Variablen

Spickzettel aus Jannicks C# Intensivkurs

Was sind Variablen?

Mit Variablen kannst du Werte während er Laufzeit deiner Anwendung speichern. Schließt du die Anwendung verfallen die Werte (sie sind also nicht permanent gespeichert).

Sie helfen dir Werte für einen gewissen Zeitraum abzulegen, damit du sie später wieder verwenden kannst.

Wie ist eine Variable aufgebaut?

Eine Variable zeichnet sich durch **drei** Bausteine aus:

1. Von welchem Datentyp ist die Variable (`int`, `string`, `float` usw.)?
2. Wie heißt die Variable?
3. Welchen Wert hat die Variable?

Tipp: Der Name einer Variablen sollte immer so deutlich wie möglich auf den Zweck bzw. Inhalt der Variablen hindeuten. Ein geeignet Name für eine Variable, die du zum speichern eines Alters verwendest, wäre beispielsweise `age` (`number1` wäre im Gegensatz ein sehr schlechtes Beispiel).

Werte zuweisen

Der Wert einer einfachen Variablen kann zu jedem Zeitpunkt verändert werden, dabei erfolgt die Zuweisung eines Wertes über das = Symbol:

```
int age = 30;
```

Tipp: Wir schließen die Zeile Code mit einem **;** ab

Wichtig ist, dass du die Variable nicht erneut definieren musst um Sie im Code zu verwenden. Folgendes Beispiel verändert den Wert der Variablen `age` im Laufe des Programms:

```
void Main() {
      int age = 30;
      Console.WriteLine(age);

      age = 18;
      Console.WriteLine(age);
```

```
        age = 90;
        Console.WriteLine(age);
}
```

Die Konsolenausgabe des Programmes würde so aussehen:
```
30
18
90
```

Wir haben den Wert der Variablen also einmal definiert (direkt am Start als Standardwert) und zweimal neu zugewiesen. **Beachte dabei, das nur beim ersten mal der Datentyp `int` angegeben wurde, danach können wir uns alleine über den Namen auf die Variable beziehen.**

Reichweite (Scope)

Je nachdem an welcher Stelle du eine Variable erstellst, wirst du feststellen, dass du sie nicht immer verwenden kannst. Dabei gilt folgende Regel:

1) Erstellst du die Variable in der Klasse, kannst du sie innerhalb der Klasse und derer Methoden verwenden.

2) Erstellst du die Variable in der Methode, kannst du sie nur innerhalb dieser einen Methode verwenden.

Tipp: Methoden unterliegen den Klassen in der Hierarchie.

Hier zwei Beispiele:

zu 1)
```
        class Program {

                int age = 18; // Definiert in der Klasse

                void Main(){
                        age = 30; // Neue Zuweisung in der Methode
                        Console.WriteLine(age);
                }
        }
```

zu 2)

```
class Program {

	void Main(){
		int age = 18; // Definiert in der Methode
		Console.WriteLine(age);
	}

}
```

Fehlerhaftes Beispiel

```
class Program {

	void Main(){
		int age = 18; // Definiert in der Methode
		Console.WriteLine(age);
	}

	void AnotherMethod(){
		age = 40; // ERROR: age ist nicht bekannt
	}
}
```

Das letzte Beispiel wirft einen Error, weil `age` in der `AnotherMethod` Methode nicht definiert ist. Wäre `age` in der Klasse und nicht in der `Main` Methode definiert, dann würde der Code einwandfrei funktionieren:

Korrigiertes Beispiel

```
class Program {
	int age = 90; // Definiert in der Klasse

	void Main(){
		age = 18;
		Console.WriteLine(age);
	}

	void AnotherMethod(){
		age = 40; // Verfügbar auch in dieser Methode
	}
}
```

Jetzt wird age in der Klasse definiert und kann in allen Methoden der Klasse verwendet werden.