

0 IDEs

Zur Entwicklung von komplexeren Anwendungen für den praktischen Einsatz empfiehlt es sich, eine Entwicklungsumgebung (Integrated Development Environment, IDE) mit Unterstützung für Syntaxhervorhebung (Code-Highlighting) und -vervollständigung (Code-Completion) zu verwenden. Die am häufigsten genutzten Anwendungen für Java sind:

- IntelliJ (<https://www.jetbrains.com/idea>)
- Eclipse (<https://www.eclipse.org>)
- NetBeans (<http://www.netbeans.org>)
- BlueJ (<https://www.bluej.org>)

Im Rahmen des Kurses Programmierung und Programmiersprachen wird die **IntelliJ IDE** zur Entwicklung genutzt (bereits auf den Rechnern der CIP-Pools vorinstalliert). Damit diese ein effektives Werkzeug zur Programmierung darstellt, muss der Nutzer sich einiger Funktionen und Vorteile bewusst sein.



Tipp

Jetbrains stellt auch eigene Videotutorials zur Benutzung von IntelliJ zur Verfügung: [“Getting To Know Your IDE”](#) auf YouTube.

0.1 Einführung IntelliJ

Beim ersten Start von IntelliJ wird man durch ein Setup geführt, in dem man Themes und Plugins wählen kann. Diese Schritte können übersprungen werden, es werden keine Plugins benötigt.

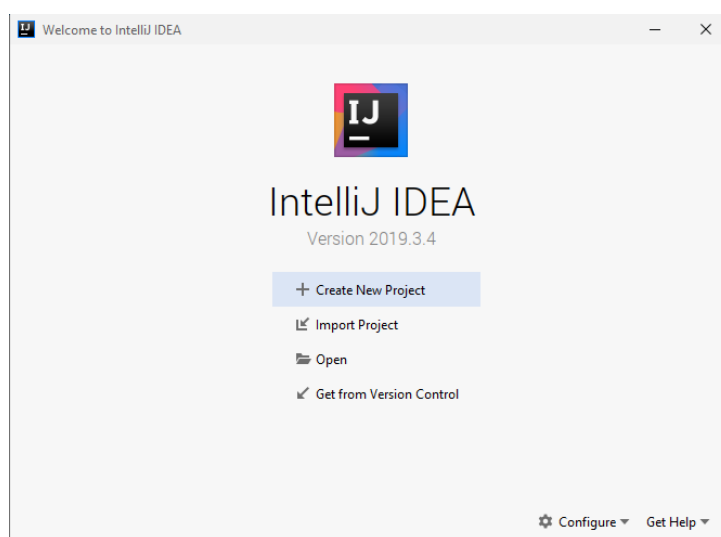


Abbildung 1: IntelliJ Willkommensbildschirm

Als nächstes wird das Willkommensfenster angezeigt (Abb. 1). Wir wählen “Create New Project” aus. Im folgenden Fenster muss eine Projekt-SDK gewählt werden. Wurde Java korrekt installiert sollte diese bereits ausgewählt werden (zur Erinnerung: Mindestversion für diese Vorlesung ist Java 8). Ist noch keine SDK vorhanden kann mit “New” eine SDK ausgewählt werden. Dazu sollte der Ordner in dem die gewünschte Javaversion liegt ausgewählt werden (siehe Abb. 2).

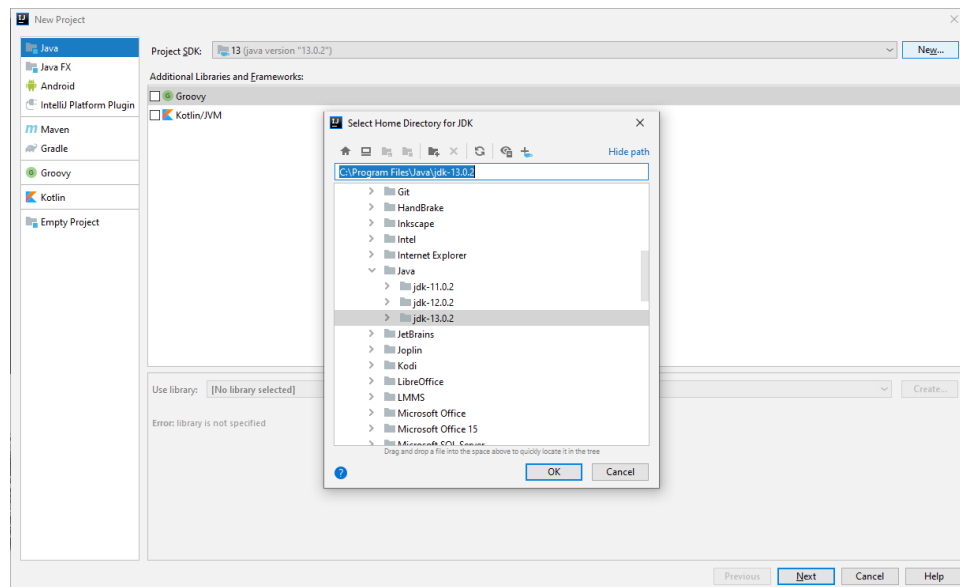



Abbildung 2: Java SDK auswählen

Mit einem Klick auf “Next” können wir noch ein Template auswählen. Ein sinnvolles Template für die meisten Übungsaufgaben ist “Command Line App”, welches ein Grundgerüst für eine Konsolenapplikation bereitstellt. Als nächstes muss ein Projektname und Ort ausgewählt werden. Sollten Sie in einem CIP-Pool arbeiten, legen Sie ihre Projekte am besten auf ihrem Studierendenlaufwerk (meistens Z:) ab. Das Basispaket können Sie entweder leer lassen oder `de.rub` nutzen.

0.2 Der Workspace

Der IntelliJ-Workspace ist standardmäßig aufgeteilt in Editor, Projektoutline, und Ausgabekonsole (siehe Abb. 3). Der Editor ist der Bereich, in dem der Code geschrieben wird. Anfangs ist hier die `main`-Methode zu sehen, der Einstiegspunkt des Programms. In Kapitel 0.4 werden einige Tipps für den Umgang mit dem Editor vorgestellt. Die Projektoutline ist wie ein Dateieexplorer für das Projekt. Hier werden alle Java-Klassen und weitere Dateien aufgelistet, die Teil des Projektes sind. Die Ausgabekonsole wird genutzt um das Programm zu testen. Alle `print`-Befehle geben hier ihren Text aus. Alternativ wird hier auch der Debugger gesteuert, mit welchem man Bugs im Code finden kann.

In der oberen rechten Toolbar befinden sich die Buttons zum Bauen und Ausführen des Programms. Drückt man den “Run”-Button (oder drückt  + `F10`), wird das Programm in der Ausgabekonsole ausgeführt.

Alle Views in IntelliJ können verschoben und angepasst werden. Zur Not kann das

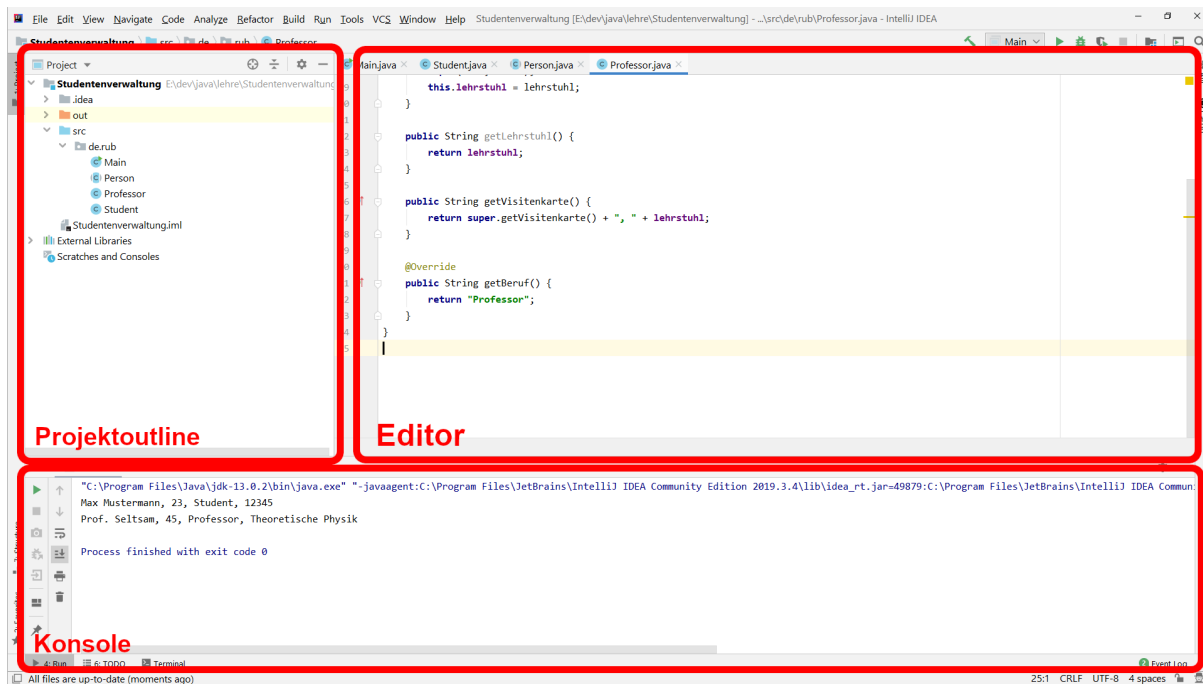



Abbildung 3: Der IntelliJ Workspace





Default-Layout mit  + **F12** wiederhergestellt werden.

0.3 Neue Klassen erstellen

Um eine neue Klasse zu erstellen, wählt man mit Rechtsklick ein Paket im Projektoutline an (Abb. 4). Mit **New** ▸ **Java Class** öffnet sich ein Fenster, in dem man den Klassennamen eingeben kann und zwischen Klasse, Interface, und Enum wählen kann. Die neue Klasse wird im Editor geöffnet. Alternativ kann man auch eine neue Klasse per **File** ▸ **New** ▸ **Java Class** erstellen.

0.4 Tipps für den IntelliJ-Editor

Codevervollständigung

IntelliJ bietet eine umfangreiche Codevervollständigung für Java an. Im Editor kann man zu jeder Zeit **Strg** +  drücken, um eine Liste von möglichen Vervollständigungen zu erhalten (siehe Abb. 5). Mit  und  wird eine Vervollständigungen ausgewählt und mit  oder **Enter** bestätigt. Die Vorschläge sind dabei Kontextbasiert.

Besonders nützlich ist die Codevervollständigung zum Ausschreiben von langen Klassennamen. Klassen die mit Codevervollständigung referenziert werden, werden auch automatisch als **import**-Statement der aktuellen Klasse hinzugefügt.

Auch nützlich ist die Liste von Methoden und Attributen die die Codevervollständigung angeben kann. Schreibt man eine Objektreferenz und hängt diese mit einem Punkt an, z.B. "**str.**", und ruft dann die Codevervollständigung auf, so schlägt IntelliJ alle Metho-

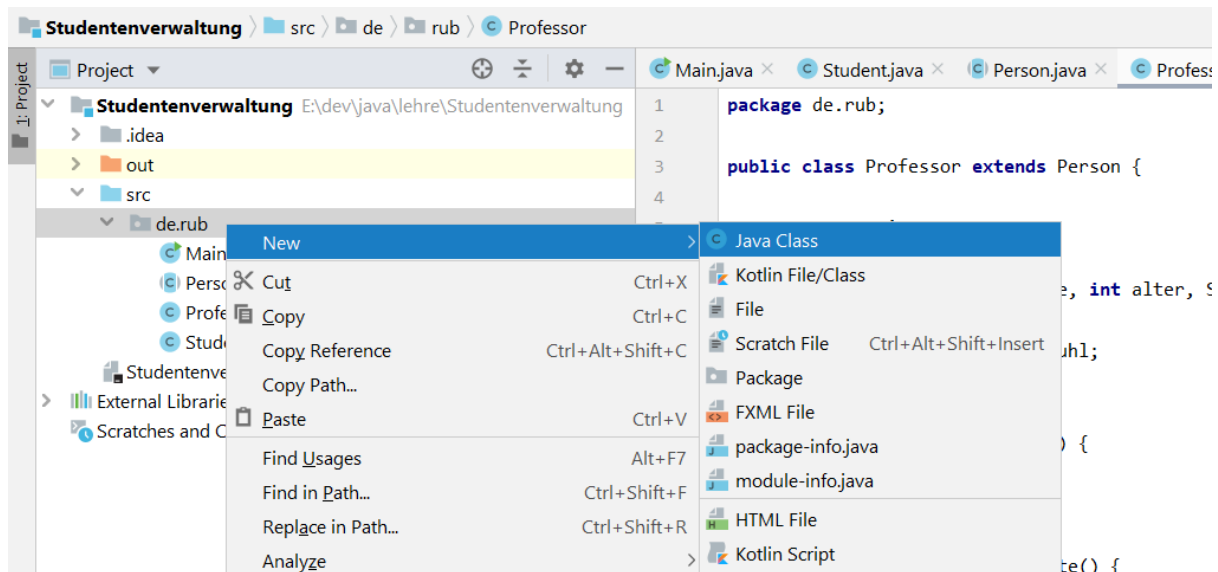


Abbildung 4: Neue Klasse erstellen

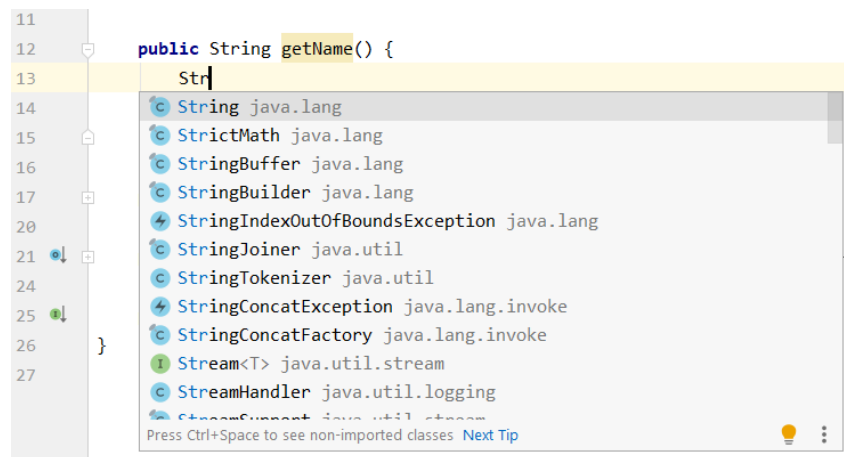


Abbildung 5: Codevervollständigung in IntelliJ



den und Attribute für dieses Objekt vor welche verfügbar sind.

Weiterhin bietet die Codevervollständigung Templates für häufig verwendete Strukturen. "fori" erstellt eine for-Schleife mit Index, "souf" ist eine Shorthand für Konsolenausgaben per `System.out.println()`, und "main" erstellt die Main-Methode für eine Klasse. Alle Templates können mit `[Strg] + [J]` eingesehen werden.

Codekorrektur




IntelliJ prüft konstant den Code nach Fehlern die das Kompilieren des Codes verhindern. Hat IntelliJ einen Fehler erkannt, wird die entsprechende Stelle rot unterstrichen. Bewegt man die Maus über die fehlerhafte Stelle wird eine Erklärung des Fehlers angezeigt. Mit `[Alt] + [Enter]` kann man unter dem Cursor mögliche Fehlerbehebungen anzeigen und anwenden lassen.

Generieren von Code

Javaklassen haben häufig Methoden welche automatisch generiert werden können. Dies schließt Konstruktoren sowie Getter und Setter-Methoden ein. Drückt man  +  so erhält man eine Liste von Elementen die man automatisch generieren lassen kann. Meist folgt der Auswahl noch ein Fenster in dem man eventuelle Attribute auswählen kann welche in Betracht gezogen werden sollen. Die Codevervollständigung und Codegenerierung nimmt dem Programmierer einiges an Schreibarbeit ab, jedoch sollte man als Student nicht vergessen, dass Konstruktoren und Getter- und Setter-Methoden **in der Klausur** immer noch **per Hand** geschrieben werden müssen!




Codenavigation

In größeren Projekten kann es nützlich sein, sich schnell durch den Code bewegen zu können. IntelliJ bietet dazu mehrere Möglichkeiten:

- Hält man  gedrückt und klickt auf ein Objekt oder eine Klasse, so wird man zur Definition navigiert.
- Mit  +  kann man direkt per Suchfeld zu einer Klasse springen.





Tipp

Hat man den Tastenkürzel für einen bestimmten Befehl vergessen, so kann man mit  +  +  alle Aktionen durchsuchen.

Debugging

Beim Programmieren ist es unvermeidlich, dass sich Fehler in den Code einschleichen. Debugging-Tools können helfen, Fehler zu identifizieren.

Über  +  wird das Programm im Debug-Modus gestartet. Es gibt drei Arten wie der Debugger während der Laufzeit aktiviert werden kann:

1. Das Programm hält an einem gesetzten Breakpoint an (Siehe Abb. 6).
2. Führt ein Fehler (speziell, eine Exception) zu einem Absturz des Programms so wird automatisch der Debugger an der verantwortlichen Stelle aufgerufen.
3. Mit dem Pause-Button kann das Programm zu jeder Zeit manuell angehalten und debuggt werden (nützlich wenn man z.B. in einer Endlosschleife steckt).

Ist der Debugger aufgerufen worden, wird die Ausführung des Programms angehalten bis es manuell fortgeführt wird. Das Debuggerfenster zeigt auf der linken Seite den aktuellen Frame auf dem Ausführungsstack an. Durch den Ausführungsstack kann man nachvollziehen, an welchem Punkt in der Ausführung man sich genau befindet. Die aufrufenden Funktionen werden hierarchisch von der main-Methode bis zur aktuellen Codezeile aufgelistet. Auf der rechten Seite werden alle Variablen aufgelistet die im aktuellen Scope

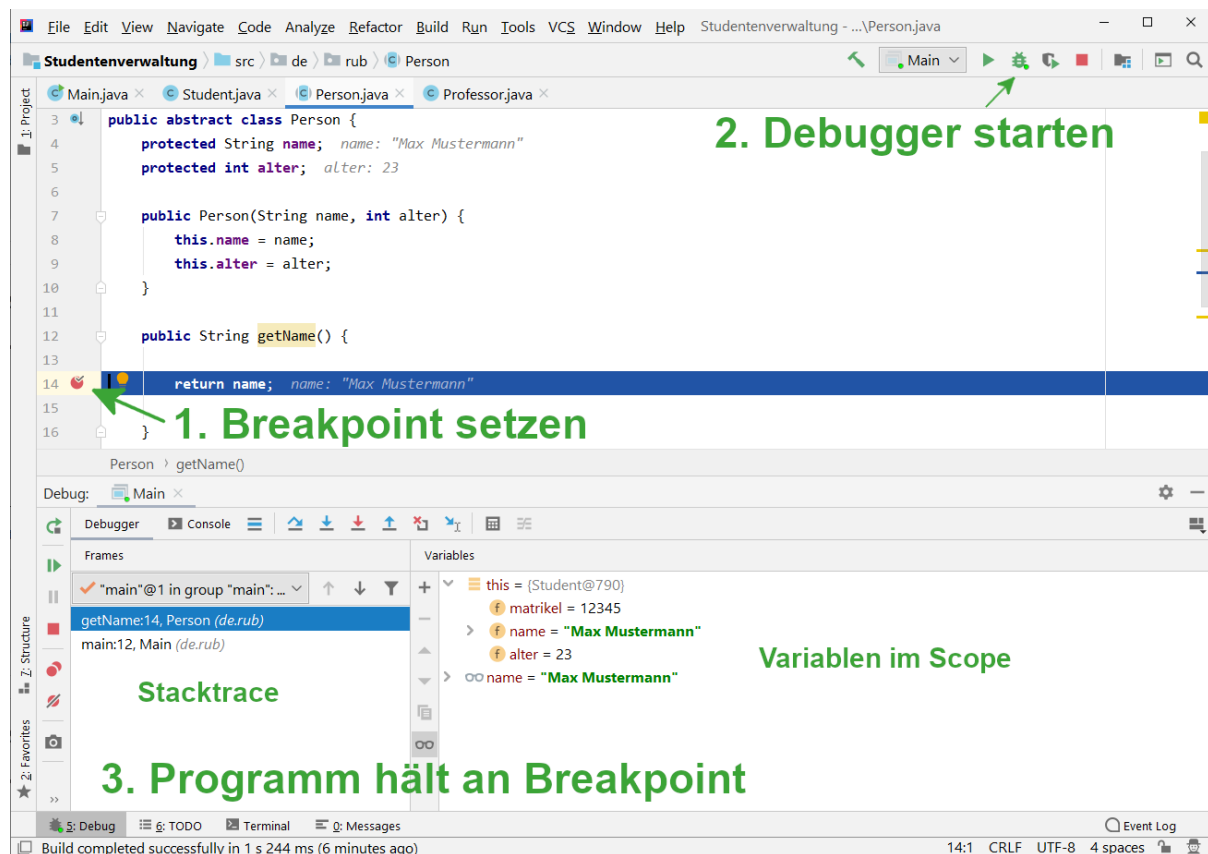


Abbildung 6: Debugging in IntelliJ

verfügbar sind. Alternativ kann man auch die Variablenwerte direkt im Code einsehen. In der Toolbar oben im Debuggingfenster sind die Buttons “Step Over”, “Step Into”, und “Step Out” zu finden. Mit diesen Buttons kann man sich Schritt für Schritt durch den Code bewegen. “Step Over” führt die aktuelle Codezeile aus und springt zur nächsten weiter innerhalb des aktuellen Frames. “Step Into” erlaubt es, in die Funktion in der aktuellen Codezeile hereinzuspringen und steigt damit einen Frame tiefer. “Step Out” führt den Rest des aktuellen Frames aus und springt dann einen Frame nach oben. Um den Debugger zu beenden und die Ausführung fortzuführen, drückt man den Play-Button auf der linken Seite des Debuggers.



Tipp

Der Debuggingprozess in IntelliJ wird im Detail hier beschrieben:
<https://www.jetbrains.com/help/idea/debugging-code.html>

0.5 Exportieren von Projekten

Projekte können in ein portables Format exportiert werden, um sie an andere Personen weiterzugeben. Dazu wählen wir **File** > **Export** > **Project to Zip File**. Im nächsten Fenster kann dann der Speicherort und der Dateiname angegeben werden.