1 Bedingte Ausführung

1.1 Zwei Alternativen

1.1.1 Aufgabe

Implementiere eine Funktion `boolToWord`, die ein *Boolean* übergeben bekommt. Wenn dieses `true` ist, soll `"Yes"` zurückgegeben werden. Wenn es `false` ist, wird `"No"` zurückgegeben.

```
boolToWord(true)
```

```
Yes
```

```
boolToWord(false)
```

```
No
```

1.1.2 Aufgabe

Implementiere eine Funktion `hoopCount`, die dich motiviert Hulla-Hoop zu trainieren. Ihr wird die Anzahl der geschafften Drehungen übergeben. Wenn mehr als 10 Umdrehungen geschafft wurden, gibt die Funktion `"Great, now move on to tricks"` zurück. Bei weniger als 10 Umdrehungen soll `"Keep at it until you get it"` zurück gegeben werden.

```
hoopCount(100)
```

```
Great, now move on to tricks
```

```
hoopCount(1)
```

```
Keep at it until you get it
```

1.1.3 Aufgabe

Deine Kollegen bitten dich darum ihnen Unterlagen zu kopieren. Implementiere eine Funktion `paperwork`, die dir hilft die Anzahl der benötigten Seiten Papier zu berechnen. Der Funktion werden die Anzahl der Kollegen und die Anzahl der Seiten pro Kollege übergeben. Wenn negative Zahlen übergeben werden, wurden fehlerhafte Daten übertragen. In diesem Fall soll `0` zurückgegeben werden.

```
paperwork(3, 5)
```

```
15
```

```
paperwork(-4, 6)
```

```
0
```

```
paperwork(2, 60)
```

```
120
```

1.1.4 Aufgabe

Implementiere eine Funktion `bonusTime`, die das Gesamtgehalt eines Mitarbeiters als String zurückgibt. Die Funktion bekommt das Grundgehalt als *Integer* und ein *Boolean*, das angibt, ob der Mitarbeiter einen Bonus bekommt. Wenn dies der Fall ist bekommt der Mitarbeiter das zehnfache seines Grundgehalt. Ansonsten bekommt er nur das Grundgehalt.

```
bonusTime(100, false)
```

```
$100
```

```
bonusTime(50, true)
```

```
$500
```

1.2 Mehrere Alternativen

1.2.1 Aufgabe

Implementiere eine Funktion `rps`, die genutzt werden kann um Schere-Stein-Papier zu spielen. Ihr werden die Entscheidungen der beiden Spieler übergeben und sie gibt zurück, wer gewonnen hat.

```
rps("rock", "scissors")
```

```
Player 1 won!
```

```
rps("scissors", "paper")
```

```
Player 1 won!
```

```
rps("rock", "rock")
```

```
Draw!
```

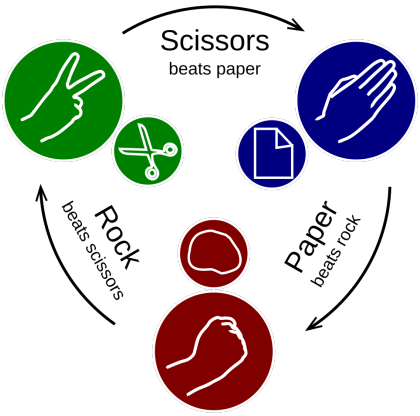

Abbildung 1: https://upload.wikimedia.org/wikipedia/commons/6/67/Rock-paper-scissors.svg

```
>>> play_rps()
 Hello player_1. Please choose rock, paper or scissors! paper
 Hello player_2. Please choose rock, paper or scissors! scissors
 Player 2 won!
```

Abbildung 2: image

1.2.2 Aufgabe

Implementiere eine Funktion `playRps`. Mit dieser Funktion kann man zu zweit and der Konsole Schere-Stein-Papier spielen. Die Funktion fordert nacheinander die Spieler auf, sich für eine der drei Möglichkeiten zu entscheiden. Anschließend wird der Gewinner bekannt gegeben. Gehe davon aus, dass beide Spieler eine korrekte Eingabe eintippen.

1.2.3 Aufgabe

Implementiere eine Funktion `getGrade`, die einer Lehrerin an einer High-School hilft, die Noten in einem Schuljahr zu berechnen. Der Funktion werden die erzielten Punkte bei drei Klassenarbeiten übergeben. Sie gibt die erzielte Gesamtnote als String zurück. In jeder Arbeit konnten maximal 100 Punkte erreicht werden.

Umrechnungstabelle	
Durchschnittliche Punktzahl	Note
$90 \leq$ Durchschnitt ≤ 100	A
$80 \leq$ Durchschnitt ≤ 90	B
$70 \leq$ Durchschnitt ≤ 80	C
$60 \leq$ Durchschnitt ≤ 70	D
$0 \leq$ Durchschnitt ≤ 60	F

```
1 getGrade(90, 91, 95)
```

```
A
```

```
getGrade(40, 60, 100)
```

```
E
```

```
getGrade(10, 50, 70)
```

```
E
```

1.2.4 Aufgabe

Passe die Funktion `getGrade` so an, dass der Durchschnitt nur einmal berechnet wird.

1.2.5 Aufgabe

Implementiere eine Funktion `evenOrOdd`, die eine ganze Zahl übergeben bekommt und `even` zurückgibt, wenn diese Zahl gerade ist. Wenn die Zahl ungerade ist, soll `odd` zurückgegeben werden.

```
evenOrOdd(5)
```

```
Odd
```

```
evenOrOdd(4)
```

```
Even
```

```
https://www.codewars.com/kata/53da3dbb4a5168369a0000fe/train/kotlin
```

1.2.6 Aufgabe

Schreibe eine Funktion `century`, die berechnet in welchen Jahrhundert ein Jahr ist.

```
century(350)
```

```
4
```

```
century(400)
```

```
4
```

```
century(401)
```

```
5
```

```
century(402)
```

```
5
```

https://www.codewars.com/kata/5a3fe3dde1ce0e8ed6000097/kotlin

1.2.7 Aufgabe

Implementiere eine Funktion `simpleMultiplication` die ein Integer übergeben bekommt. Wenn dieses Integer gerade ist, gibt sie das achtfache zurück. Wenn es nicht gerade ist, gibt sie das neunfache zurück.

```
simpleMultiplication(2)
```

```
16
```

```
simpleMultiplication(3)
```

```
27
```

https://www.codewars.com/kata/583710ccaa6717322c000105/train/kotlin

1.2.8 Aufgabe

Implementiere eine Funktion `multiple`, die ein Integer übergeben bekommt. Wenn dieses Integer durch fünf und drei teilbar ist, gibt sie `"BangBoom"` zurück. Wenn das Integer durch fünf aber nicht durch drei teilbar ist, gibt sie `"Boom"` zurück. Wenn es durch drei aber nicht durch fünf teilbar ist, gibt sie `"Bang"` zurück. Falls keine der Bedingungen erfüllt ist, so gibt sie `"Miss"` zurück.

```
multiple(2)
```

```
Miss
```

```
multiple(30)
```

```
BangBoom
```

```
multiple(20)
```

```
Boom
```

```
multiple(12)
```

```
Bang
```

https://www.codewars.com/kata/55a8a36703fe4c45ed00005b/train/kotlin