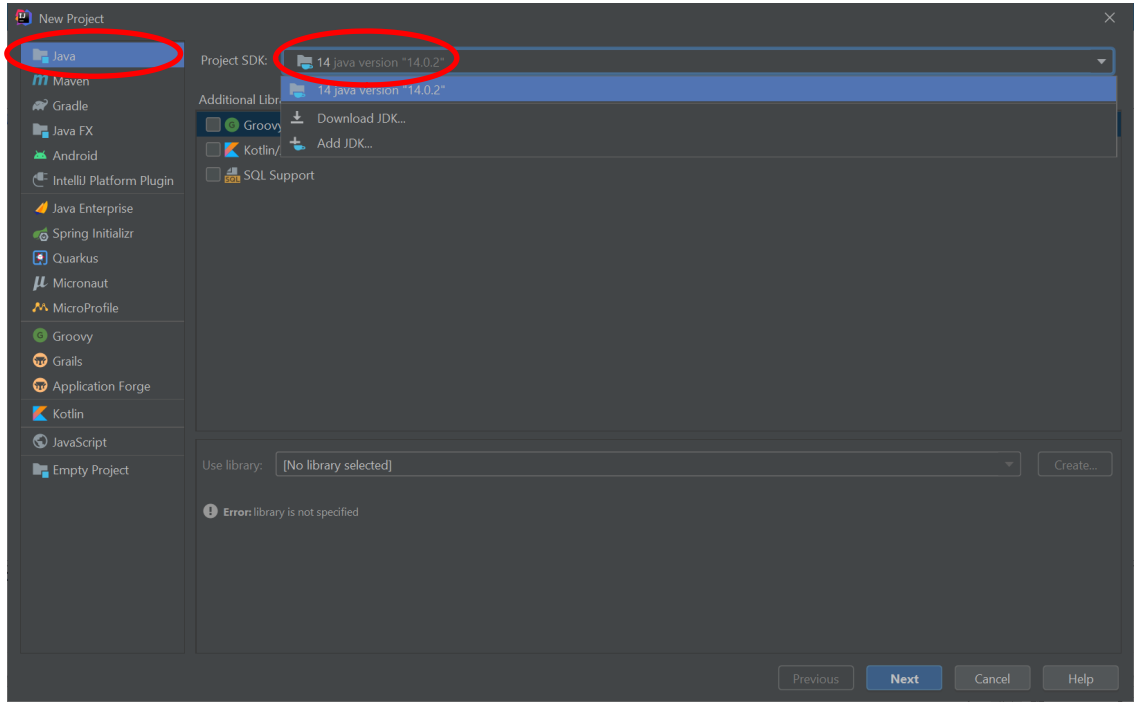
1. Wählen der Projektart
   - Bei Java: Auswahl der JDK
2. Template verwenden oder nicht
3. Projektnamen und Speicherort festlegen

- Klick auf **`New Project`**

Die Projektart wählen
Ein neues Projekt erstellen

Im Folgenden sind die relevanten Stellen auf dem Bild rot markiert

1. Als Projektart *Java* auswählen
2. Ein SDK auswählen.
   Falls IntelliJ nicht automatisch ein Java SDK (JDK) findet, kann man über das Kontextmenü direkt aus IntelliJ heraus ein JDK downloaden oder aus dem Dateisystem ein bereits installiertes JDK hinzufügen.



Man erreicht dieses Menü auch über:

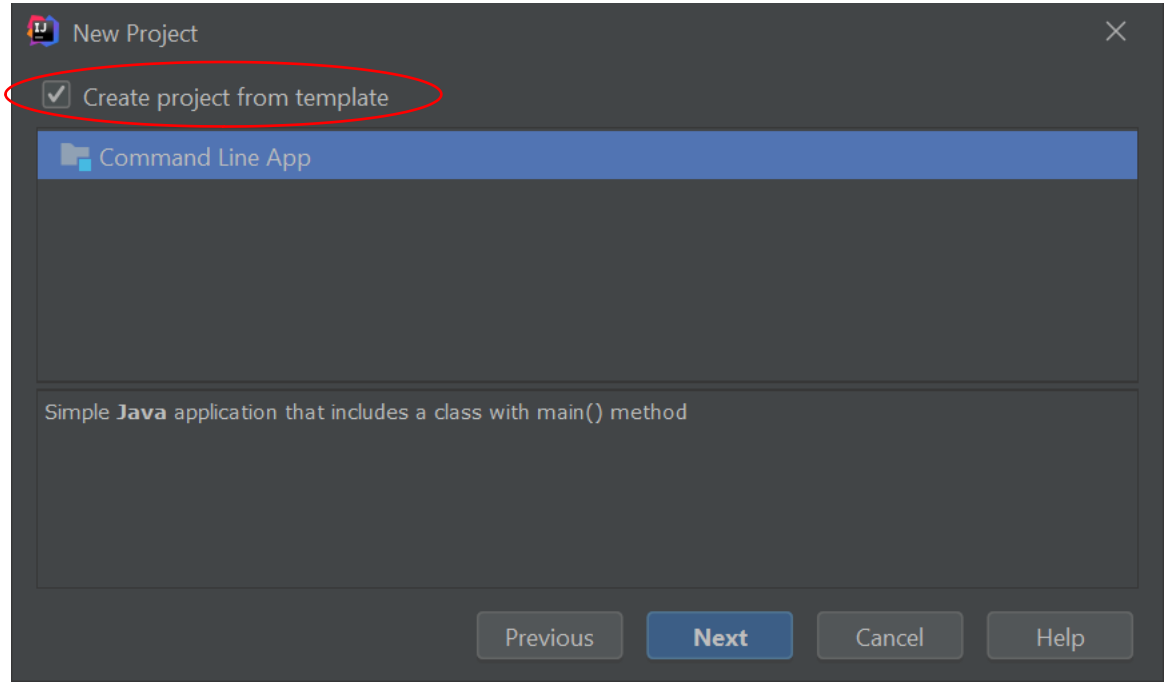
`File` ✉ `New` ✉ `Project`

Template verwenden (oder nicht)
Ein neues Projekt erstellen

Nun hat man die Wahl ob man ein **Template** verwenden möchte.

Ohne Template (Haken im Fenster wählen) wird einfach ein leeres Projekt erstellt.

Standardmäßig steht das Template **`Command Line App`** zur Verfügung, welches automatisch eine Main-Klasse mit leerer main-Methode anlegt.



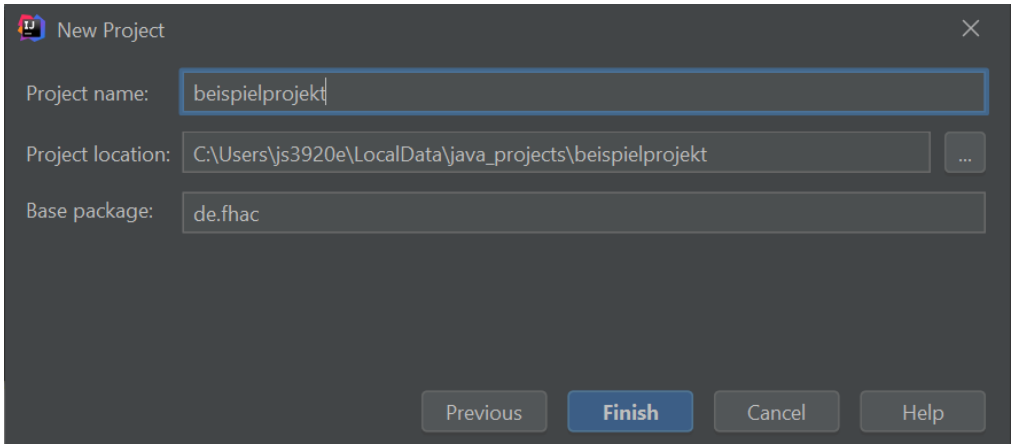
Projektnamen vergeben und Speicherort wählen
Ein neues Projekt erstellen

Als Letztes muss ein Projektname vergeben werden.
Der Name sollte:

- Keine Leerzeichen enthalten
- Keine Sonderzeichen enthalten
- Passend zum Einsatzzweck sein

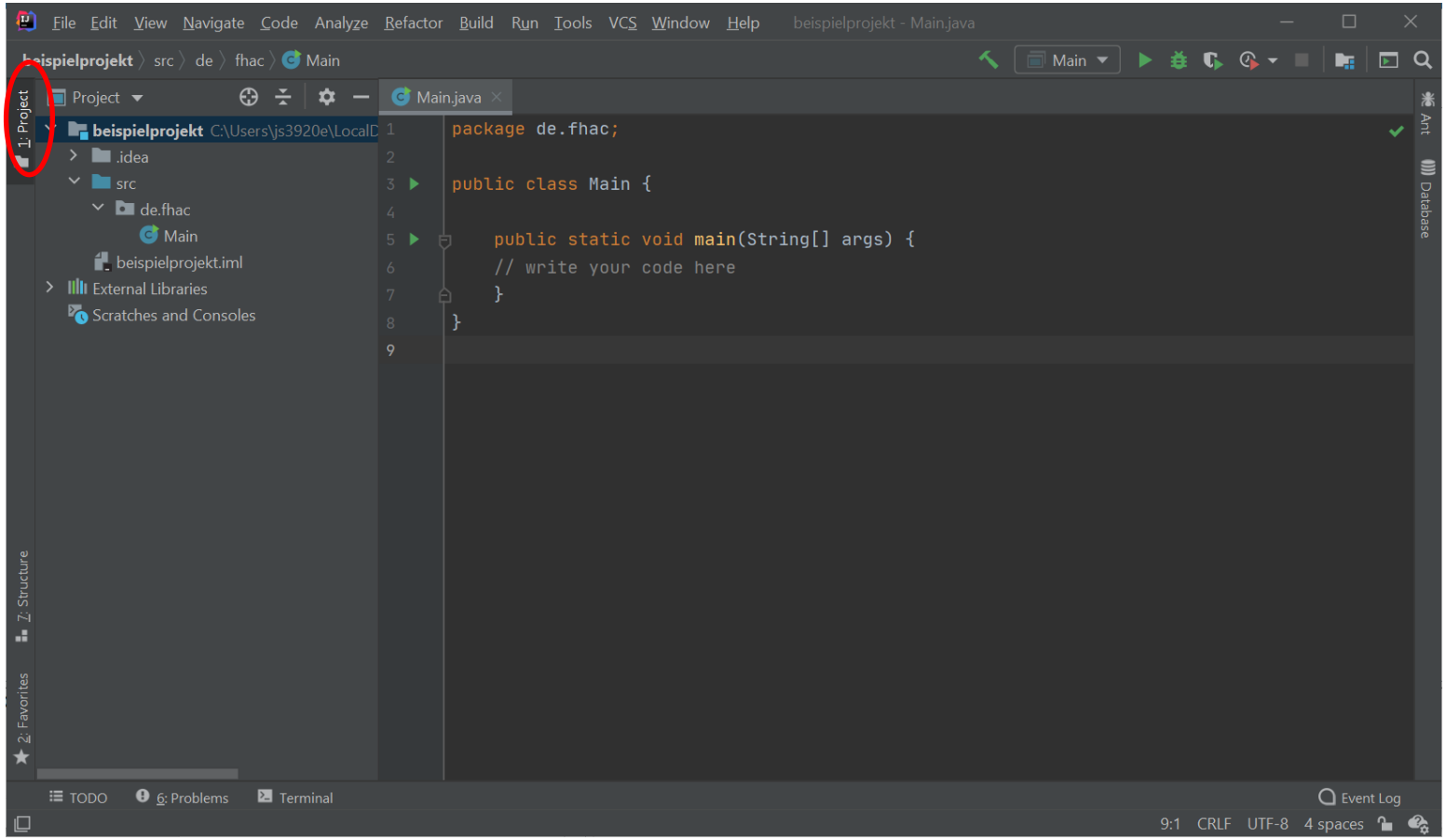
Der Name entspricht optimalerweise dem Ordnernamen unter dem das Projekt abgelegt wird.
Am Besten legt man auf dem Computer einen Ordner an unter dem man alle zukünftigen Projekte ablegen wird.

Als Base package kann man z.B. de.fhac angeben.



Die Entwicklungsansicht
Einführung in IntelliJ IDEA

Wenn das neue Projekt fertig erstellt ist oder ein vorhandenes Projekt geöffnet wird (**`File` ✉ `Open`**) wechselt IntelliJ in die Projektansicht:



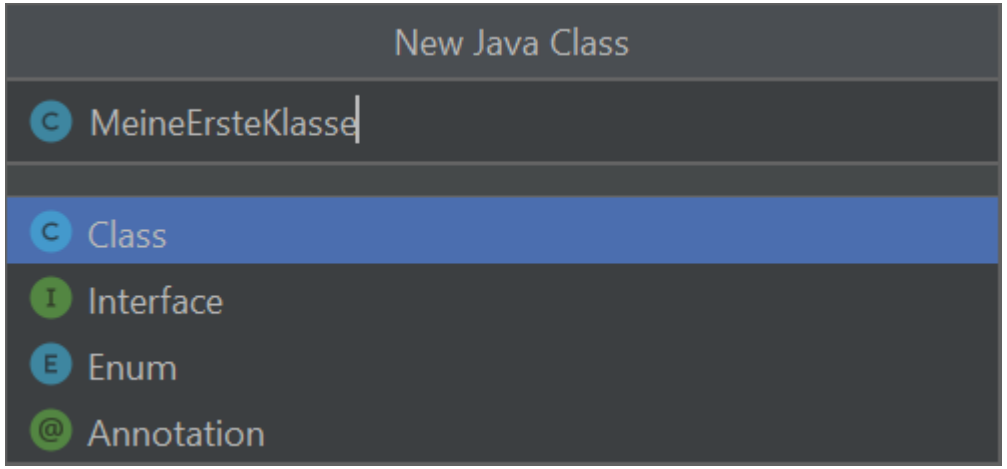
Wenn die linke Leiste ausgeblendet ist hilft ein Klick auf die markierte Stelle.

Eine neue Java-Klasse erstellen
Einführung in IntelliJ IDEA

Um eine neue **Klasse zu erstellen** klicken Sie ihr Projekt an und wählen Sie aus dem Menü **`File` ✉ `New` ✉ `Java Class`**

Danach erscheint ein kleines Fenster, in dem Sie den Klassennamen eingeben müssen.

Der Klassenname sollte
- Mit einem Großbuchstaben beginnen
- keine Sonderzeichen enthalten
- keine Leerzeichen enthalten

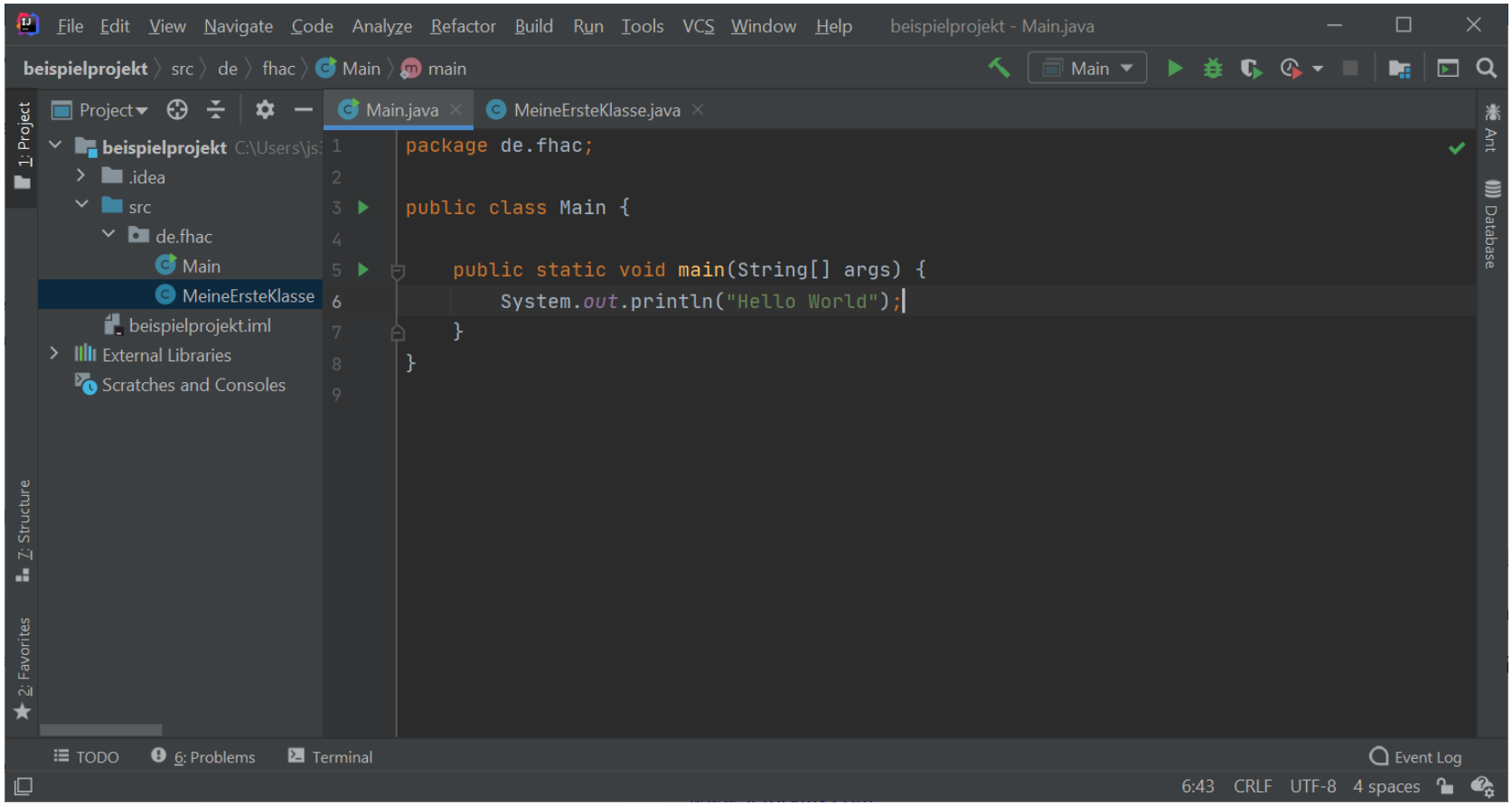


Drücken Sie **`Enter`** auf der Tastatur um die Klasse erstellen zu lassen

Die Entwicklungsumgebung
Einführung in IntelliJ IDEA

Auf der linken Seite ist die Projektansicht, die die Pakete und Klassen anzeigt.

In der Mitte ist der Editor, in dem die Klassen verändert werden können.

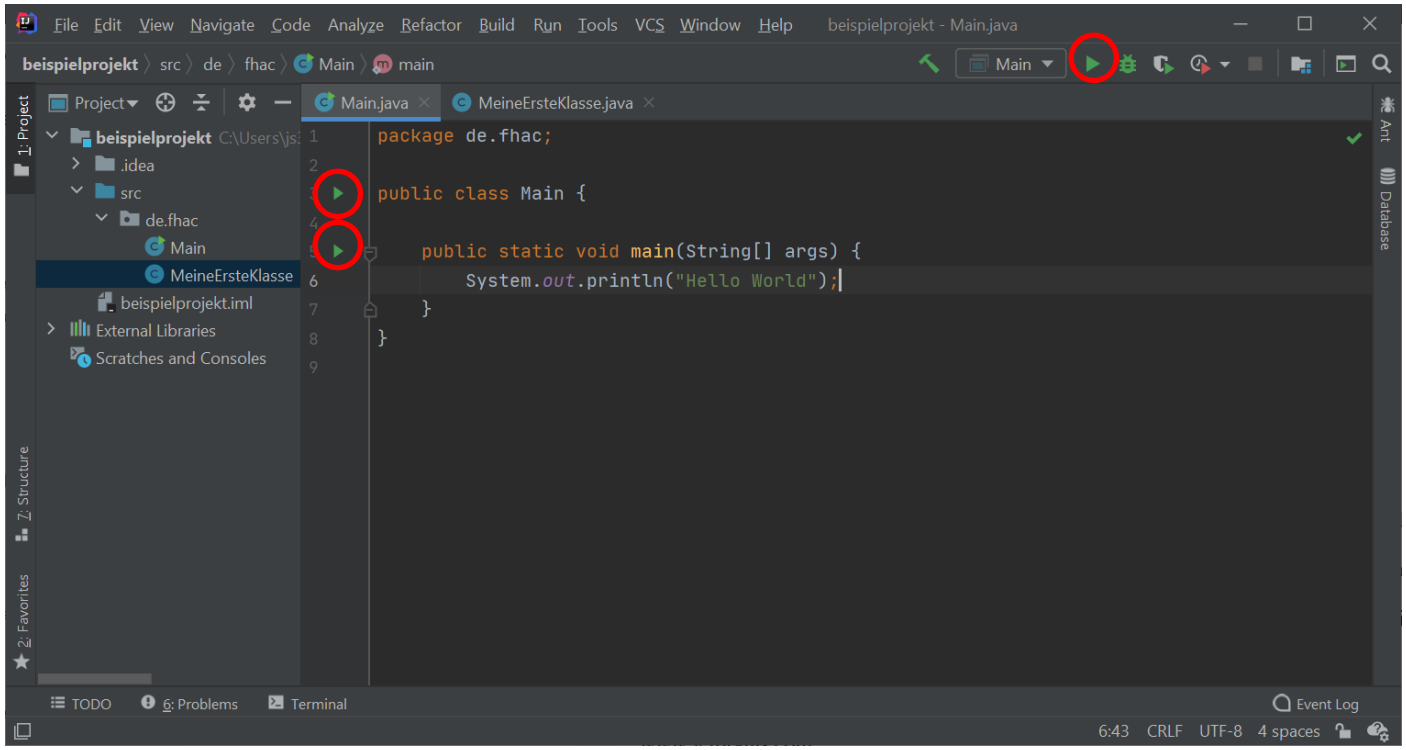


Kompilieren & Ausführen
Einführung in IntelliJ IDEA

Der **Programmstart** erfolgt durch

- Rechtsklick auf die Klasse mit main-Methode
- Im Kontextmenü **`Run 'Main'`**
  (**`Main`** ist hier der Name der Klasse mit main-Methode)

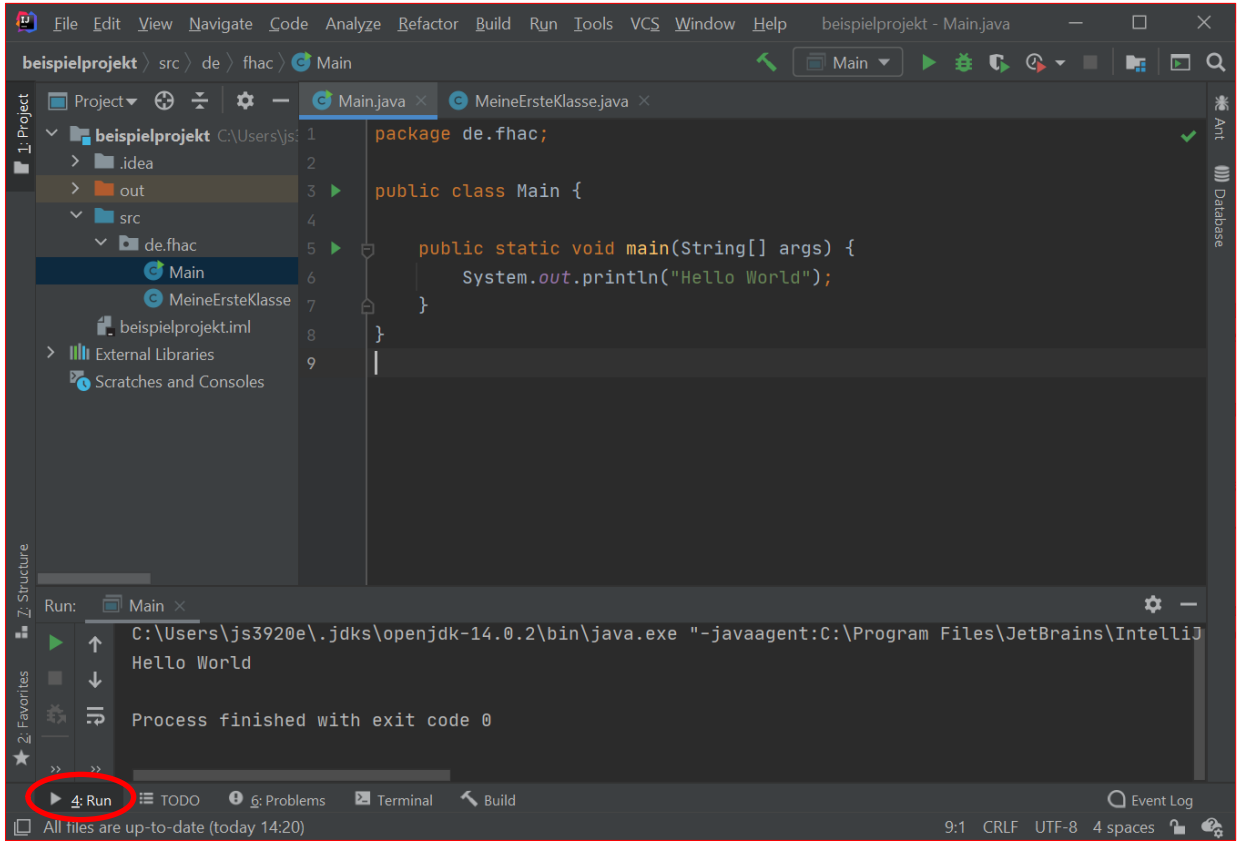
Alternativ über Tastenkürzel **`STRG + SHIFT + F10`** oder durch Klick auf eines der markierten grünen Play-Symbole



Das Ausgabefenster
Einführung in IntelliJ IDEA

Im unteren Bereich sollte nun das **"Run"-Fenster** zu sehen sein.

Falls das Unterfenster nicht erscheinen sollte, kann es auch durch Klick auf den unten markierten Reiter aufgerufen werden.



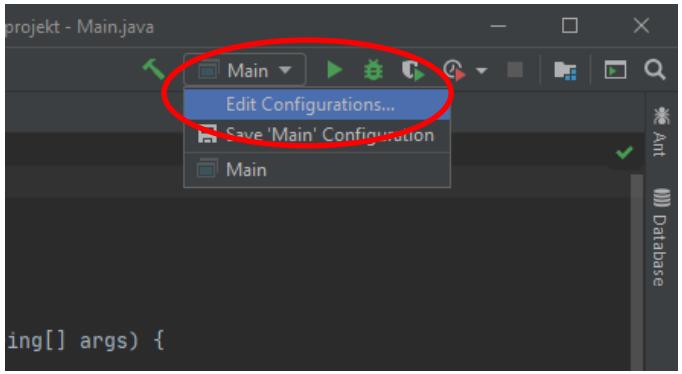
Die erste Zeile zeigt den Befehl zum Starten des Programmes.

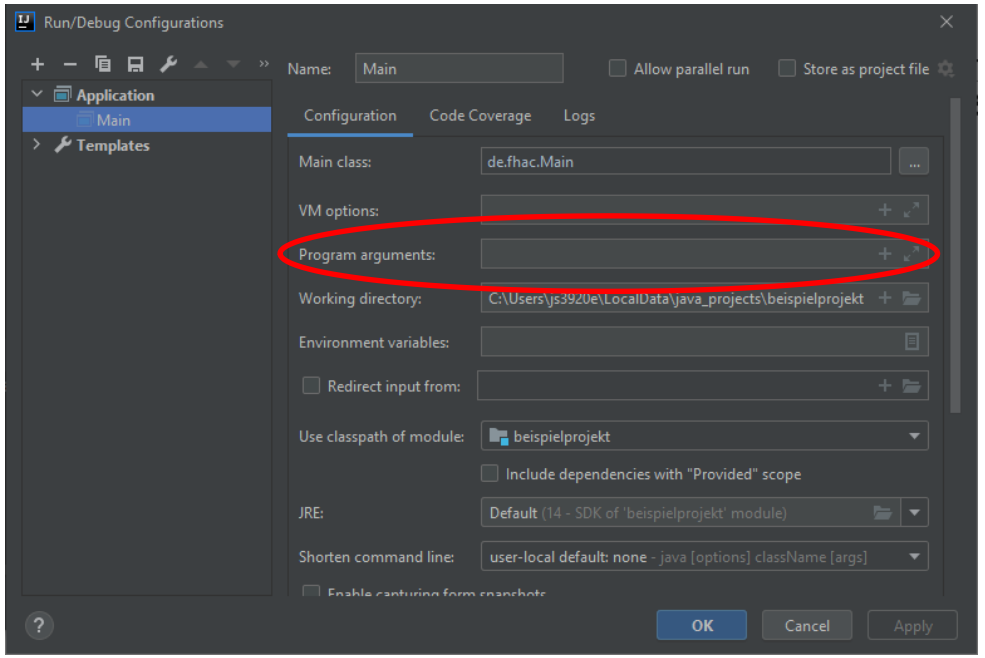
Darunter findet sich die **Konsolenausgabe** des Programmes (in diesem Fall `Hello World`).

Zuletzt folgt die Meldung `Process finished` inklusive exit code.

Konsolenparameter übergeben
Einführung in IntelliJ IDEA

Konsolenparameter (`args` der main-Methode) lassen sich in IntelliJ wie folgt einstellen:



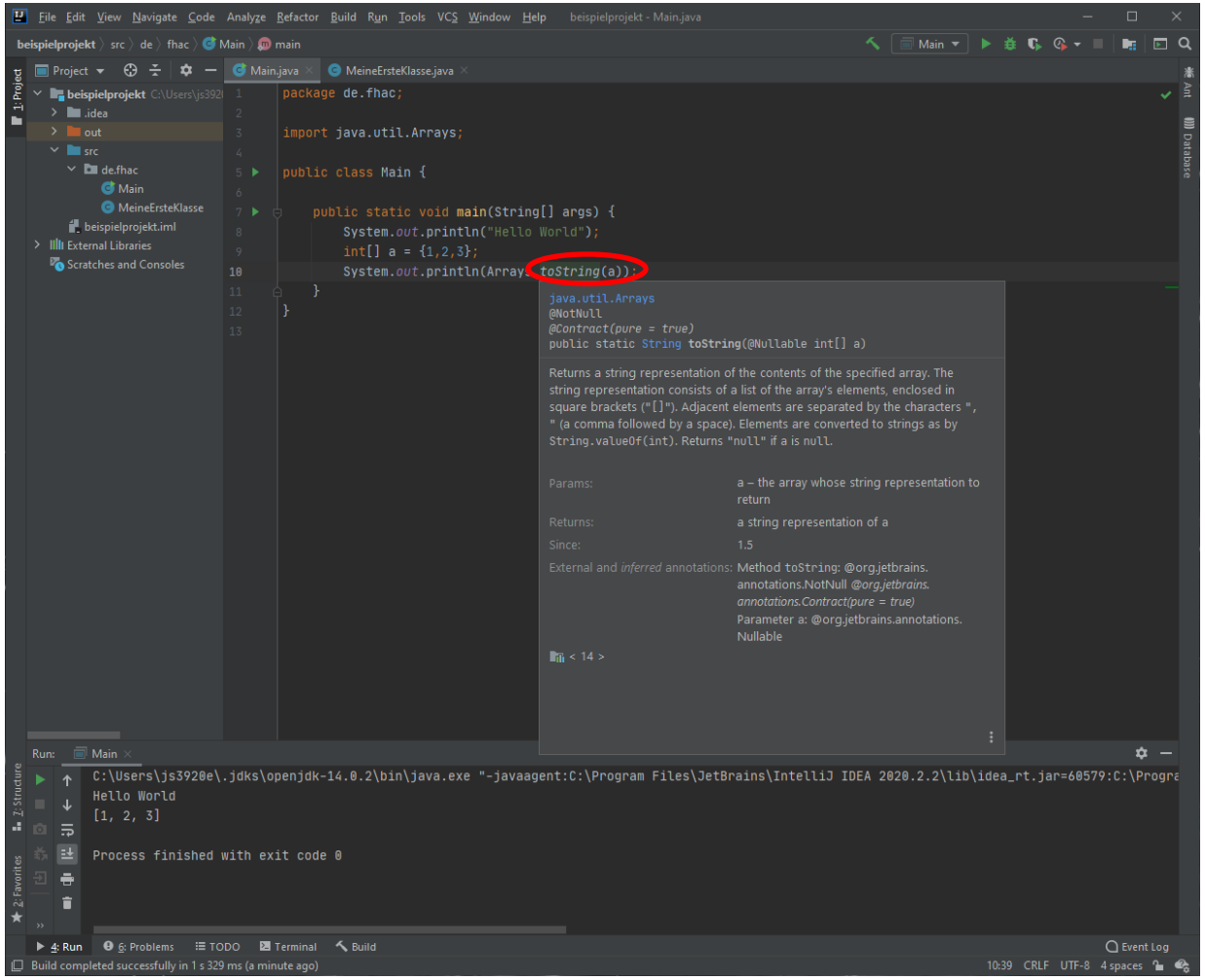


Die Argumente können (durch Leerzeichen) getrennt im Feld *Program arguments* eingegeben werden und mit Klick auf *OK* bestätigt werden

Zuletzt einfach wie gewohnt das Programm starten (Play-Knopf oder Tastenkombination **`Shift + F10`**)

Java API-Dokumentation verwenden
Einführung in IntelliJ IDEA

Die **Java API-Dokumentation** ist bereits in IntelliJ IDEA integriert.



Bei *Mouseover* eines Methodennamen (in diesem Fall **`toString`**) wird die Dokumentation in einem Tooltip angezeigt.

Alternativ kann per **`STRG + Klick`** auf den Methodennamen die entsprechende Codestelle der Implementierung aufgerufen werden. Dort findet sich die Dokumentation in den Kommentaren.

Weitere nützliche Features
Einführung in IntelliJ IDEA

Features um dem Benutzer die **Codenavigation** zu erleichtern:

- Sprung zur Deklaration/Implementierung
  Über **`STRG + Klick`** auf den Methoden- oder Variablen-Namen lässt sich die Deklaration bzw. Implementierung aufrufen

- Anzeige aller Nutzungen der Variablen/Methode im Code (**`Find Usages`**)
  über Rechtsklick auf die Variable/Methode und Auswahl von **`Find Usages`**
  ✉ Zeigt in einem Unterfenster die Liste der Vorkommnisse

Refactoring ermöglicht es z.B. eine Variable/Methode und gleichzeitig alle ihre weiteren Vorkommnisse umzubenennen (Rechtsklick auf Variable/Methode ✉ **`Refactor`** ✉ **`Rename`**)

Das Debuggen besteht grob aus drei Phasen:

1. Breakpoint setzen
2. Debugger starten
3. Debugging-Ansicht

Einen Breakpoint setzen
Einführung in IntelliJ IDEA

Beim Debuggen kann das Programm bis zu einem bestimmten Punkt laufengelassen und dann **pausiert** werden

- Diesen Punkt nennt man **Breakpoint**

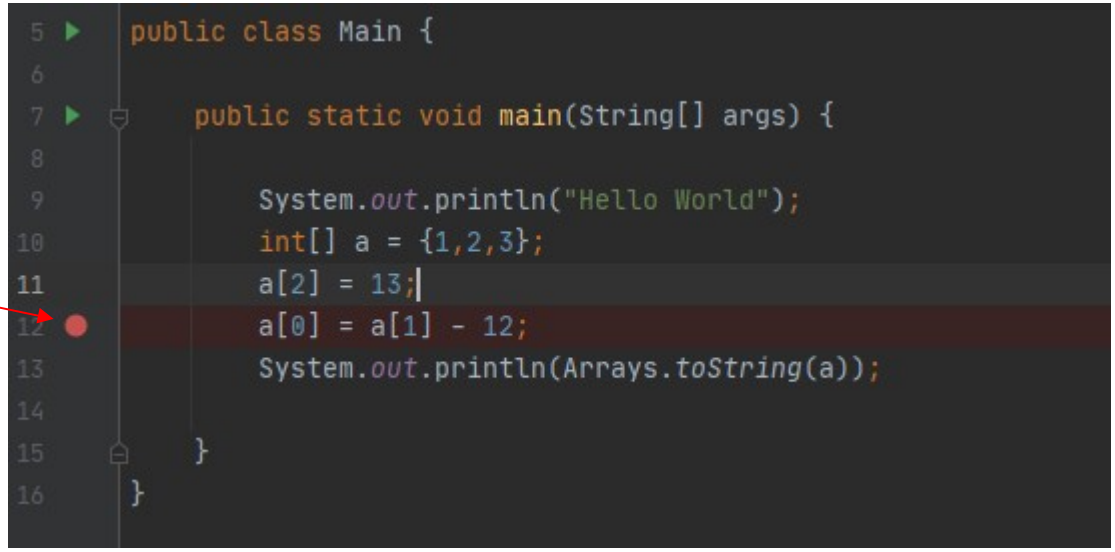
Ein Breakpoint lässt sich in einer beliebigen Programmzeile setzen

Um einen **Breakpoint zu setzen**
- Klick auf den Randbereich vor der gewünschten Zeile

Alternativ Tastenkürzel:
`Strg+F8`

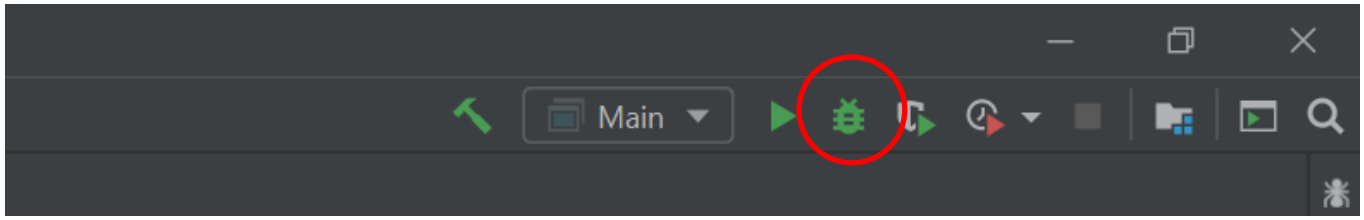
Der rote Punkt markiert den Breakpoint.

Es können auch mehrere Breakpoints gesetzt werden.

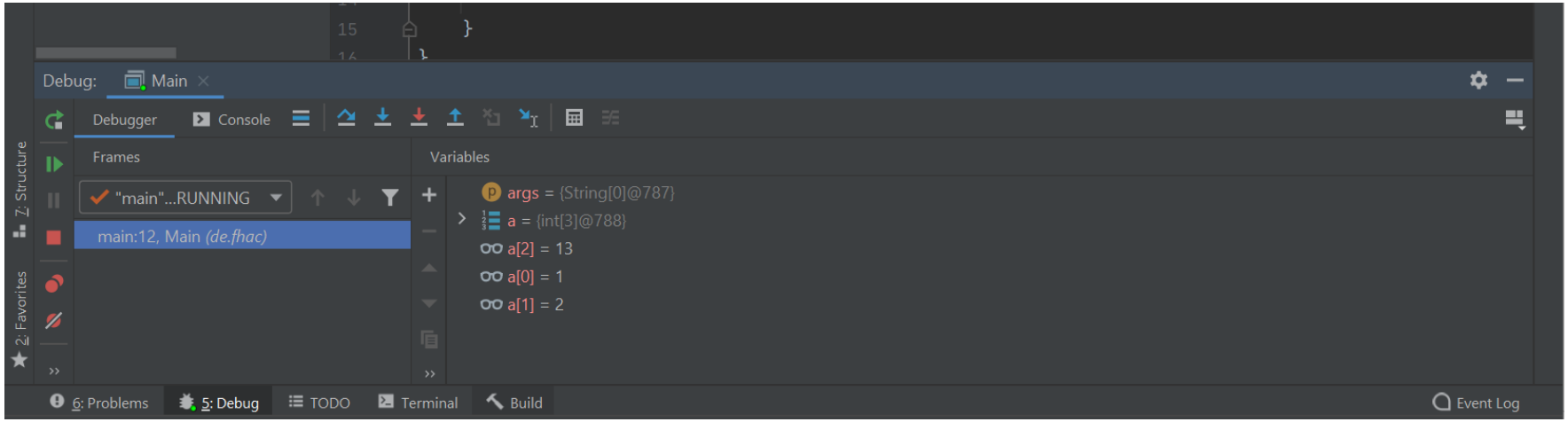


Den Debugger starten
Einführung in IntelliJ IDEA

Zum Starten des Debuggers wird auf das **Käfer** (engl. Bug) **Symbol** geklickt oder die Tastenkombination **`STRG + SHIFT + F9`** verwendet.



Für den Debugger hat IntelliJ ein eigenes Teilfenster welches üblicherweise im unteren Bereich angezeigt wird:



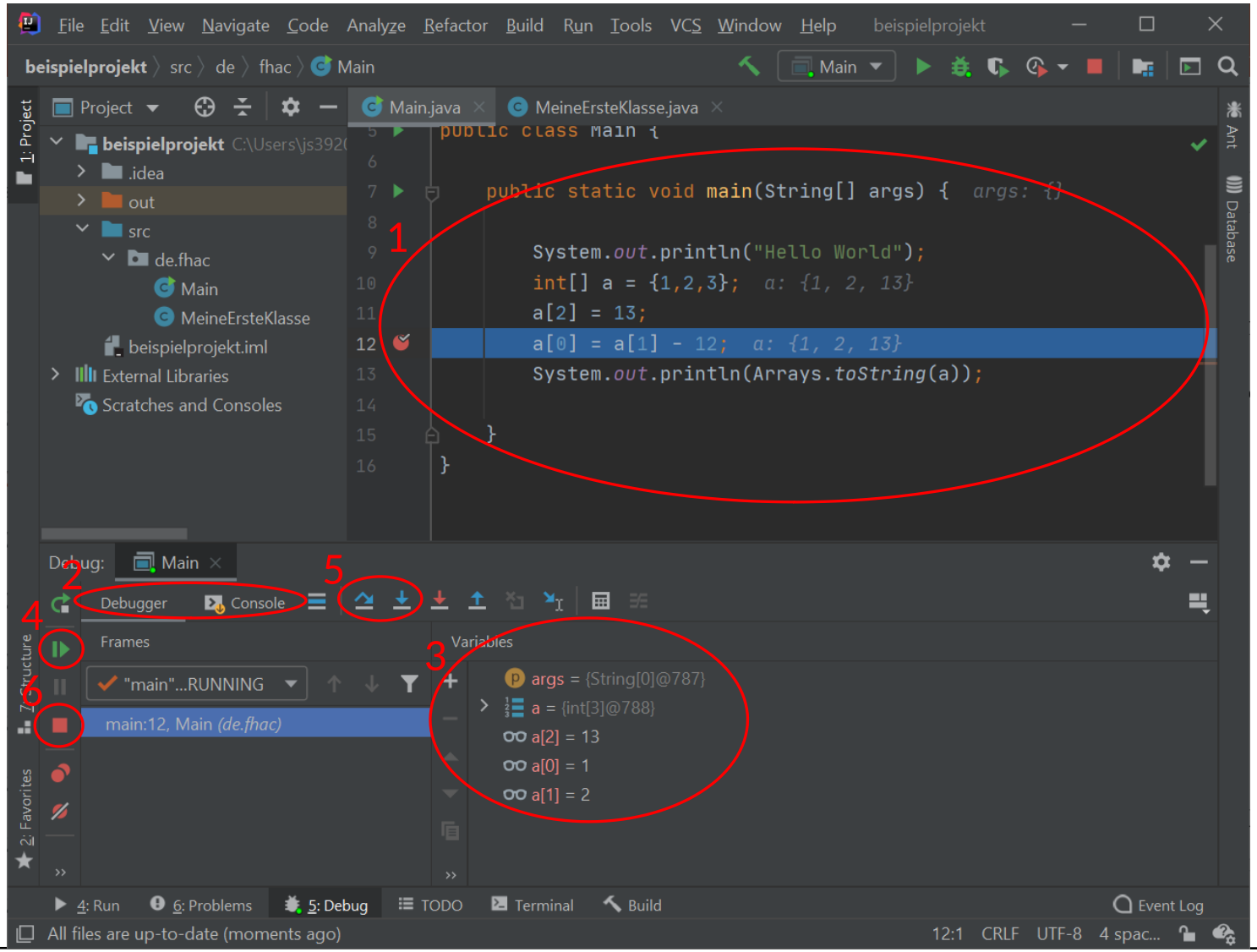
Die Debugging-Ansicht
Einführung in IntelliJ IDEA

Die **Debugging-Ansicht** ist wie folgt strukturiert:

1. **Das Programm**, das bis zum vorher gesetzten Breakpoint gelaufen ist. Die Zeile, bei der das Programm gestoppt ist, wird (hier blau) hervorgehoben
2. Hier kann zwischen Konsolen- und Debugger-Ansicht gewechselt werden
3. Die beobachteten **Variablen** (hier `args` und `a`)
4. Die grüne Resumetaste zum Weiterlaufen lassen (bis zum nächsten Breakpoint – alternativ `F9`)
5. Tasten zum **zeilenweisen Weiterlauf**
   - Ersten Taste: Nächste Anweisung ausführen ohne(!) in ein eventuelles Unterprogramm zu springen (Step Over – `F8`)
   - Zweiten Taste: Nächste Anweisung ausführen und in ein eventuelles Unterprogramm wechseln falls nötig Step Into – `F7`)
6. Stop-Knopf um das Debugging zu Beenden

Auf der nächsten Folie zeigt ein Screenshot die entsprechenden Stellen im IntelliJ Fenster.

Die Debugging-Ansicht
Einführung in IntelliJ IDEA

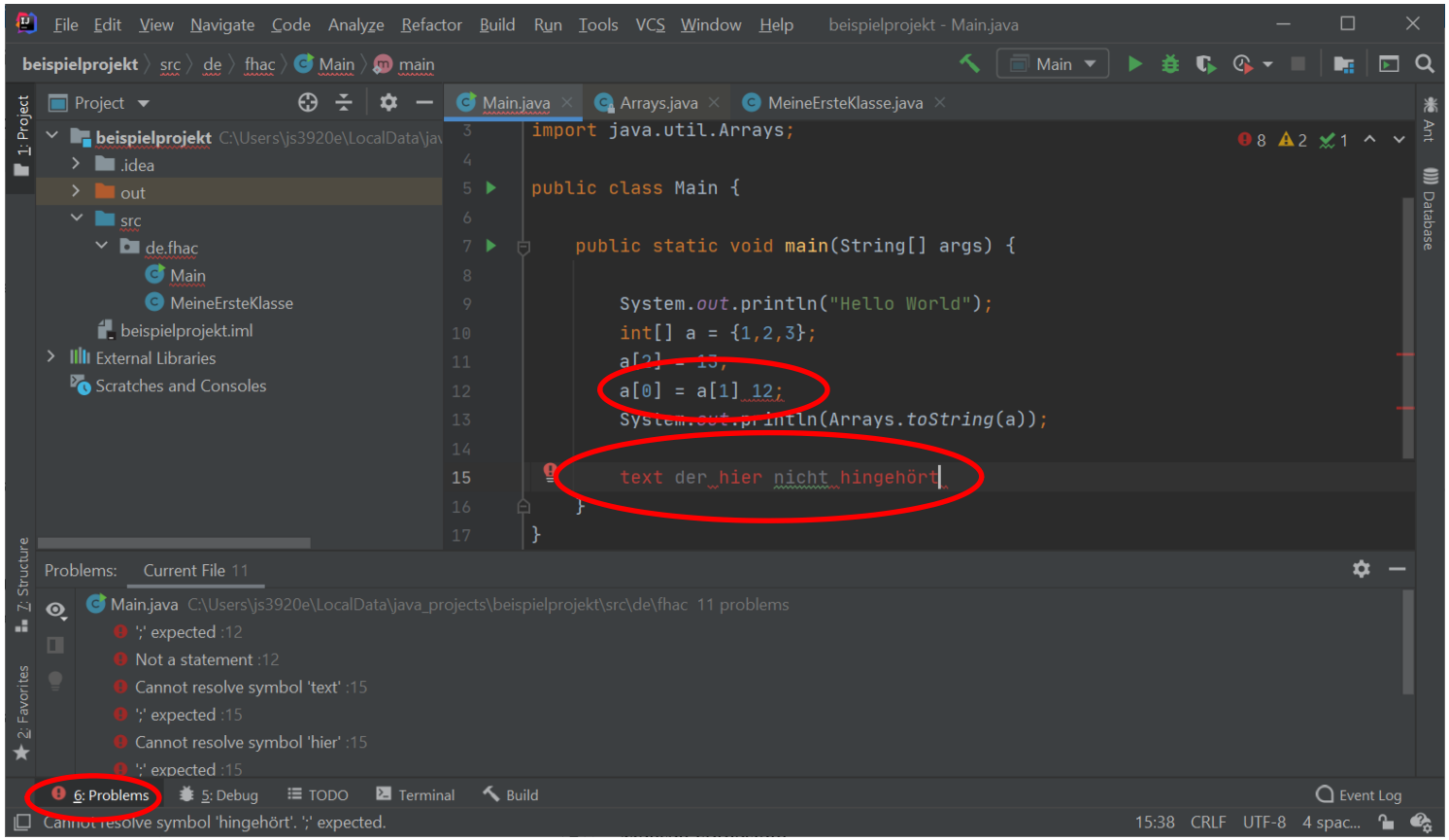


Fehlermeldungen
Einführung in IntelliJ IDEA

IntelliJ markiert bereits während der Eingabe **Syntaxfehler**

→ Im Editorfenster rot geschlängelt unterstrichen

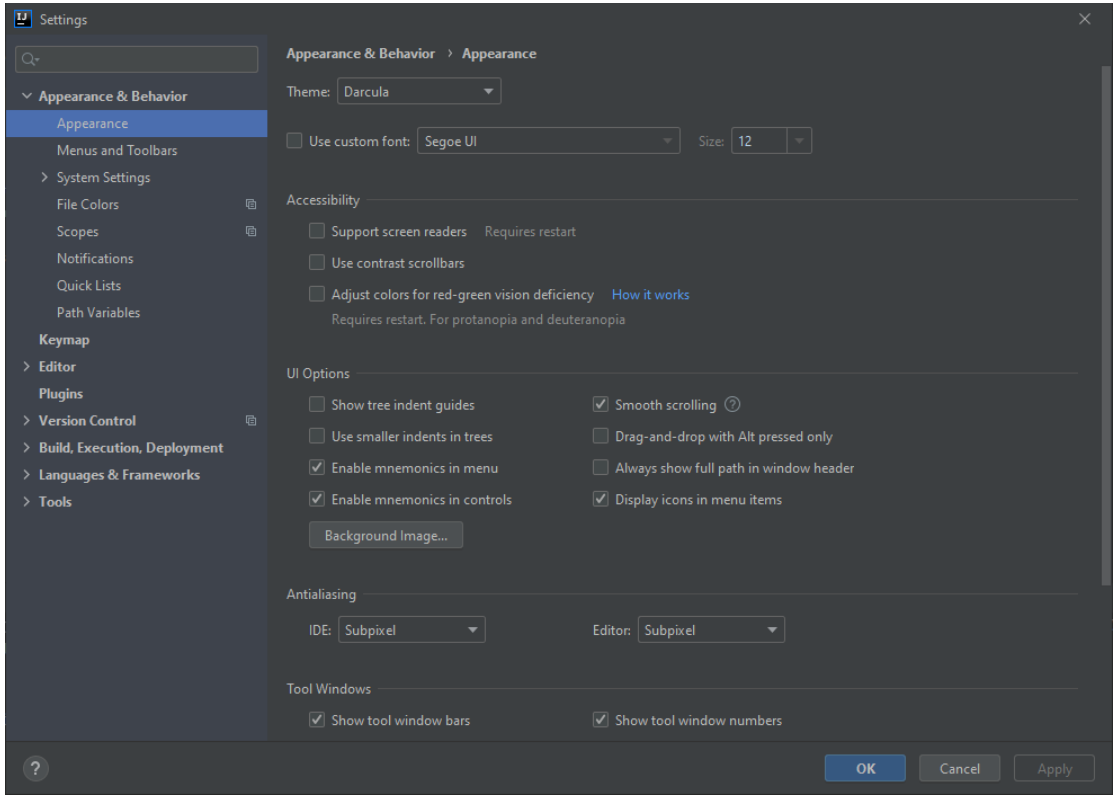
→ Im Reiter **`Problems`** angezeigt (Falls dieser nicht sichtbar ist: Auf den im Bild markierten Reiter klicken)



Einstellungen
Einführung in IntelliJ IDEA

Über Menüpunkt **`File`** ✉ **`Settings`** können allgemeine Einstellungen vorgenommen werden

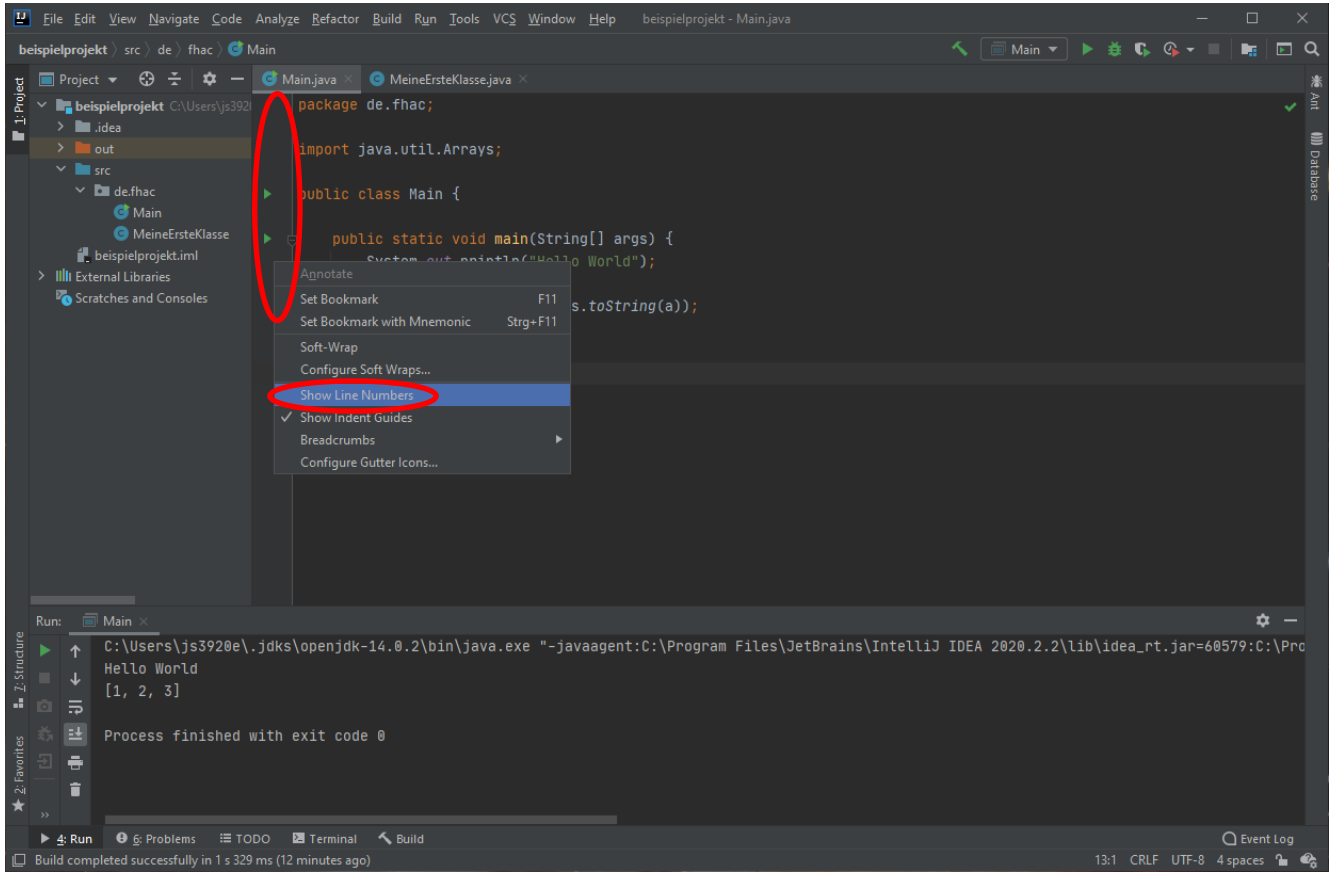
Einstellmöglichkeiten findet man bei Bedarf schnell über das Eingabefeld links oben (mit dem Lupensymbol)



Zeilennummern
Einführung in IntelliJ IDEA

Zeilennummerierung vor den Zeilen werden standardmäßig angezeigt.
Falls dies aus irgendeinem Grund nicht der Fall sein sollte:

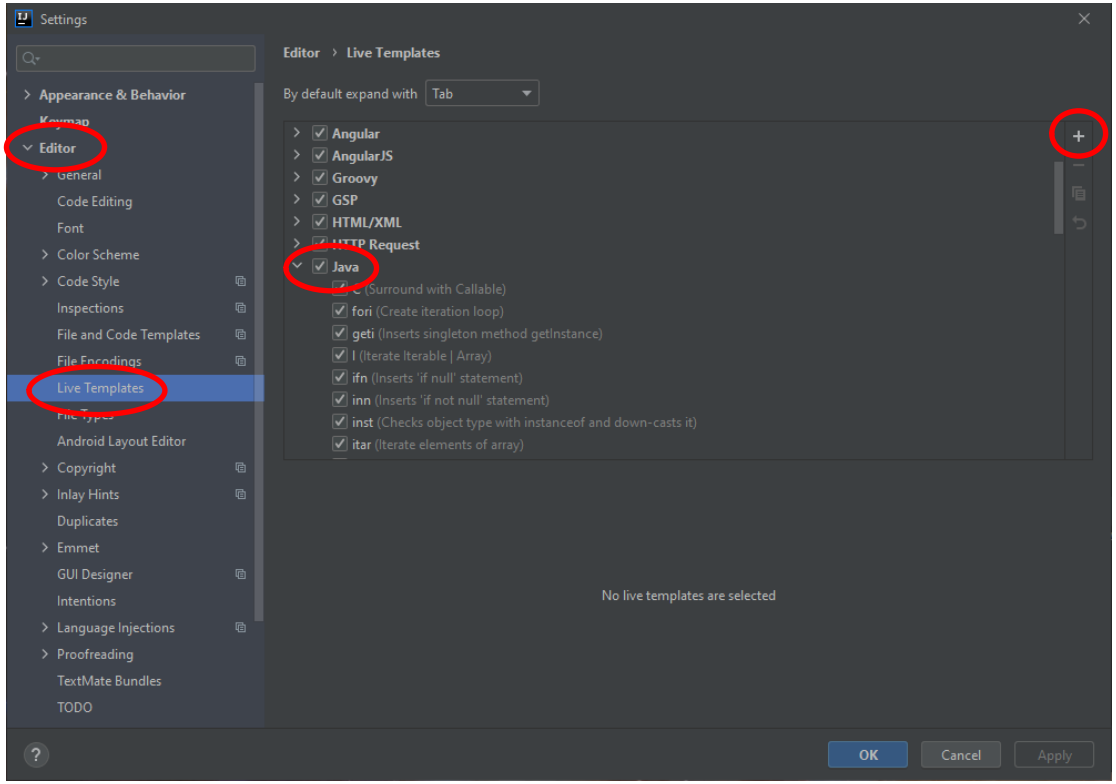
- Rechtsklick auf die kleine Leiste neben dem Quellcode
- Haken bei **`Show Line Numbers`** setzen



Wichtige Templates
Einführung in IntelliJ IDEA

IntelliJ IDEA bietet **Templates** für wichtige Befehle.

Aufruf der Templates über den Kurznamen des Templates und dann drücken von **`Tab`** auf der Tastatur
✉ Der Code wird automatisch vervollständigt



Verfügbare Templates ansehen:

- Menü **`File`** ✉ **`Settings`**
- Dort über die Gruppe **`Editor`** ✉ **`Live Templates`**
- In der Liste **`Java`** expandieren

Tipp: Über das **+** an der Seite können eigene Templates hinzugefügt werden

Wichtige Templates
Einführung in IntelliJ IDEA

Template-Name	Beschreibung	Erzeugter Code
sout	print to standard out	`System.out.println();`
ifn	if statement (mit == null)	`if ( == null) {` `}`
main psvm	main method	`public static void main(String[] args) {` `}`
fori itar	iterate over array	`for (int i = 0; i < ; i++) {` `}`
ritar	iterate over array in reverse	`for (int i = a.length - 1; i >= 0; i--) {` `}`

Schnelle Shortcuts
Einführung in IntelliJ IDEA

Tastenkombination	Auswirkung
Shift + F10	Java Anwendung ausführen
Shift + F9	Java Anwendung debuggen
F9	Debugger weiterlaufen lassen (Resume)
F7	Debugger: Unterprogrammaufruf (Step Into)
F8	Debugger: Unterprogrammaufruf überspringen (Step over)
Strg + F8	Breakpoint ein-/ausschalten
Strg + / Strg + Shift + /	Zeile(n) aus-/einkommentieren Blockkommentar `/* … */`
Strg + D	Zeile oder Auswahl dublizieren
Shift + Alt + Up/Down	Zeile hoch/runter verschieben
Strg + Alt + O	Imports automatisch organisieren
Strg + Alt + L	Code automatisch formatieren
Shift + F6	Refector ✉ Rename

Zusammenfassung: Besondere Features von IntelliJ

Arbeiten mit einer integrierten Entwicklungsumgebung

- Inkrementeller Übersetzer
- Markierung der Fehlerstelle
- Semi-Automatische Fehlerkorrektur
- Kontextabhängige Code-Completion
- Refactoring
- Integrierter Debugger
- Navigation zur Definition und Verwendung
- Erweiterbarkeit
- Code-Templates
- Team-Unterstützung
- Grafisches, sprachsensitives Diff
- …

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Prof. Dr. Bodo Kraft

FH Aachen
University of Applied Sciences

Fachbereich Medizintechnik und Technomathematik
Heinrich-Mußmann-Str. 1
52428 Jülich

Telefon +49. 241. 6009 53757
Telefax +49. 241. 6009 53118

E-Mail: kraft@fh-aachen.de