Aufgabenblatt 2

Kompetenzstufe 1

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihr IntelliJ-Projekt bis spätestens **Donnerstag, 24.11.2022 20:00 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)

Erweitern Sie die Methode `main`:

- Implementieren Sie das in Abbildung 1 gezeigte Muster[1], bestehend aus Kreisen und Quadraten.
- Erstellen Sie ein quadratisches Fenster der Größe $ws \times ws$ Pixel mit $ws = 400$.
- Das Muster besteht aus 15 Zeilen mit jeweils 15 Kreisen, deren Radius `r` gleich $\frac{1}{60} \cdot ws$ entspricht. Der linke obere Kreis hat den Mittelpunkt bei ($x = 2 \cdot r$, $y = 2 \cdot r$). Die Kreise werden dann in gleichen Abständen ($4 \cdot r$) und entsprechend der Abbildung 1 eingezeichnet. Die Kreise werden schwarz (`Palette.BLACK`) gefüllt. Zusätzlich werden bei den gleichen Kreismittelpunkten nicht gefüllte Kreise darüber gezeichnet, die die Farbe grau (`Palette.GRAY`) haben und eine Liniendicke von 3 (`setLineWidth(...)`) aufweisen.
- Im nächsten Schritt wird das große Quadrat in der Mitte gezeichnet. Es wird weiß gefüllt und die obere linke Ecke des Quadrats befindet sich bei ($x = 0.25 \cdot ws + r$, $y = 0.25 \cdot ws + r$). Die Größe des Quadrats ist $0.5 \cdot ws - 2 \cdot r$. Zusätzlich wird ein gleich großes nicht gefülltes Quadrat mit schwarzem Rand und einer Liniendicke von 1 bei den gleichen Koordinaten gezeichnet.
- Zuletzt werden noch die kleinen nicht gefüllten Quadrate im Inneren gezeichnet. Das Quadrat oben links hat die Koordinaten ($x = 0.25 \cdot ws + 2 \cdot r$, $y = 0.25 \cdot ws + 2 \cdot r$) und eine Größe von $2 \cdot r$. Die Liniendicke hat ebenfalls den Wert von 1. Zeichnen Sie 7 Zeilen mit jeweils 7 Quadrate mit gleichen Abständen.

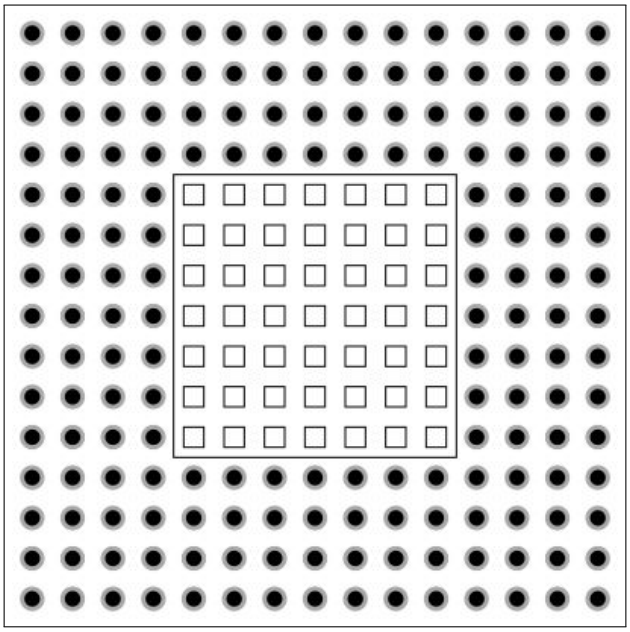


Abbildung 1: Das Ergebnis des Musters ergibt eine optische Täuschung.

[1] Optische Täuschung: Pinna and Spillmann. *A new illusion of floating motion in depth.* Perception vol. 31,12 (2002): 1501-2. doi:10.1068/p3112pp.

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` vom Typ `char` und soll alle Buchstaben von `startChar` bis zum Buchstaben `'a'` rückwärts mit jeweils einem Leerzeichen voneinander getrennt ausgeben. Dazu soll die bereits vorhandene Methode `printChar` verwendet werden. Überprüfen Sie zusätzlich, ob das Zeichen `startChar` ein Kleinbuchstabe zwischen `'a'` und `'z'` ist, ansonsten wird nichts ausgegeben.
- 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. Vorbedingungen: `start` ≥ 1, `end` ≥ 1 und `start` $<=$ `end`.
- 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` (beides inklusive) liegt. Ist `sign` in diesem Intervall, dann wird `true` zurückgegeben, ansonsten `false`.
- Implementieren Sie eine Methode `removeInString789` mit dem Rückgabetyp `String`. Diese Methode hat einen Parameter `text` vom Typ `String` und erstellt einen neuen String, bei dem alle Zeichen `'7'` bis `'9'` aus dem String `text` entfernt werden. Am Ende der Methode wird dieser neu erstellte String zurückgegeben. Vorbedingung: `text != null`.

Aufgabe 3 (1 Punkt)

Aufgabenstellung:

- Implementieren Sie eine Methode `isHappyNumber`:

  ```
  boolean isHappyNumber(int number)
  ```

  Diese Methode überprüft, ob es sich bei einer gegebenen positiven ganzen Zahl `number` um eine *fröhliche* oder eine *traurige* Zahl handelt. Es handelt sich dann um eine fröhliche Zahl (die Methode liefert `true` zurück), wenn folgende Rechenvorschrift mit einer endlichen Anzahl an Iterationsschritten zur Zahl 1 führt:

  - Es wird jede Ziffer einer Zahl quadriert.
  - Die Quadrate werden summiert.

  Auf die Summe der Quadrate wird erneut diese Rechenvorschrift angewendet. Wird die Zahl 1 erreicht, dann ist die Berechnung am Ende angekommen und es handelt sich um eine fröhliche Zahl.

  Folgende Beispiele repräsentieren fröhliche Zahlen:
  $\mathbf{23} \rightarrow 2^2 + 3^2 = 13 \rightarrow 1^2 + 3^2 = 10 \rightarrow 1^2 + 0^2 = \mathbf{1}$
  $\mathbf{97} \rightarrow 9^2 + 7^2 = 130 \rightarrow 1^2 + 3^2 + 0^2 = 10 \rightarrow 1^2 + 0^2 = \mathbf{1}$
  $\mathbf{7} \rightarrow 7^2 = 49 \rightarrow 4^2 + 9^2 = 97 \rightarrow 9^2 + 7^2 = 130 \rightarrow 1^2 + 3^2 + 0^2 = 10 \rightarrow 1^2 + 0^2 = \mathbf{1}$

  Sollte die oben beschriebene Rechenvorschrift (Ziffern quadrieren und summieren) mit einer endlichen Anzahl an Iterationsschritten zur Zahl 4 führen und nicht zur Zahl 1, dann handelt es sich um eine **traurige Zahl** und die Methode liefert `false` zurück. Nachdem die Zahl 4 erreicht wurde, beginnt ein Zyklus, der immer wieder zur Zahl 4 führt.

  Folgende Beispiele repräsentieren traurige Zahlen:
  $\mathbf{58} \rightarrow 5^2 + 8^2 = 89 \rightarrow 8^2 + 9^2 = 145 \rightarrow 1^2 + 4^2 + 5^2 = 42 \rightarrow 4^2 + 2^2 = 20 \rightarrow 2^2 + 0^2 = \mathbf{4}$
  $\mathbf{40} \rightarrow 4^2 + 0^2 = 16 \rightarrow 1^2 + 6^2 = 37 \rightarrow 3^2 + 7^2 = 58 \rightarrow 89 \rightarrow 145 \rightarrow 42 \rightarrow 20 \rightarrow \mathbf{4}$
  $\mathbf{5} \rightarrow 5^2 = 25 \rightarrow 2^2 + 5^2 = 29 \rightarrow 2^2 + 9^2 = 85 \rightarrow 8^2 + 5^2 = 89 \rightarrow 145 \rightarrow 42 \rightarrow 20 \rightarrow \mathbf{4}$

  Vorbedingung: `number` > 0.

- Implementieren Sie eine Methode `countHappyNumbers`:

  ```
  int countHappyNumbers(int start, int end)
  ```

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

- Implementieren Sie eine Methode `printHappyNumbers`:

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

  Diese Methode gibt alle fröhlichen Zahlen im Intervall [`start`, `end`] mittels `System.out.println()` auf der Konsole aus.
  Vorbedingungen: `start` > 0, `end` > 0 und `start` $<=$ `end`.

(!) Für die Realisierung des Beispiels dürfen keinerlei Strings oder Arrays verwendet werden. Auch Methoden aus diesen Klassen dürfen nicht zum Einsatz kommen.

Aufgabe 4 (2 Punkte)

Aufgabenstellung:

- Implementieren Sie das *Guessing Game* aus der Vorlesung mit zusätzlichen Erweiterungen (Vorlage auf TUWEL darf verwendet werden).
- Es müssen folgende drei Methoden implementiert und in der `main`-Methode verwendet werden:
  1. Eine Methode für die Generierung der Zufallszahl. Die generierte Zufallszahl liegt im Intervall von [0, 200].
  2. Eine Methode für das Einlesen der Eingaben. Verwenden Sie, wie in der Vorlesung vorgestellt, die Klasse `Scanner`, um die Daten von der Konsole einzulesen. Falsche Eingaben werden innerhalb der Methode abgefangen, bis ein korrekter `int`-Wert im angegebenen Intervall eingegeben wurde. Anschließend geben Sie den eingelesenen Wert zurück.
  3. Eine Methode, die eine Nachricht (`String`) für die spielende Person auf der Konsole ausgeben kann. Es soll der `System.out.println(...)`-Aufruf gekapselt werden.

  Sie können weitere Methoden implementieren, falls dadurch das Programm übersichtlicher wird.
- Nachfolgend wird der Spielablauf und der Umgang mit Sonderfällen beschrieben:
  - Spielablauf: Eine spielende Person gibt eine Zahl im Intervall von [0, 200] ein (entspricht einem Rateversuch). Danach wird ein Hinweis auf der Konsole ausgegeben, ob die gesuchte Zahl größer oder kleiner als der Rateversuch ist. Zusätzlich wird noch in jeder Runde der spielenden Person auf der Konsole mitgeteilt, wie viele Rateversuche übrig sind. Hat die spielende Person die gesuchte Zahl erraten, ist das Spiel gewonnen und beginnt von vorne. Wurde die gesuchte Zahl nach 8 Rateversuchen nicht erraten, ist das Spiel verloren und das Spiel beginnt ebenfalls von vorne. Beides wird auf der Konsole mit einer entsprechenden Meldung angezeigt und danach wird das Spiel auf den Anfang zurückgesetzt und eine neue Runde beginnt.
  - Sonderfälle: Da das Spiel endlos läuft, soll mit der Eingabe des Kleinbuchstabens `'q'` das Spiel beendet werden können. Andere falsche Eingaben wie Fließkommazahlen oder andere Zeichen werden ignoriert und es geht kein Rateversuch verloren. Auch die Eingabe einer Zahl kleiner 0 oder größer 200 wird ignoriert. Die spielende Person wird immer mit einer Meldung auf der Konsole informiert, dass die Eingabe nicht gültig ist und danach geht das Spiel weiter.

  (!) Hinweis: Für die Beendigung des Spiels nach dem Drücken der Taste `'q'`, dürfen Sie das Kommando `System.exit(0);` verwenden.

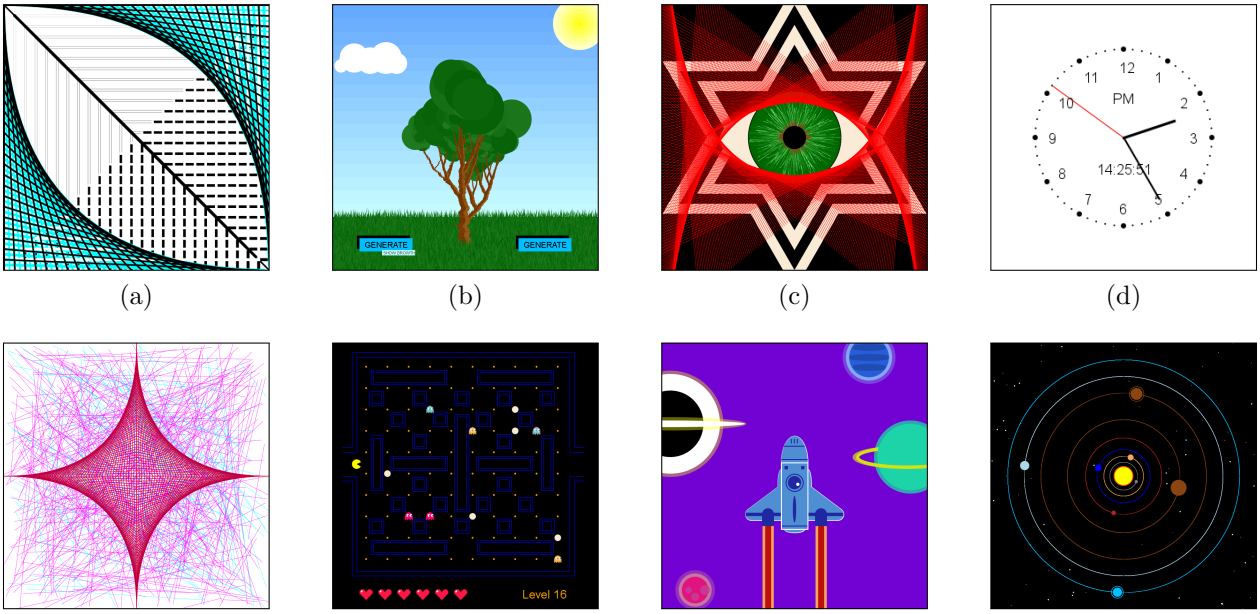
Aufgabe 5 (3 Punkte)

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 sinnvolle Verzweigungen vorkommen.
- Es müssen zumindest zwei sinnvolle 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 Anweisungen 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.