IDE, Debugger, SDK

Eine **integrierte Entwicklungsumgebung (IDE)** ist eine Software, die mehrere Entwicklungswerkzeuge in einer grafischen Schnittstelle vereint. Die IDE bietet alle Funktionen, die für das Schreiben, Ausführen, Dokumentieren und Testen von Software notwendig sind.

Bestandteil einer IDE ist meistens auch ein Tool namens **Debugger**. Mit einem Debugger kann man ein Programm schrittweise ausführen und logische Fehler im Programmcode aufspüren.

Ein **Software Development Kit (SDK)** ist ein Bündel aus Entwicklungswerkzeugen, das für die Übersetzung, Ausführung und das Testing verwendet wird. Wir nutzen in diesem Modul das **Java Development Kit (JDK)**, um Java-Programme zu entwickeln.

Datentypen

Im Speicher befinden sich nur "Einsen und Nullen". Diese Bitmuster müssen aber interpretiert werden, also mit einer Bedeutung versehen werden. Hier kommen die Datentypen ins Spiel.

Ein Datentyp legt folgendes fest:

- Die **Bedeutung** eines Bitmusters, das sich im Speicher befindet
- Einen **Wertebereich**
    - Beispiel: Datentyp Zahl erlaubt die Zahlen im Bereich -128 bis +127.
    - Beispiel: Datentyp Wahrheitswert erlaubt nur die Werte "wahr" und "falsch".
    - Beispiel: Datentyp Zeichenkette erlaubt unendlich viele Werte, also beliebige Verkettungen von einzelnen Zeichen.
- Die **Operationen**, die auf einen Wert angewandt werden können.
    - Beispiel: Datentyp Zahl erlaubt Addition, Subtraktion, Multiplikation.
    - Beispiel: Datentyp Zeichenkette erlaubt Zusammenfügen, Ersetzen, Suchen etc.
    - Beispiel: Datentyp Wahrheitswert erlaubt nur Vergleiche mit "wahr" und "falsch".
- Verhindert unsinnige bzw. fehlerhafte Anweisungen wie "Subtrahieren zweier Zeichenketten" oder "Aufrufen einer Zeichenkette".
- Fördert das **Verständnis** und die **Lesbarkeit** von Quelltext.
- Definiert den **Speicherbedarf** von Werten bzw. Variablen.
    - Beispiel: In Java hat eine ganze Zahl einen Speicherbedarf von 32 Bits und eine Gleitkommazahl einen Speicherbedarf von 64 Bits.

Es gibt Programmiersprachen, die statisches Type-Checking verwenden und Programmiersprachen, die dynamisches Type-Checking benutzen.

Beim **statischen Type-Checking** überprüft der Compiler *vor der Ausführung* des Programms, ob dieses typkonforme Anweisungen enthält.

Beim **dynamischen Type-Checking** wird *während der Ausführung* des Programms geprüft, ob die Anweisungen typkonform sind.

Java verwendet sowohl statisches Type-Checking durch den Compiler und dynamisches Type-Checking durch die Java Virtual Machine.

Verletzt ein Programm die Typregeln, führt das beim statischen Type-Checking zu einem Übersetzungsabbruch und beim dynamischen Type-Checking zu einem Programmabbruch.