Greedy-Münzwechsler

(a) Nehmen Sie an, es stehen beliebig viele 5-Cent, 2-Cent und 1-Cent-Münzen zur Verfügung. Die Aufgabe besteht darin, für einen gegebenen Cent-Betrag möglichst wenig Münzen zu verbrauchen. Entwerfen Sie eine Methode

```
1 public void wechselgeld (int n)
```

die diese Aufgabe mit einem Greedy-Algorithmus löst und für den Betrag von n Cent die Anzahl $c5$ der 5-Cent-Münzen, die Anzahl $c2$ der 2-Cent-Münzen und die Anzahl $c1$ der 1-Cent-Münzen berechnet und diese auf der Konsole ausgibt. Sie können dabei den Operator / für die ganzzahlige Division und den Operator % für den Rest bei der ganzzahligen Division verwenden. [1]

```
19   public static void wechsle(int betrag) {
20     int rest;
21     int c5 = betrag / 5;
22     rest = betrag % 5;
23     int c2 = rest / 2;
24     int c1 = rest % 2;
25
26
       ↪ System.out.println(String.format("Für den Betrag von %s Cent werden \n"
       ↪ + "%s Fünf-Cent-Münzen, \n"
27         + "%s Zwei-Cent-Münzen und \n" +
           ↪ "%s Ein-Cent-Münzen ausgegeben.", betrag, c5, c2, c1));
28   }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/muster/greedy/Muenzwechsler.java`

(b) Es kann gezeigt werden, dass der Greedy-Algorithmus für den obigen Fall der Münzwerte 5, 2 und 1 optimal ist, d. h. dass er immer die Gesamtzahl der Münzen minimiert. Nehmen Sie nun an, es gibt die Münzwerte 5 und 1. Ist es dann möglich, einen dritten Münzwert so zu wählen, dass der Greedy-Algorithmus mit den drei Münzen nicht mehr optimal ist? Begründen Sie Ihre Antwort.

> Falls der dritte Münzwert 4 ist, ist der Greedy-Algorithmus nicht mehr optimal. Der Greedy-Algorithmus benutzt zunächst so viele 5-Cent-Münzen wie möglich und dann so viele 4-Cent-Münzen wie möglich. Ein Betrag von 8 Cent wird also in eine 5-Cent und drei 1-Cent-Münzen aufgeteilt. Optimal ist aber die Aufteilung in zwei 4-Cent-Münzen.

Additum

```
3 /**
4  * Qualifizierungsmaßnahme Informatik: Algorithmen und Datenstrukturen:
```

[1] Quelle möglicherweise von `https://www.yumpu.com/de/document/read/17936760/ubungen-zum-prasenzmodul-algorithmen-und-datenstrukturen`

```
5   * Aufgabenblatt 3: Algorithmenmuster.
6   *
7   * <a href="https://www.studon.fau.de/file2521908_download.html">Angabe:
    ↪  AB_3
8   * Greedy_DP_Backtracking.pdf</a>
9   * <a href="https://www.studon.fau.de/file2521907_download.html">Lösung:
    ↪  AB_3
10  * Greedy_DP_Backtracking_Lsg.pdf</a>
11  */
12 public class Muenzwechsler {
13
14   /**
15    * Wechsle einen Cent-Betrag in die Münzen 5-Cent, 2-Cent and 1-Cent.
16    *
17    * @param betrag Geldbetrag in Cent.
18    */
19   public static void wechsle(int betrag) {
20     int rest;
21     int c5 = betrag / 5;
22     rest = betrag % 5;
23     int c2 = rest / 2;
24     int c1 = rest % 2;
25
26     System.out.println(String.format("Für den Betrag von %s Cent werden \n"
       ↪  + "%s Fünf-Cent-Münzen, \n"
27         + "%s Zwei-Cent-Münzen und \n" + "%s Ein-Cent-Münzen ausgegeben.",
           ↪  betrag, c5, c2, c1));
28   }
29
30   public static void main(String[] args) {
31     wechsle(1);
32     wechsle(20);
33     wechsle(23);
34     wechsle(42);
35   }
36
37 }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/muster/greedy/Muenzwechsler.java`