Aufgabenblatt 2

Kompetenzstufe 1

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihr IntelliJ-Projekt bis spätestens **Mittwoch, 06.11.2024 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`, `Character`, `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

Aufgabe 1 (1 Punkt)

Aufgabenstellung:

- 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.

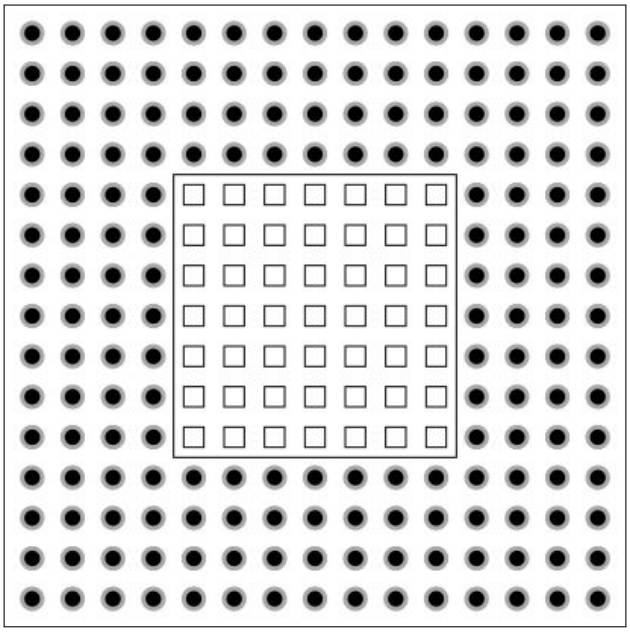


Abbildung 1: Das Ergebnis des Musters ergibt eine optische Täuschung.

[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 `printNumCharsInString` mit dem Rückgabetyp `void`. Diese Methode hat den Parameter `text` vom Typ `String`, den Parameter `startIndex` vom Typ `int` und den Parameter `numChar` vom Typ `int`. Es sollen `numChar` Zeichen von `text`, beginnend mit `startIndex`, auf der Konsole ausgegeben werden. Wenn man am Ende des Strings angelangt ist und noch Zeichen ausgegeben werden müssen, um auf die Anzahl von `numChar` Zeichen zu kommen, dann wird der String von vorne weiter ausgegeben.

  Vorbedingung: `text != null`, `text.length()` $>$ `0`, `0` $\leq$ `startIndex` $<$ `text.length()` und `numChar` $\geq$ `0`.

- Implementieren Sie eine Methode `printNumbersInInterval` mit dem Rückgabetyp `void`. Diese Methode hat zwei Parameter `start` und `end` vom Typ `int`. Es sollen alle restlos durch 3 teilbaren Zahlen im Intervall `[start, end]` absteigend auf der Konsole mittels `System.out.print()` und durch Leerzeichen getrennt ausgegeben werden.

  Vorbedingung: `start` $<$ `end`.

- Implementieren Sie eine Methode `isCharNTimesInString` mit dem Rückgabetyp `boolean`. Diese Methode hat den Parameter `text` vom Typ `String`, den Parameter `character` vom Typ `char` und den Parameter `nTimes` vom Typ `int`. Es wird von der Methode überprüft, ob das Zeichen `character` im String `text` genau `nTimes` mal vorkommt. Kommt `character` `nTimes` mal vor, dann wird `true` zurückgegeben, ansonsten `false`.

  Vorbedingung: `text != null` und `nTimes` $\geq$ `0`.

- Implementieren Sie eine Methode `changeLettersInString` mit dem Rückgabetyp `String`. Diese Methode hat den Parameter `text` vom Typ String und erstellt einen neuen String, bei dem alle Buchstaben von `'b'` bis `'z'` und `'B'` bis `'Z'` um einen Buchstaben nach links im Alphabet verschoben werden. Alle anderen Zeichen werden aus `text` so übernommen wie sie sind (aus `"ABCDYZ_abcdyz"` wird `"AABCXY_aabcxy"`). 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 diesen Klassen dürfen nicht zum Einsatz kommen.

- Implementieren Sie eine Methode `isHarshadNumber`:

  ```
  boolean isHarshadNumber(int number)
  ```

  Diese Methode überprüft, ob es sich bei einer gegebenen positiven ganzen Zahl `number` um eine *Harshad-Zahl* handelt. Es handelt sich dann um eine Harshad-Zahl (HZ), wenn die Zahl durch die Quersumme aller Ziffern der Zahl ohne Rest teilbar ist.
  Beispiele:
  1 $\rightarrow 1 \rightarrow 1 \,\%\, 1 = 0 \rightarrow$ teilbar und somit eine HZ
  4 $\rightarrow 4 \rightarrow 4 \,\%\, 4 = 0 \rightarrow$ teilbar und somit eine HZ
  13 $\rightarrow 1 + 3 = 4 \rightarrow 13 \,\%\, 4 = 1 \rightarrow$ nicht teilbar und somit keine HZ
  97 $\rightarrow 9 + 7 = 16 \rightarrow 97 \,\%\, 16 = 1 \rightarrow$ nicht teilbar und somit keine HZ
  777 $\rightarrow 7 + 7 + 7 = 21 \rightarrow 777 \,\%\, 21 = 0 \rightarrow$ teilbar und somit eine HZ
  8316 $\rightarrow 8 + 3 + 1 + 6 = 18 \rightarrow 8316 \,\%\, 18 = 0 \rightarrow$ teilbar und somit eine HZ
  9214 $\rightarrow 9 + 2 + 1 + 4 = 16 \rightarrow 9214 \,\%\, 16 = 14 \rightarrow$ nicht teilbar und somit keine HZ
  172986 $\rightarrow 1 + 7 + 2 + 9 + 8 + 6 = 33 \rightarrow 172986 \,\%\, 33 = 0 \rightarrow$ teilbar und somit eine HZ

  Vorbedingung: `number` > 0.

- Implementieren Sie eine Methode `printHarshadNumbersInInterval`:

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

  Diese Methode gibt alle Harshad-Zahlen im Intervall [`start`, `end`] mittels `System.out.print()` auf der Konsole aus.

  Vorbedingungen: `start` > 0, `end` > 0 und `start` $\leq$ `end`

  Beispiel:
  `printHarshadNumbersInInterval(51, 79)` liefert `54 60 63 70 72`

Aufgabe 4 (2 Punkte)

Aufgabenstellung:

- Implementieren Sie einen simplen Passworttester, der zu Beginn aus einem vorgegebenen `characterSet` mit Ziffern, Klein- und Großbuchstaben ein Passwort mit vorgegebener Länge `passwordLength` generiert. Dazu wird ein Zufallszahlengenerator verwendet. Mit dem Aufruf `myRand.nextInt(characterSet.length())` wird ein zufälliger Index des Strings `characterSet` ermittelt, mit dem dann ein Zeichen auch dem String `characterSet` ausgelesen werden kann. Dieser Vorgang wird so oft wiederholt, bis das Passwort der gewünschten Länge erstellt wurde.

  Danach soll mittels selbst geschriebener Methode die Entropie des Passwortes berechnet werden. Anschließend wird die Entropie und das Passwort auf der Konsole ausgegeben. Zusätzlich wird die Passwortstärke kategorisiert und dem User auf der Konsole mitgeteilt.

- Implementieren Sie eine Methode `calculateEntropy`:

  ```
  double calculateEntropy(String password)
  ```

  Diese Methode berechnet die Entropie des Passwortes `password`. Die Entropie E wird berechnet mit

  $$E = L \cdot \frac{\log_{10}(N)}{\log_{10}(2)}, \tag{1}$$

  wobei L der Länge des eingegebenen Passwortes entspricht und N die Anzahl der möglichen Zeichen (siehe Tabelle 1) für die Passworteingabe repräsentiert. Geben Sie anschließend die berechnete Entropie als `double`-Wert zurück.

N	Im Passwort enthaltene Zeichen
10	Nur Ziffern
26	Nur Groß- oder nur Kleinbuchstaben
36	Ziffern und nur ein Typ von Buchstaben
52	Keine Ziffern, aber Groß- und Kleinbuchstaben
62	Ziffern, Groß- und Kleinbuchstaben

Tabelle 1: Werte für N, je nach den enthaltenen Zeichen im Passwort.

Hinweis: Verwenden Sie die statische Methode `log10(...)` aus der Klasse `Math`, um Logarithmen zur Basis 10 zu berechnen. Die Methode soll auch funktionieren wenn zum Beispiel nur Ziffern, nur Kleinbuchstaben, usw. in einem Passwort vorhanden sind. Die Methoden `isLowerCase(...)`, `isUpperCase(...)` und `isDigit(...)` aus der Klasse `Character` können bei dieser Aufgabe hilfreich sein.

- In `main` wird die Entropie auf der Konsole ausgegeben und noch zusätzlich bewertet, wie sicher das Passwort ist. Dazu verwenden Sie folgende Skala:

Entropie	**Sicherheitsstufe und Textausgabe**
$E < 60$	weak
$60 \leq E < 120$	strong
$E \geq 120$	very strong

Testen Sie das Programm mit `passwordLength = 10`, `passwordLength = 20` und `passwordLength = 30`. Lassen Sie für die Tests die Variable `seed` auf 0, um die gleichen Ergebnisse zu erhalten. Die Ausgabe für die Länge 10, 20 und 30 sehen wie folgt aus:

```
The generated password is: 22Pbd7157K
Entropy of the password: 59.54196310386876 -> The password is: weak
```

```
The generated password is: 22Pbd7157KhLr8Ry8RZm
Entropy of the password: 119.08392620773752 -> The password is: strong
```

```
The generated password is: 22Pbd7157KhLr8Ry8RZmz66hYkdmKJ
Entropy of the password: 178.62588931160627 -> The password is: very strong
```

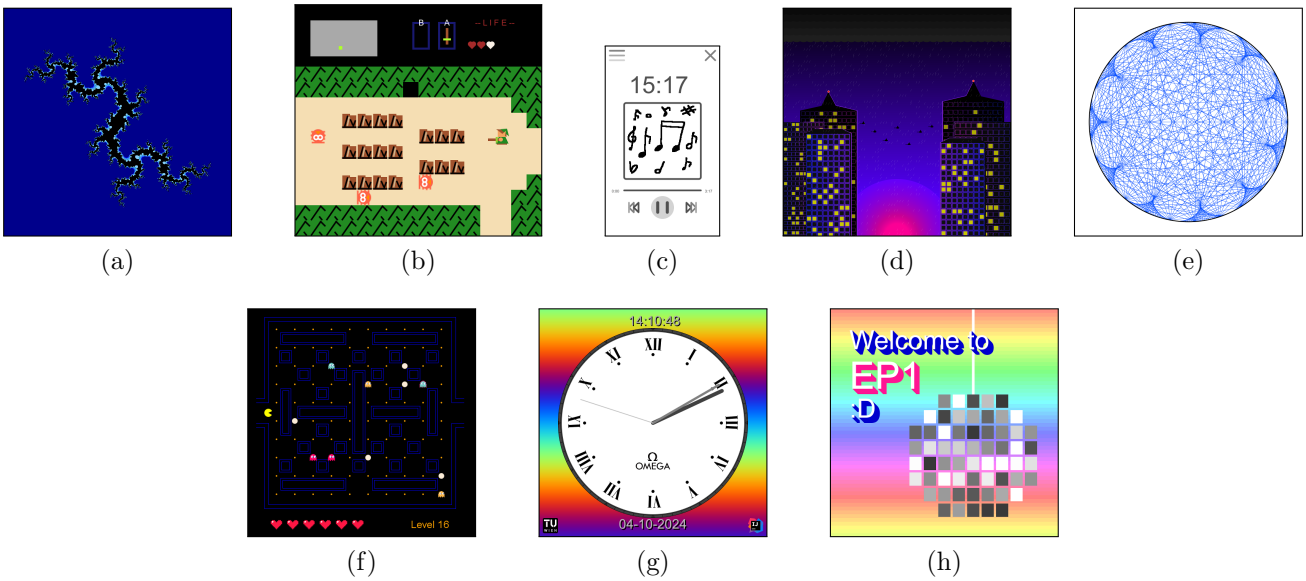
Aufgabe 5 (2 Punkte)

In dieser Aufgabe (Kreativaufgabe) 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 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.
- Bei diesem Punkt können Sie entscheiden was Sie anwenden möchten. Das heißt, einen der nachfolgenden zwei Unterpunkten müssen Sie in Ihrer Kreativaufgabe einarbeiten.
  - Sie verwenden die Bibliothek *CodeDraw*, d.h. Sie sollten eine grafische Ausgabe implementieren. Das kann entweder eine statische Zeichnung oder eine Animation sein.
  - Sie machen eine Interaktion in der Konsole mit einem User. Zum Beispiel können Sie Daten abfragen und diese visuell auf der Konsole darstellen. Oder Sie implementieren ein Spiel, das über die Konsole realisiert wird. Sie können aber auch Eingaben abfragen, die dann zu grafischen Ausgaben und Mustern auf der Konsole führen.

(!) 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 für die Kreativaufgabe unter Verwendung der CodeDraw-Bibliothek.



(a) (b) (c) (d) (e) (f) (g) (h)

Abbildung 2: Beispiele der Kreativaufgabe unter Verwendung der CodeDraw-Bibliothek.