Funktionen

Wir kennen ja bisher die Zeile:

```
var Zahl: Int = dice() // Jetzt ist Zahl eine beliebige Zahl zwischen 1 und 6
```

Hierbei handelt es sich um einen Aufruf einer Funktion. **Aber was genau macht eine Funktion ?**

Wie wir schon aus der Aufgabe mit dem Würfeln wissen ist, dass die Funktion „dice()“ uns einen Integer übergibt. Ein Fall für Funktionen ist also, dass Funktionen uns nach ihrer Ausführung einen Wert wiedergeben.

Gucken wir uns dafür die Funktion dice() genauer an:

```
fun dice(): Int {
    var x: Int =Random().nextInt( bound: 6)+1

    return x
}
```

Was sehen wir ?

1) Ein **fun** -> das ist unser Keyword damit das Programm weiß, dass wir an der Stelle eine Funktion haben wollen.

2) dice -> das ist der Name, den wir unserer Funktion geben. Hier kann alles beliebige stehen.

3) () -> die Klammern stehen immer bei einer Funktion dabei - normalerweise schreiben wir hier Variablen hin, die wir übergeben wollen, aber dazu später mehr.

4) : Int -> damit weisen wir (wie bei unseren Variablen) der Funktion einen Rückgabewert zu. Also einen Dateitypen. Die Funktion behandeln wir also als diesen Typen. In unserem Fall also einfach als Zahl. Wir können diese Zahl wie eine „2“ oder „5“ auch behandeln.

5) return x -> Return heißt „zurückgeben“. So wird es in jeder Programmiersprache auch behandelt. Wir geben am Ende unserer Funktion die Variable x zurück.
Das bedeutet für uns wenn wir etwas schreiben wie:

```
var Zahl: Int = dice() // Jetzt ist Zahl eine beliebige Zahl zwischen 1 und 6
```

Dass unsere Zahl mit dem Wert x oben zugewiesen wird.
-> Der Typ des Rückgabetypen muss der selbe wie oben angegeben sein.
x muss vom Typen Int sein

Wie schreiben wir also allgemein eine Funktion ?

```
fun verdoppeln(x: Int): Int {
    return x * 2
}
```

Wie wir sehen ist alles wie vorher nur steht jetzt in der Klammer noch etwas. Wir können Funktionen Werte mitgeben. In diesem Fall geben wir der Funktion einen Integer mit und lassen diesen in der Funktion verdoppeln.

Wie rufen wir die Funktion auf ?

```
var v = 4

v = verdoppeln(v)

var w = verdoppeln( x: 6)
```

Wir schreiben in die Klammer einfach irgendeinen Integer.

-> Wichtig! Wir nennen die Variable zwar in der Funktion x, aber das hat nichts damit zu tun wie ihr die Funktion aufruft. Sie hat nur einen Namen, damit ihr in der Funktion mit der Variable arbeiten könnt.

Wir können einer Funktion auch mehrere Variablen gleichzeitig übergeben (dabei werden sie mit einem Komma getrennt).

```
fun multiplikation(x: Int,y: Int): Int {
    return x * y
}
```

```
var v = 4

v = multiplikation(v, y: 2)

var w = multiplikation( x: 6, y: 2)
```

Arrays

Wir lernen einen neuen Datentyp kennen, den Array.

Ein Array können wir uns vorstellen Als ein Schrank mit beliebig vielen Fächern. Vergleichen wir das mit der Abbildung ein Schrank mit 5 Fächern und in jedes dieser Fächer kommt ein Element (hier Integer).

Kotlin - Kapitel 3

Wie erstellen wir einen Array ?

```
var v: IntArray = IntArray( size: 3)
```

Hier erstellen wir einen IntArray (einen Array vom Typ Int) auf der rechten Seite schreiben wir den Konstruktor vom Intarray zum zuweisen (das ist aber erstmal unwichtig). Ihr müsst euch erstmal nur merken was genau ihr schreibt. Ihr schreibt immer den gleichen Ausdruck. Ihr werdet für Arrays von Integer einfach immer nur die Zahl ändern. Diese sind die später vorhandenen Fächer.

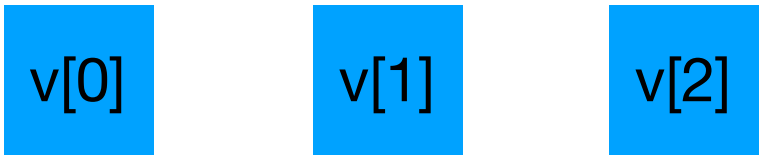
Wir haben jetzt 3 leere Fächer zur verfügung. Leer bedeutet sie haben den Wert null.

Wie arbeiten wir jetzt also mit den Fächern ?

```
v[]
```

In die Klammer muss nur noch das Fach mit dem Wir arbeiten wollen.

Damit sind unsere Fächer…



Jedes Fach können wir wie eine Variable behandeln und damit arbeiten.
-> bei einem Array fangen wir immer bei 0 an zu zählen.

Wenn wir also in unserem Array die Zahlen 1, 4 und 7 speichern wollen sieht der Code dazu so aus….

```
var v: IntArray = IntArray( size: 3)


v[0] = 1

v[1] = 4

v[2] = 7
```

-> Es gibt aber kein v[3], v[4], etc., da wir nur ein Array der Länge 3 erstellt haben!