Aufgabenblatt 2

Kompetenzstufe 1

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihr IntelliJ-Projekt bis spätestens **Donnerstag, 27.04.2023 12:00 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 in der `main`-Methode die in Abbildung 1 gezeigte optische Täuschung, basierend auf der *Ouchi- Illusion*[1].
- Setzen Sie die Fenstergröße auf 800×800 Pixel. Die einzelnen (weißen und schwarzen) Rechtecke haben eine Abmessung von 32×8 Pixel.
- Das Quadrat in der Mitte des Fensters hat eine Größe von 200×224 Pixel. Die Rechtecke im inneren Quadrat sind um 90° gedreht und haben eine Abmessung von 8×32 Pixel.

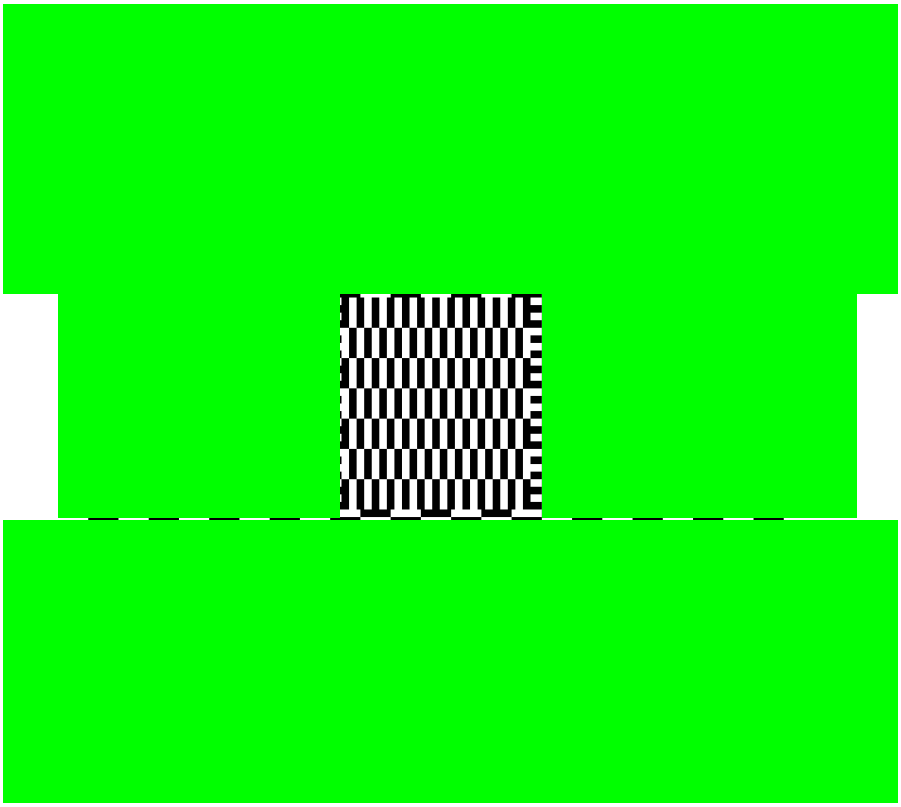


Abbildung 1: Ergebnis der optischen Täuschung laut den entsprechenden Vorgaben.

[1] Die *Ouchi-Illusion* wurde nach seinem Entdecker, dem japanischen Künstler Hajime Ouchi, benannt.

Aufgabe 2 (1 Punkt)

Aufgabenstellung:

- Implementieren Sie eine Methode `printChar` mit dem Rückgabetyp `void`. Diese Methode hat einen Parameter vom Typ `char` und gibt das Zeichen mittels `System.out.print()` auf der Konsole aus.
- Implementieren Sie eine Methode `printAlphabetPartsReverse` mit dem Rückgabetyp `void`. Diese Methode hat einen Parameter `startChar` und `endChar` vom Typ `char` und soll alle Buchstaben von `startChar` bis `endChar` rückwärts mit jeweils einem Leerzeichen voneinander getrennt ausgeben. Dazu soll die bereits vorhandene Methode `printChar` verwendet werden. Überprüfen Sie zusätzlich, ob die Zeichen `startChar` und `endChar` Kleinbuchstaben sind und daher im Bereich `'a'` bis `'z'` (beide inklusive) liegen, ansonsten wird nichts ausgegeben.
- Implementieren Sie eine Methode `calcSum` mit dem Rückgabetyp `int`. Diese Methode hat einen Parameter `end` vom Typ `int`. Es sollen alle Zahlen im Intervall `[0, end]` aufsummiert werden. Das Ergebnis wird zurückgegeben. Vorbedingung: `end` ≥ 1.
- Implementieren Sie eine Methode `isAsciiValueInRange` mit dem Rückgabetyp `boolean`. Diese Methode hat die Parameter `sign`, `start` und `end` vom Typ `char` und überprüft, ob das Zeichen `sign` im ASCII-Wert-Intervall zwischen `start` und `end` (beide inklusive) liegt. Ist `sign` in diesem Intervall, dann wird `true` zurückgegeben, ansonsten `false`.
- Implementieren Sie eine Methode `removeCharsInString` mit dem Rückgabetyp `String`. Diese Methode hat die Parameter `text` vom Typ `String` und den Parameter `sign` vom Typ `char` und erstellt einen neuen String, bei dem alle Vorkommen von `sign` 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, Array und Math (auch `Math.pow()`) dürfen nicht zum Einsatz kommen. Bitte beachten Sie, dass die beiden Methoden, die ein Intervall durchsuchen, etwas Zeit benötigen können.

- Implementieren Sie eine Methode `isNarcissisticNumber`:

  ```
  boolean isNarcissisticNumber(long number)
  ```

  Diese Methode überprüft, ob es sich bei einer gegebenen positiven ganzen Zahl `number` um eine *narzisstische* Zahl handelt. Es handelt sich dann um eine narzisstische Zahl, wenn folgende Rechenvorschrift zu einem Ergebnis führt, das mit der ursprünglichen Zahl identisch ist:

  - Es wird jede Ziffer der gegebenen Zahl mit der Anzahl der Ziffern potenziert.
  - Alle potenzierten Ziffern werden aufsummiert.

  Ist die Summe mit der ursprünglich gegebenen Zahl identisch, dann handelt es sich um eine narzisstische Zahl und es wird `true` zurückgegeben, ansonsten `false`.

  Folgende Beispiele repräsentieren narzisstische Zahlen und werden folgendermaßen berechnet:
  153 $\rightarrow 1^3 + 5^3 + 3^3 = 153$
  548834 $\rightarrow 5^6 + 4^6 + 8^6 + 8^6 + 3^6 + 4^6 = 548834$
  4679307774 $\rightarrow 4^{10} + 6^{10} + 7^{10} + 9^{10} + 3^{10} + 0^{10} + 7^{10} + 7^{10} + 7^{10} + 4^{10} = 4679307774$

  Vorbedingung: `number` ≥ 0.

- Implementieren Sie eine Methode `countNarcissisticNumbers`:

  ```
  int countNarcissisticNumbers(int start, int end)
  ```

  Diese Methode zählt, wie viele narzisstische Zahlen im Intervall [`start`, `end`] vorkommen, und gibt diese Anzahl zurück.
  Vorbedingungen: `start` ≥ 0, `end` ≥ 0 und `start` <= `end`.

- Implementieren Sie eine Methode `printNarcissisticNumbers`:

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

  Diese Methode gibt alle narzisstischen Zahlen im Intervall [`start`, `end`] nebeneinander getrennt durch Leerzeichen mittels `System.out.println()` auf der Konsole aus.
  Vorbedingungen: `start` ≥ 0, `end` ≥ 0 und `start` <= `end`.

Aufgabe 4 (1 Punkt)

Aufgabenstellung:

- Implementieren Sie einen simplen Taschenrechner, der die vier Grundrechnungsarten 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. 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`.

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

  1. Eine erste Methode für die Eingabe eines Operanden (wird in `main` dann zweimal 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 vier 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 Operator Eingabe vorhanden sein.

- Eine mögliche Ausgabe für einen vollständigen Durchlauf ohne Eingabefehler könnte so aussehen:

```
Enter the first operand:
10
Enter the second operand:
4
Enter the operation (+,-,* or /):
/
10 / 4 = 2.5
```

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

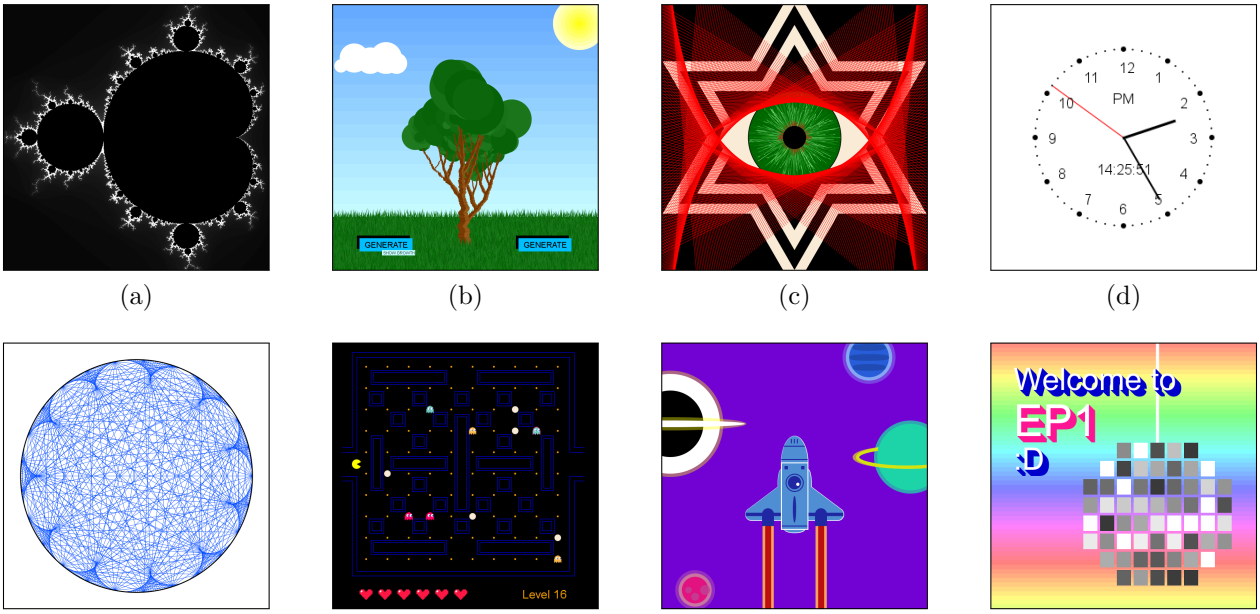
Aufgabe 5 (2 Punkte)

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.