Aufgabenblatt 2

Kompetenzstufe 1

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihr IntelliJ-Projekt bis spätestens **Mittwoch, 22.11.2023 23:55 Uhr** in TUWEL hoch.
- Zusätzlich müssen Sie in TUWEL ankreuzen, welche Aufgaben Sie gelöst haben.
- Ihre Programme müssen kompilier- und ausführbar sein.
- Ändern Sie bitte **nicht** die **Dateinamen** und die **vorhandene Ordnerstruktur**.
- Verwenden Sie, falls nicht anders angegeben, für alle Ausgaben `System.out.println()` bzw. `System.out.print()`.
- Verwenden Sie für die Lösung der Aufgaben keine Aufrufe (Klassen) aus der Java-API, außer diese sind ausdrücklich erlaubt.
- Erlaubt sind die Klassen `String`, `Math`, `CodeDraw` und `Scanner`, es sei denn, in den Hinweisen zu den einzelnen Aufgaben ist etwas anderes angegeben.
- Bitte beachten Sie die Vorbedingungen! Sie dürfen sich darauf verlassen, dass alle Aufrufe die genannten Vorbedingungen erfüllen. Sie müssen diese nicht in den Methoden überprüfen.

In diesem Aufgabenblatt werden folgende Themen behandelt:

- Schleifen und Verschachtelung von Schleifen
- Zeichnen mit `CodeDraw` unter Verwendung von Schleifen und Verzweigungen
- Implementieren und Verwenden von Methoden
- Umgang mit der Klasse `Scanner`

Aufgabe 1 (1 Punkt)

Aufgabenstellung:

- Implementieren Sie die in Abbildung 1 gezeigte optische Täuschung. Die optische Täuschung besteht aus 10 Quadraten horizontal und 11 Quadraten vertikal. Die Quadrate haben eine Abmessung von 40×40 Pixel.
- Verwenden Sie für die Quadrate folgende Farben:
  - für dunkelgrüne Quadrate `myDrawObj.setColor(new Color(90, 90, 0))`
  - für hellgrüne Quadrate `myDrawObj.setColor(new Color(166, 166, 76))`.
- Zeichnen Sie bei jedem Kreuzungspunkt zwei übereinander liegende Rechtecke (siehe Abbildung) ein. Die längere Seite eines Rechtecks ist 20 Pixel lang und die kürzere Seite 4 Pixel. Achten Sie dabei darauf, wie die schwarzen und weißen Rechtecke angeordnet werden.

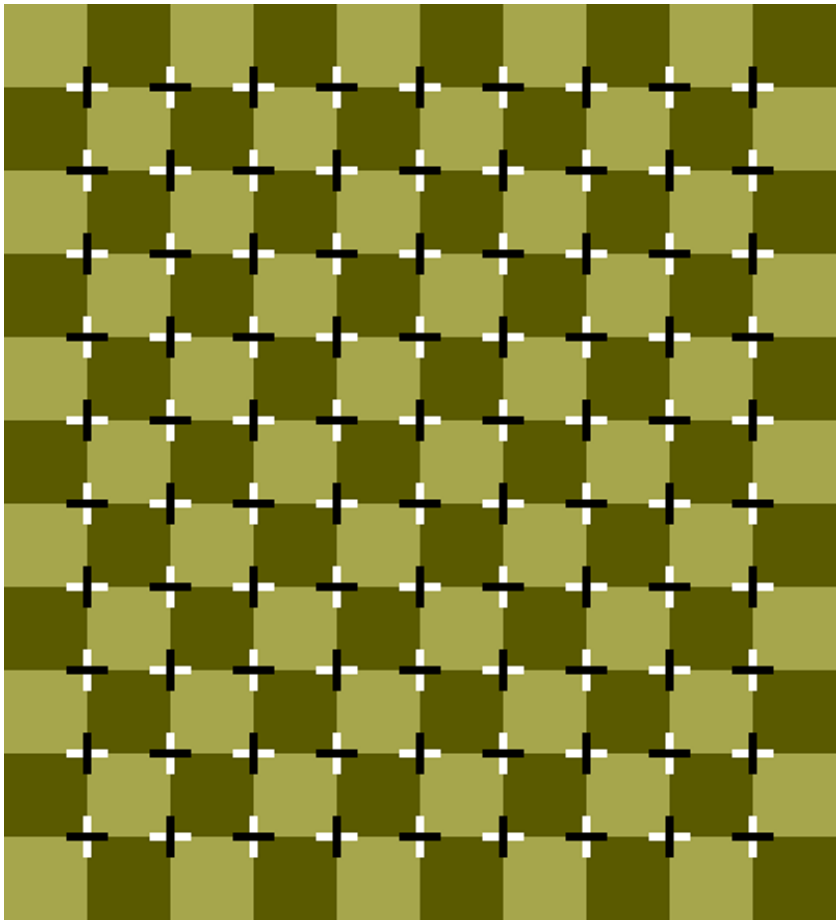


Abbildung 1: Optische Täuschung mit Quadraten und den schwarzweißen Kreuzen.

Aufgabe 2 (1 Punkt)

Aufgabenstellung:

- Implementieren Sie eine Methode `printAsciiNeighbors` mit dem Rückgabetyp `void`. Diese Methode hat einen Parameter `character` vom Typ `char` und gibt das Zeichen vor und nach dem Zeichen `character` (inklusive Zeichen `character` selbst) hintereinander mittels `System.out.print()` auf der Konsole aus. Zum Beispiel wird beim Zeichen `'o'` die Zeichenkette `"nop"`, oder beim Zeichen `'M'` die Zeichenkette `"LMN"` ausgegeben.

- Implementieren Sie eine Methode `printAlphabet` mit dem Rückgabetyp `void`. Diese Methode gibt alle Buchstaben von `'a'` bis `'z'` auf der Konsole getrennt durch Leerzeichen nebeneinander aus. Erwartete Ausgabe: a b c d e f g h i j k l m n o p q r s t u v w x y z

- Implementieren Sie eine Methode `calcSum` mit dem Rückgabetyp `int`. Diese Methode hat zwei Parameter `start` und `end` vom Typ `int`. Es sollen alle Zahlen im Intervall `[start, end]` aufsummiert werden. Das Ergebnis wird zurückgegeben. Vorbedingung: `start` $<$ `end`.

- Implementieren Sie eine Methode `isCharOnceInString` mit dem Rückgabetyp `boolean`. Diese Methode hat den Parameter `text` vom Typ `String` und den Parameter `character` vom Typ `char`. Es wird von der Methode überprüft, ob das Zeichen `character` im String `text` genau einmal vorkommt. Kommt `character` einmal vor, dann wird `true` zurückgegeben, ansonsten `false`. Vorbedingung: `text != null`.

- Implementieren Sie eine Methode `removeNumbersInString` mit dem Rückgabetyp `String`. Diese Methode hat den Parameter `text` und erstellt einen neuen String, bei dem alle Vorkommen von Ziffern in `text` entfernt werden. Am Ende der Methode wird dieser neu erstellte String zurückgegeben. Vorbedingung: `text != null`.

Aufgabe 3 (1 Punkt)

Aufgabenstellung:

(!) Für die Realisierung des Beispiels dürfen keinerlei Strings oder Arrays verwendet werden. Auch Methoden aus den Klassen `String`, `Arrays` und `Math` (auch `Math.pow()`) dürfen nicht zum Einsatz kommen. Bitte beachten Sie, dass die beiden Methoden, die ein Intervall durchsuchen, etwas Zeit bei der Ausführung benötigen können.

- Implementieren Sie eine Methode `calcCrossSum`:

  ```
  int calcCrossSum(int number)
  ```

  Diese Methode berechnet die Summe aller Ziffern (Quersumme) der Zahl `number` und gibt diese zurück. Vorbedingung: `number` > 0.

  Beispiele:
  `calcCrossSum(1)` liefert 1
  `calcCrossSum(102)` liefert 3
  `calcCrossSum(1234)` liefert 10
  `calcCrossSum(10000)` liefert 1
  `calcCrossSum(935943789)` liefert 57
  `calcCrossSum(1234567890)` liefert 45

- Implementieren Sie eine Methode `isCrossSumDividable`:

  ```
  boolean isCrossSumDividable(int number, int crossSum)
  ```

  Diese Methode prüft, ob eine Zahl `number` durch ihre Quersumme `crossSum` teilbar ist. Wenn dies der Fall ist, dann wird `true` zurückgegeben, ansonsten `false`. Vorbedingung: `number` $>$ 0 und `crossSum` > 0 und `crossSum` ist die Quersumme von `number`.

  Beispiele:
  `isCrossSumDividable(1, 1)` liefert true
  `isCrossSumDividable(24, 6)` liefert true
  `isCrossSumDividable(364, 13)` liefert true
  `isCrossSumDividable(123, 6)` liefert false

- Implementieren Sie eine Methode `countCrossSumDividable`:

  ```
  int countCrossSumDividable(int start, int end)
  ```

  Diese Methode zählt, wie viele Zahlen im Intervall [`start`, `end`] durch ihre Quersumme teilbar sind, und gibt diese Anzahl zurück.
  Vorbedingungen: `start` > 0 und `start` $\leq$ `end`.

- Implementieren Sie eine Methode `printCrossSumDividable`:

  ```
  void printCrossSumDividable(int start, int end)
  ```

  Diese Methode gibt alle Zahlen im Intervall [`start`, `end`], die durch ihre Quersumme teilbar sind, untereinander wie folgt auf der Konsole aus:
  `System.out.println("Die Zahl <number> ist durch die Quersumme <crossSum> teilbar.")`
  Vorbedingungen: `start` > 0 und `start` $\leq$ `end`.

Aufgabe 4 (1 Punkt)

Aufgabenstellung:

- Implementieren Sie einen simplen Taschenrechner, der die vier Grundrechnungsarten und die Modulo-Operation beherrscht.

  Es sollen zuerst hintereinander zwei Operanden eingelesen werden und danach noch der gewünschte Operator. Nach der korrekten Eingabe aller drei Komponenten wird die Berechnung durchgeführt und das Resultat auf der Konsole ausgegeben. Danach wird überprüft, ob eine weitere Berechnung durchgeführt werden soll, oder das Programm beendet werden soll.

  Die Berechnung und die Ausgabe kann direkt in der `main`-Methode erfolgen. Verwenden Sie bei der Division geeignete Datentypen, um auch hier für nicht ganzzahlige Ergebnisse eine richtige Ausgabe zu erhalten. Für das Einlesen verwenden Sie bitte den `Scanner`. Für das kontinuierliche Berechnen weiterer Ergebnisse müssen Sie sich eine geeignete Struktur mit einer Schleife in der `main`-Methode überlegen.

- Es müssen folgende drei Methoden für den Taschenrechner implementiert und in der `main`-Methode verwendet werden:

  1. Eine erste Methode für die Eingabe eines Operanden (wird in `main` für eine Berechnung zweimal hintereinander aufgerufen). Überlegen Sie sich was der Methode übergeben werden muss und welchen Rückgabetyp diese haben muss. Zusätzlich behandeln Sie in dieser Methode alle falschen Eingaben, die keine ganzen Zahlen (`int`-Werte) sind und geben eine Fehlermeldung auf der Konsole aus. Lesen Sie solange neue Werte ein, bis ein korrekter Integer-Wert eingegeben wurde.
  2. Eine zweite Methode für die Eingabe des Operators. Überlegen Sie sich, was der Methode übergeben werden muss und welchen Rückgabetyp diese haben muss. Zusätzlich behandeln Sie in dieser Methode alle falschen Eingaben, die nicht den fünf Zeichen (Operatoren) `'+'`, `'-'`, `'*'`, `'/'` und `'%'` entsprechen und geben eine Fehlermeldung auf der Konsole aus. Lesen Sie in der Methode solange ein, bis ein korrektes Zeichen eingegeben wurde. Auch ein korrektes Zeichen (Operator) gefolgt von mehreren falschen/korrekten Zeichen soll als falsche Eingabe interpretiert werden. Es darf nur ein Zeichen bei der Eingabe des Operators vorhanden sein.
  3. Eine dritte Methode für die Eingabe eines Zeichens, mit dem gesteuert werden soll, ob eine weitere Berechnung durchgeführt werden soll, oder ob das Programm beendet werden soll. Überlegen Sie sich, was der Methode übergeben werden muss und welchen Rückgabetyp diese haben muss. Es soll bei Eingabe von `q` und `Q` das Programm beendet werden. Mit der Eingabe `c` und `C` wird eine erneute Berechnung ermöglicht und wieder damit begonnen die zwei Operanden einzulesen. Es wird solange eingelesen, bis eines dieser vier Zeichen einzeln eingegeben wurde. Bei falschen Eingaben geben Sie eine Fehlermeldung auf der Konsole aus.

- Eine mögliche Ausgabe für einen vollständigen Durchlauf ohne Eingabefehler und Beendigung des Programms nach Berechnung könnte so aussehen:

```
Enter the first operand:
12
```

```
Enter the second operand:
6
Enter operation (+,-,*,/ or %):
+
12 + 6 = 18
Enter symbol q/Q for quitting or c/C for another calculation:
Q
Calculation finished!
```

- Eine mögliche Ausgabe für einen vollständigen Durchlauf ohne Eingabefehler und Durchführung einer weiteren Berechnung könnte so aussehen:

```
Enter the first operand:
21
Enter the second operand:
3
Enter operation (+,-,*,/ or %):
*
21 * 3 = 63
Enter symbol q/Q for quitting or c/C for another calculation:
C
Next calculation:
Enter the first operand:
45
Enter the second operand:
2
Enter operation (+,-,*,/ or %):
/
45 / 2 = 22.5
Enter symbol q/Q for quitting or c/C for another calculation:
Q
Calculation finished!
```

  Bei Fehlern kommen zusätzliche Ausgaben hinzu. Sie können hier relativ frei entscheiden, wie Sie die Ausgaben formatieren.

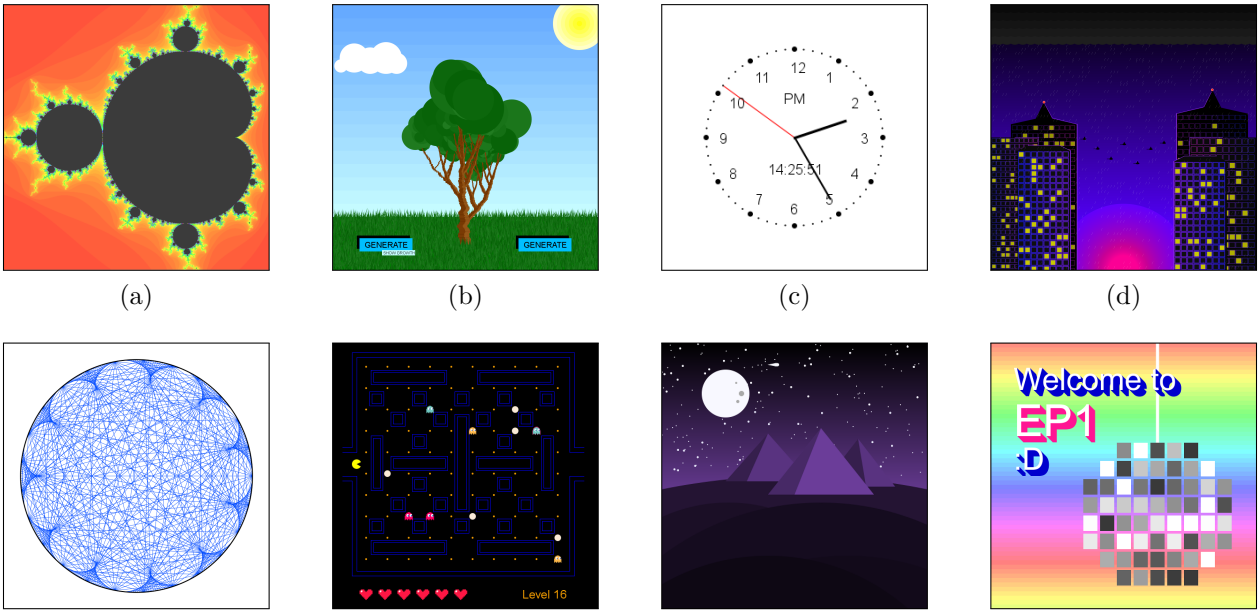
Aufgabe 5 (2 Punkte)

In dieser Aufgabe (Designaufgabe) haben Sie die Möglichkeit, Ihrer Kreativität freien Lauf zu lassen und das Gelernte umzusetzen. Sie können ein beliebiges Programm selbst erstellen. Es muss aber folgende Anforderungen erfüllen:

- Es müssen zumindest zwei Verzweigungen vorkommen.
- Es müssen zumindest zwei Schleifen vorkommen.
- Es muss die Bibliothek *CodeDraw* zum Einsatz kommen, d.h. Sie sollten eine grafische Ausgabe implementieren. Das kann entweder eine statische Zeichnung oder eine Animation sein.
- Es dürfen keine Programme aus der Vorlesung oder Übung adaptiert werden!
- Das Programm soll mindestens 50 und maximal 200 Codezeilen haben.
- Sie dürfen auch eigene Methoden implementieren und verwenden.

(!) Für diese Aufgabe gelten nicht die Einschränkungen der Java-API. Sie dürfen hier für Ihre Implementierung auch andere Aufrufe (Klassen) aus der Java-API verwenden.

In Abbildung 2 finden Sie einige Beispiele aus den vergangenen Semestern.



(a) (b) (c) (d)

(e) (f) (g) (h)

Abbildung 2: Einige Beispiele der Ergebnisse zur Designaufgabe aus den vergangenen Semestern.