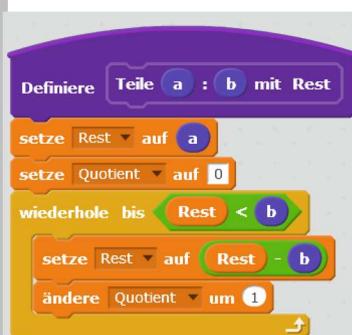


III | Kapitel 1 Unterprogramme mit Rechnungen

Dieses Scratch-Unterprogramm kennst du aus letztem Schuljahr.

Erkläre welche Aufgabe es erfüllt.

Vergleiche es mit dem entsprechendem Java-Unterprogramm.



p Sketch Kap1_Bsp

```
// Berechnet den Rest bei der Division von a durch b
int rest(int a, int b) {
    int r = a;
    while (r >= b) {
        r = r - b;
    }
    return r;
}

void setup() {
    int ergebnis = rest(93, 19);
    println(ergebnis);
}
```

Diese beiden Zeilen kann man zu
`println(rest(93, 19));`
zusammenfassen.

1 p Unterprogramme für die Division mit Rest in Java

- Übernehme das Beispiel für die Division mit Rest.
- Ergänze ein weiteres Unterprogramm zur Berechnung des Quotienten.

Bonus: Passe deine Unterprogramme an, dass sie auch mit negativen Zahlen funktionierten.

Unterprogramme

Ein Unterprogramm ist ein mehrfach wiederverwendbarer Programmblöck. Benötigt das Unterprogramm weitere Informationen verwendet man *Parameter*. Beim Ausführen des Unterprogramms werden die Parametervariablen mit den angebenden Werten belegt.

Syntax von einem Unterprogramm

```
<Rückgabetyp> <Bezeichner>(<Parameter1>, <Parameter2>, ... ) {  
    // ...  
    return <Wert>;  
}
```

Die Definition eines Unterprogramm besteht aus einem Bezeichner und der Parameterliste in der runden Klammer. Jeder Parameter wird mit Datentyp und Bezeichner angegeben. Ein Unterprogramm kann auch keine Parameter benötigen.

Ein Unterprogramm kann genau einen Wert als Ergebnis an die Stelle zurückliefern, von der es aufgerufen wird. Man nennt diesen Wert den Rückgabewert. Den Typ des Rückgabewerts wird bei der Definition angegeben. Sobald das Unterprogramm den gesuchten Wert berechnet hat, wird dieser mit der `return`-Anweisung zurückgegeben und das Unterprogramm beendet.

Bekannte Unterprogramme

Du kennst bereits die Unterprogramme `setup` und `draw` mit ihren Sonderrollen. Sie liefern kein Ergebniswert zurück. In solchen Fällen wird der Rückgabetyp `void` (engl.: „Leere“) verwendet.



```
void setup() {  
    // Wird einmalig zu Beginn  
    // des Sketches ausgeführt  
}  
void draw() {  
    // Wird wiederholt  
    // immer wieder ausgeführt  
}
```

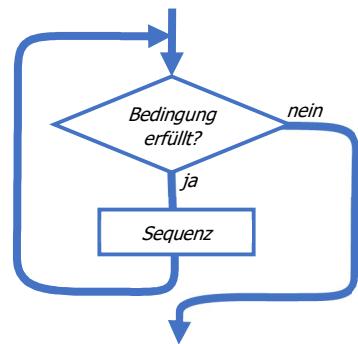


while-Schleife

Eine while-Schleife wiederholt die Sequenz im Schleifenblock so lange, bis die Bedingung erfüllt ist.

while-Schleife:

```
while ( Laufbedingung ) {  
    Sequenz  
}
```



Im Vergleich zur Zählschleife ist sie oft geschickt, wenn zu Beginn noch nicht klar ist, wie oft die Schleife durchlaufen werden muss.

GUI und CLI

GUI steht für Graphical User Interface (dt.: grafische Benutzeroberfläche.). Damit werden Programme mit Fenstern, Ein- und Ausgabefeldern und anderen grafischen Elementen zur Benutzerinteraktion bezeichnet.

Im Gegensatz dazu stehen Programme mit CLI (Command Line Interface). Diese werden über eine Text-Konsole gesteuert.

2 ③ Division mit Rest (GUI)

Kopiere dir das Projekt „Kap1A2_DivisionMitRest_Vorlage“ auf dein Home-Verzeichnis und öffne das Projekt in Processing.

```
Kap1A2_DivisionMitRest_Vorlage | gui | ▾  
1 import g4p_controls.*;  
2  
3 void setup() {  
4     size(480, 320, JAVA2D);  
5     createGUI();  
6 }  
7  
8 void draw() {  
9     background(240);  
10    // Hier nichts weiteres einfügen  
11    // Das Fenster wird von der G4P-Bibliothek gezeichnet.  
12 }  
13  
14 // Berechnet den Rest bei der Division von a durch b  
15 int rest(int a, int b) {  
16     return -1;  
17 }  
18  
19 // Berechnet den Quotient bei der Division von a durch b  
20 int quotient(int a, int b) {  
21     return -1;  
22 }  
23
```

Ersetze die Unterprogramme `rest` und `quotient` mit deinen Lösungen aus 1 und teste das Programm.

3 ③ Collatz-Folge

Die Zahlenfolge 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, ... ist nach dieser Regel erstellt: Falls die aktuelle Zahl n gerade ist, dann ist die nächste Zahl $n/2$. Falls die aktuelle Zahl ungerade ist, dann ist die nächste Zahl $3n+1$.

Kopiere dir das Projekt „Kap1A3_Collatz_Vorlage“ auf dein Home-Verzeichnis und öffne das Projekt in Processing.

- Vervollständige das Unterprogramm `istGerade`. Es hat den Rückgabetyp boolean. Das heißt es kann nur die Werte `true` oder `false` zurückgeben. Es soll `true` zurückgeben, wenn der Wert des Parameters gerade ist.
Tipp: Verwende den Rest bei der Division durch 2 und gehe ähnlich vor wie im Unterprogramm `rest` aus 1.
- Vervollständige das Unterprogramm `naechsteCollatzZahl`. Es soll nach der obigen Regel die nächste Zahl der Zahlenfolge berechnen und zurückgeben.
- Wann erreicht die Collatzfolge für die Startzahl 43 die Zahl 1?
- Verändere das Programm im `setup`-Block, dass die Folge für jede Startzahl bis zur Zahl 1 berechnet wird. Ersetze dazu die `for`-Zählschleife durch eine `while`-Schleife.

4 ③ Collatz-Folge GUI

Kopiere dir das Projekt „Kap1A4_CollatzGUI_Vorlage“ auf dein Home-Verzeichnis und öffne das Projekt in Processing.

- Kopiere deine Lösungen der Unterprogramme aus 3.
- Vervollständige das Unterprogramm `laengeCollatzFolge`. Es soll die Anzahl der Schritte in der Collatz-Folge, bis die Zahl 1 erreicht wird, zählen.
- Vervollständige das Unterprogramm `zeichneDiagramm` und berechne darin die Länge der Collatz-Folgen für die Startzahlen bis 1000. Du kannst das bereits vorhandene Unterprogramm `void setztePunkt(int startzahl, int laenge)` aufrufen, um im Diagramm zu zeichnen.

