Programmierwerkzeuge

Unterrichtseinheit 4: Integrierte Entwicklungsumgebung

Lehr- und Lernmaterialien

Zielgruppe	Lernende, die schon eine Programmiersprache gelernt und erste Erfahrungen in der Programmierung gesammelt haben.
Lernziele	Die Lernenden haben einen Überblick über integrierten Entwicklungsumgebungen für Java. Sie haben eine konkrete IDE auf dem eigenen Rechner installiert und können damit ein Hello-World-Programm erstellen. Sie können das Prinzip „Project“ und „Workspace“ erläutern. Sie kennen eine typische Projektverzeichnisstruktur. Sie haben einen Überblick über die wichtigsten Projekt- und Workspace-Eigenschaften und können diese in der IDE ändern.
Stichwörter	Programmierwerkzeuge, IDE, Integrierte Entwicklungsumgebung, integrated development environment, workspace, project

Fragen und Verbesserungsvorschläge bitte an Prof. Dr. Reinhard Brocks, Hochschule für Technik und Wirtschaft des Saarlandes.
E-Mail: reinhard.brocks@htwsaar.de

Einen Überblick über die Unterrichtseinheiten zu „Programmierwerkzeuge“ findet man hier.

Versionsgeschichte

Datum	Änderung
26. Oktober 2019	Erste Version

Für die Nutzung und die Bearbeitung steht das Dokument hier auch im LaTeX-Format zur Verfügung.

Programmierwerkzeuge - Integrierte Entwicklungsumgebung

Aufgabe: Integrierte Entwicklungsumgebung, 45 Minuten

Entscheiden Sie sich für eine Java-IDE, entweder Eclipse, IntelliJ IDEA oder NetBeans IDE und installieren Sie diese auf Ihrem Rechner.

Aufgabe: Workspace / Project, 120 Minuten

1. Legen Sie ein Hello-World Java-Programm mit der IDE Ihrer Wahl an. Listen Sie die wichtigsten Eigenschaften eines Projekts und eines Workspaces auf, die Sie beim Erstellen eines Projekts übernehmen oder angeben müssen. Was sind allgemeine Eigenschaften und welche sind Java-spezifisch? Wo werden die Eigenschaften gespeichert?

2. Führen Sie das Hello-World-Programm aus. Schauen Sie sich die generierten Dateien an. Nach welchen Prinzipien sind die Verzeichnisse generiert worden?

3. Wie groß ist das Projektverzeichnis? Was können Sie alles löschen, so dass das Projekt dennoch wieder in der IDE geöffnet und ausgeführt werden kann? Wie groß ist dann das Verzeichnis?

4. Definieren Sie die Begriffe „Project“ und „Workspace“.

5. Fassen Sie in Gruppenarbeit die Ergebnisse auf einer Flipchart zusammen.

Aufgabe: Prüfungsleistung

Legen Sie für Ihr Projekt der Prüfungsleistung eine Workspace-/Projekt- und Package-Struktur fest. Passen Sie die Eigenschaften des IDE-Projekts so an, dass es in der IDE kompiliert und ausgeführt werden kann.

Arbeitsergebnisse

Definition Integrierte Entwicklungsumgebung, IDE: *Eine Integrated Development Environment ist eine Sammlung von Tools (CASE tools - Computer-Aided Software Engineering tools) innerhalb eines Programms für die Softwareentwicklung. Die Basiswerkzeuge sind Source-Code-Editor, Compiler, Debugger und Build-Werkzeuge (ant, maven, gradle). Weitere Tools sind z. B. Versionsverwaltung, Aufgabenlisten / issue tracker, Javadoc-Generator.*

Definition Projekt: *Innerhalb einer IDE gruppiert ein Projekt alle projektrelevanten Dateien. Diese werden strukturiert in einem gemeinsamen Projektverzeichnis gespeichert (siehe Abbildung 1). Die Projekteigenschaften sind normalerweise in einer textbasierten und IDE-spezifischen Projektdatei.*
*Das Ergebnis eines Build-Vorgangs auf Projektebene, das **Artefakt**, ist üblicherweise ein ausführbares Programm oder eine Bibliothek.*

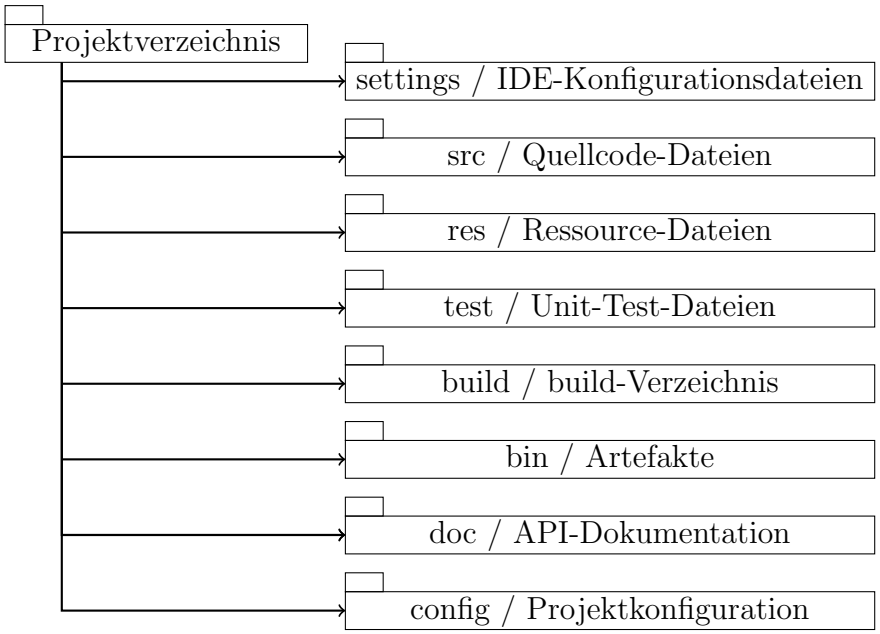

Abbildung 1: Schematische Struktur eines Projektverzeichnisses

Abbildung 1 stellt schematisch ein minimales Projektverzeichnis dar. Die Projektdateien, inklusive die generierten, werden auf Verzeichnisse aufgeteilt. Die IDE schlägt eine erste Verzeichnisstruktur vor. Üblicherweise gibt es Verzeichnisse für:

- (settings) IDE-Projektkonfigurationsdateien
- (src) Quellcode-Dateien
- (res) **Ressource**-Dateien wie Wörterbücher verschiedener Sprachen, Icons, Bilder, Audio, Video.

- (test) Testprogramme mit Testdaten
- (build, out) temporäre Build-Dateien (*.class, *.o, *.obj)
- (bin, target) Artefakte, Dateien wie *.jar, *.exe, *.dll, die deployed werden.
- (doc) Verzeichnis für die generierte API-Dokumentation
- (config) Verzeichnis für die Konfigurationsdateien des Projekts

Wichtig ist, dass man die selbst erstellten und die generierten Dateien klar voneinander trennt. Die aus anderen Projekten genutzten Dateien, wie z. B. benötigte Bibliotheken oder Frameworks sollten, falls möglich außerhalb des Projektverzeichnisses liegen.
Je nach Programmiersprache haben die Verzeichnisse auch andere Namen und sind anders strukturiert. In der Regel kommen dann noch versteckte Verzeichnisse für die Versionsverwaltung hinzu.

Projekteigenschaften sind z. B.:

- Projektname
- Verzeichnisse für generierte und temporäre Dateien
- Projekttyp / Zielartefakt (Bibliothek, Programm)
- Compiler-Version
- Verzeichnis für die Quellcode-Dokumentationsgenerierung
- Kommandozeilenargumente
- Liste von Source-Dateien oder nur die Liste der Source-Verzeichnisse

Hinzu kommen bei Java-Projekten folgende Eigenschaften:

- Version des Java Runtime Environments (JRE)
- Version des Java Development Kits (JDK)
- classpath: Pfade zu abhängigen Bibliotoheken (jar-Dateien)

Die üblichen Artefakte sind für Java-Projekte jar oder runnable jar Dateien.

Definition Workspace: *Ein Workspace gruppiert innerhalb einer IDE zusammengehörige Projekte.*
Das können z. B. ein Server-Programm und ein zugehöriges Client-Programm sein. Ein Projekt kann auch eine Bibliothek erstellen, die von einem anderen

Projekt genutzt wird. Diese Abhängigkeiten definieren eine Build-Reihenfolge, die in der IDE festgelegt werden kann oder muss.
Die Projektliste und Workspace-Eigenschaften werden in einer häufig textbasierten Workspace-Datei gespeichert. Einen Workspace zu kompilieren heißt, alle Projekte in einer vordefinierten Reihenfolge zu kompilieren (siehe Abbildung 2).

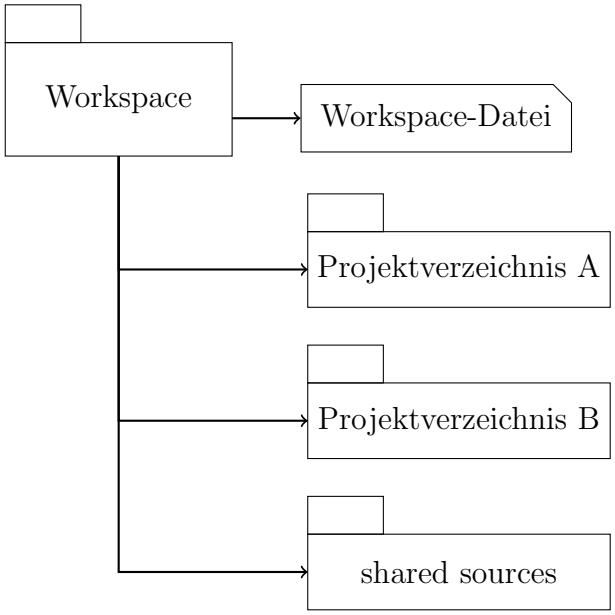

Abbildung 2: Schematische Struktur eines Workspace-Verzeichnisses

Workspace-Eigenschaften, nicht nur bei Java-Projekten, sind z. B.:

- Workspace-Name
- Projektliste
- IDE Version
- Projektabhängigkeiten
- Einstellungen zum automatischen Speichern oder Kompilieren

Die Projektverzeichnisse sollten auch physikalisch in einem workspace-Verzeichnis sein. Es gibt aber auch das Konzept eines virtuellen Workspaces, das nur in der IDE die Projekte gruppiert. Hinzu kommen vielleicht noch Einstellungen von Plug-ins der IDE, die in Konfigurationsdateien abgelegt sind. Ferner werden die Einstellungen der Benutzerschnittstelle, wie Ansichten, Fensterpositionen, etc. gespeichert.

Hinweise für den Dozenten

Der Begriff „Projekt“ ist wichtig. Es kommt immer wieder vor, dass Lernende in einer Entwicklungsumgebung eine Quellcodedatei erstellen, diese aber nicht kompilieren und ausführen können. Das liegt dann meistens daran, dass die Datei keinem Projekt zugeordnet ist oder falls doch, diese nicht als Quellcodedatei kenntlich gemacht wurde. Es kann auch sein, dass ein falscher Projekttyp bei der Erstellung ausgewählt wurde.
Ebenso kommt es vor, dass die Lernenden in einem Projekt mehrere Programme verwalten. Die Strukturierung wird dann durch Pakete erreicht. Dennoch ist dies nicht optimal. Versuchen Sie zu erläutern, dass es strukturierter ist, für jedes ausführbare Programm ein Projekt anzulegen. Gemeinsame Sourcen können dann in einem gemeinsamen Source-Verzeichnis liegen oder in einem Projekt verwaltet werden, das eine Bibliothek als Artefakt generiert.
Die Lernenden sollen auch wissen, dass nicht alle Dateien eines Projektverzeichnisses wichtig sind und dass nur wenige gesichert werden müssen. Dies reduziert die Datenmenge bei der Datensicherung und beim Austausch mit anderen Lernenden oder dem Dozenten. Dieses Verständnis ist notwendig für die korrekte Nutzung einer Versionsverwaltung.