252-0027-00: Einführung in die Programmierung

Übungsblatt 4

Abgabe: 14. Oktober 2025, 23:59

Checken Sie mit IntelliJ die neue Vorlage aus, indem Sie wie in Übung 1 und 2 im Menü zuerst auf Ihren Projekt-Ordner klicken und *Git* → *Pull* wählen.

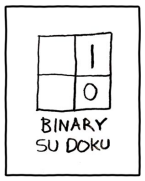
Beachten Sie, dass Sie in dieser Übung mehrere unabhängige Programme im selben IntelliJ-Projekt haben werden. Bevor Sie ein Programm starten, achten Sie deshalb darauf, dass Sie die richtige Datei ausgewählt oder im Editor geöffnet haben. *Vergessen Sie nicht, Ihren Programmcode zu kommentieren!*

Aufgabe 1: Binärdarstellung

Schreiben Sie ein Programm "Binaer.java", das eine positive Zahl Z einliest und dann die Binärdarstellung druckt.

Hinweis: Finden Sie zuerst die grösste Zahl k, so dass die Zweierpotenz $K = 2^k$ kleiner oder gleich Z ist. Dies ist das erste 1-Bit, danach können Sie die Zweierpotenz abziehen und absteigend durch die Zweierpotenzen gehen. Falls $K = 2^{k-1}$ grösser als die verbleibende Zahl ist, so ist das nächste Bit 1, sonst 0, usw.

Beispiel: Für $Z = 14$ sollte Ihr Programm `1110` ausgeben.
Beachten Sie, dass wir hier kein Zweierkomplement verwenden und nur positive Zahlen repräsentiert werden sollen.



xkcd: Su Doku by Randall Munroe
(CC BY-NC 2.5)

Aufgabe 2: Grösster gemeinsamer Teiler

Schreiben Sie ein Programm "GGT.java", das den grössten gemeinsamen Teiler (ggT) zweier ganzer Zahlen mithilfe des Euklidischen Algorithmus berechnet. Hierbei handelt es sich um eine iterative Berechnung, die auf folgender Beobachtung basiert:

1. Wenn x grösser oder gleich y ist und sich x durch y teilen lässt, dann ist der ggT von x und y gleich y;

2. andernfalls ist der ggT von x und y der gleiche wie der ggT von y und $x \,\%\, y$.

Üben Sie diesen Algorithmus zuerst von Hand an ein paar Beispielen und schreiben Sie dann das Java-Programm.

Beispiel: Für $x = 36$ und $y = 44$:

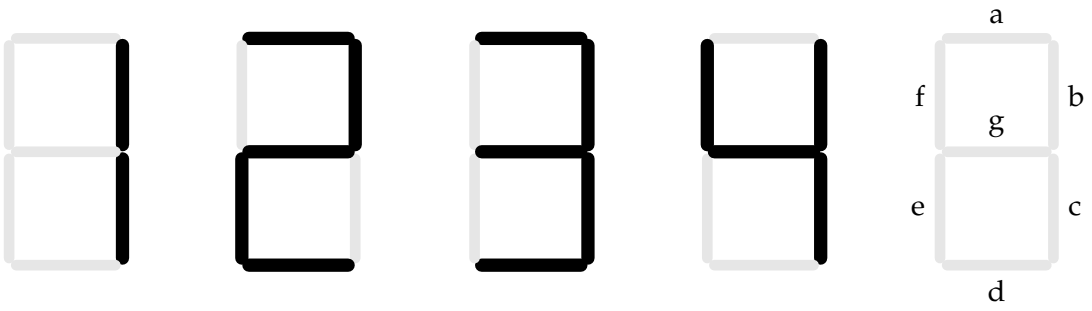
$$ggT(\underset{x}{36}, \underset{y}{44}) \overset{2.}{=} ggT(44, \underbrace{36 \,\%\, 44}_{36}) \overset{2.}{=} ggT(36, \underbrace{44 \,\%\, 36}_{8}) \overset{2.}{=} ggT(8, \underbrace{36 \,\%\, 8}_{4}) \overset{1.}{=} 4$$

Aufgabe 3: Zahlenerkennung

Für diese Aufgabe verwenden wir einen String um die erleuchtenden Segmente einer Siebensegmentanzeige zu kodieren. Die Segmente sind, wie im Bild gezeigt, von a bis g nummeriert. Die Kodierung einer möglichen Anzeige ist ein String, in welchem der Buchstabe x genau dann vorkommt, wenn das x-te Segment der Anzeige erleuchtet ist. Zum Beispiel wird die Zahl 2 kodiert durch 'abged'. Zur Einfachheit darf angenommen werden, dass kein Buchstabe mehr als einmal in der Kodierung vorkommt und dass nur die Zahlen 0 bis 9 kodiert werden.

Schreiben Sie ein Programm "Zahlen.java", das einen String, der eine Anzeige kodiert, einliest und die kodierte Zahl als Integer ausgibt. Überlegen Sie, wie viele if-Blöcke benötigt werden, um jede Zahl zu erkennen.

Tipp: Sie können `str.contains('a')` verwenden, um zu überprüfen, ob ein String `str` den Buchstaben 'a' enthält.



Aufgabe 4: Berechnungen

Schreiben Sie Ihre Lösung für diese Aufgabe in die Klasse "Calculations.java".

1. Implementieren sie die Methode `Calculations.magic7(int a, int b)`. Die Methode gibt einen Boolean zurück. Die Methode soll `true` zurückgeben, wenn mindestens eine der folgenden Bedingungen erfüllt sind, der erste Parameter ist 7, der zweite Parameter ist 7, die Summe oder Differenzen der Parameter ist 7. Ansonsten soll die Methode `false` zurückgeben. Die Differenzen von den Parametern `a` und `b` sind definiert als `a - b` und `b - a`.

Beispiele

- `magic7(2,5)` gibt `true` zurück.
- `magic7(7,9)` gibt `true` zurück.
- `magic7(14,7)` gibt `true` zurück.
- `magic7(7,14)` gibt `true` zurück.
- `magic7(5,6)` gibt `false` zurück.

2. Implementieren Sie die Methode `Calculations.fast12(int z)`. Das Argument `z` ist nicht negativ. Die Methode gibt einen Boolean zurück. Die Methode soll `true` zurückgeben, wenn `z` nahe an einem Vielfachen von `12` ist. Eine Zahl `x` ist nahe an einer Zahl `y`, wenn eine der Zahlen um maximal `2` grösser oder kleiner ist als die andere Zahl. Ansonsten soll die Method `false` zurückgeben.

   - `fast12(12)` gibt `true` zurück.
   - `fast12(14)` gibt `true` zurück.
   - `fast12(10)` gibt `true` zurück.
   - `fast12(15)` gibt `false` zurück.

   Hinweis: Mit der Funktion `Math.abs(num)` können Sie den absoluten Wert einer Zahl `num` erhalten.

Aufgabe 5: Scrabble

In dieser Aufgabe sollen Sie Scrabble-Steine legen, mittels ASCII-Art auf der Konsole. Vervollständigen Sie die Methode `drawNameSquare` in der Klasse `Scrabble`. Diese Methode nimmt einen Namen als String-Parameter und soll den Namen als in einem Quadrat angeordnete Scrabble-Steine auf der Konsole (`System.out`) ausgeben. Wenn z.B. der String `Alfred` übergeben wird, sollte folgendes Bild ausgegeben werden:

```
+---+---+---+---+---+---+
| A | L | F | R | E | D |
+---+---+---+---+---+---+
| L |               | E |
+---+               +---+
| F |               | R |
+---+               +---+
| R |               | F |
+---+               +---+
| E |               | L |
+---+---+---+---+---+---+
| D | E | R | F | L | A |
+---+---+---+---+---+---+
```

Ihr Code braucht nur Namen der Länge 3 oder länger unterstützen. Für einen Namen mit Länge 3, z.B. `Jim`, sollte die Ausgabe so aussehen:

```
+---+---+---+
| J | I | M |
+---+---+---+
| I |   | I |
+---+---+---+
| M | I | J |
+---+---+---+
```

Beachten Sie, dass Ihr Programm keinerlei andere Ausgabe als das Scrabble-Quadrat machen darf.