Aufgabenblatt 2

Kompetenzstufe 1

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihr IntelliJ-Projekt bis spätestens **Freitag, 17.04.2020 13:00 Uhr** in TUWEL hoch.
- Zusätzlich müssen Sie in TUWEL ankreuzen, welche Aufgaben Sie gelöst haben.
- Ihre Programme müssen kompilierbar und ausführbar sein.
- Ändern Sie bitte **nicht** die **Dateinamen** und die **vorhandene Ordnerstruktur**.
- Bei den Aufgaben finden Sie Zusatzfragen. Diese Zusatzfragen beziehen sich thematisch auf das erstellte Programm. Sie müssen bei diesem Aufgabenblatt die Antworten als Java-Kommentare in die Dateien schreiben (unter den Lösungen).
- 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`, `StdDraw` und `Scanner` oder Klassen, die in den Hinweisen zu den einzelnen Aufgaben aufscheinen.

In diesem Aufgabenblatt werden folgende Themen behandelt:

- for-Schleife, while-Schleife
- Verschachtelung von Schleifen
- Zeichnen mit `StdDraw` unter Verwendung von Schleifen und Verzweigungen
- Implementieren und Verwenden von Methoden
- Umgang mit der Klasse `Scanner`

Aufgabe 1

Erweitern Sie die Methode `main`:

- Implementieren Sie die in Abbildung 1 gezeigte optische Täuschung.
- Setzen Sie die Fenstergröße auf 840×480 Pixel und eine Hintergrundfarbe auf den Grauwert 240. Mit dem Aufruf `StdDraw.clear(new Color(Grauwert, Grauwert, Grauwert));` können Sie einen entsprechenden Grauwert als Hintergrund definieren.
- Die Zeichnung besteht aus Kreismustern. Es sind vier Zeilen mit jeweils 7 Kreismustern vorhanden.
- Jedes Kreismuster besteht aus 6 konzentrischen Kreisen, wobei der größte Kreis einen Radius von 60 Pixel aufweist. Der Radius verringert sich bei jedem Kreis um 10 Pixel.
- Die Grauwerte bei den konzentrischen Kreisen werden bei jedem kleineren Kreis um 20 Graustufen dunkler, wobei der äußerste konzentrische Kreis einen Grauwert von 220 aufweist.
- Zeichnen Sie zusätzlich schwarze Umrandungen bei den konzentrischen Kreisen. Die Liniendicke entspricht 0.0015, die Sie mit der Methode `StdDraw.setPenRadius(...)` einstellen können.
- Zusätzlich soll bei jedem Kreismuster noch eine Raute, bestehend aus vier schwarzen Linien, gezeichnet werden (siehe Abbildung 1). Die Linienstärke soll dafür auf 0.003 gesetzt werden.

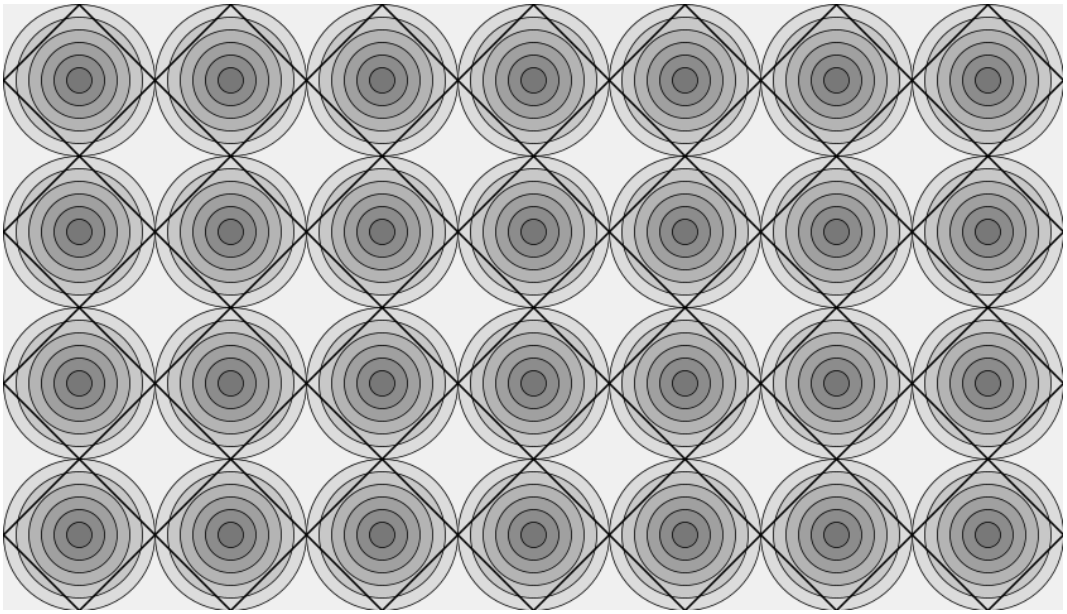


Abbildung 1: Ergebnis der optischen Täuschung laut den entsprechenden Vorgaben.

Aufgabe 2

Erweitern Sie die Methode `main`:

- Implementieren Sie ein Programm, welches die unten gezeigten Baumfiguren auf der Konsole erzeugt. Die Höhe der Figuren wird mit der Variablen `int treeHeight` gesteuert. Die Höhe gibt die Anzahl der Baumabschnitte an. Im Beispiel unten ganz links ist `treeHeight` auf 5 gesetzt, dann werden 5 Baumabschnitte gezeichnet. Der erste Abschnitt besteht aus einer Zeile mit einem Element, der zweite Abschnitt besteht aus zwei Zeilen mit 1 und 3 Elementen, der dritte Abschnitt besteht aus drei Zeilen mit 1, 3 und 5 Elementen, usw., bis der letzte Baumabschnitt mit `treeHeight` Zeilen gezeichnet wird. Beachten Sie zusätzlich, dass das oberste Element des Baumes ein `+`-Zeichen ist, die äußeren Elemente jedes Baumabschnittes aus `*`-Zeichen bestehen und alle restlichen Elemente durch `#`-Zeichen repräsentiert sind. Sie können den Wert für `treeHeight` gerne mittels Scanner einlesen.
  Beispiele:

```
    +
    *
   *#*
    *
   *#*
  *###*          +
    *            *
   *#*          *#*
  *###*          *
 *#####*        *#*          +
    *          *###*         *
   *#*          *           *#*
  *###*        *#*          *          +
 *#####*      *###*        *#*         *
*#######*    *#####*      *###*       *#*       +
```

Ausgaben (von links nach rechts) mit `treeHeight` 5, 4, 3, 2 und 1.

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

Zusatzfrage(n):

1. Kann das Beispiel mit einer einzelnen Schleife gelöst werden? Wenn ja, wie würden Sie bei der Implementierung vorgehen?

Aufgabe 3

Aufgabenstellung:

- 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
  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
  Vorbedingung(en): `number` > 0.

(!) 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 `countHarshadNumbers`:

  ```
  int countHarshadNumbers(int start, int end)
  ```

  Diese Methode zählt, wie viele Harshad-Zahlen im Intervall [`start`, `end`] vorkommen und gibt diese Anzahl zurück. Vorbedingung(en): `start` > 0, `end` > 0 und `start` $\leq$ `end`
  Beispiel:
  `countHarshadNumbers(51, 79) liefert 5`

- Implementieren Sie eine Methode `printHarshadNumbers`:

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

  Diese Methode gibt alle Harshad-Zahlen im Intervall [`start`, `end`] mittels `System.out.print()` auf der Konsole aus. Vorbedingung(en): `start` > 0, `end` > 0 und `start` $\leq$ `end`
  Beispiel:
  `printHarshadNumbers(51, 79)` liefert `54 60 63 70 72`

Zusatzfrage(n)

1. Wie viele Parameter können einer Methode übergeben werden?
2. Bei einer void-Methode dürfen keine Parameter übergeben werden! Ist diese Aussage richtig?
3. Kann eine Methode mehr als einen Rückgabewert haben?
4. Eine ganzzahlige Variable `i` wird innerhalb einer Methode deklariert. Kann auf diese Variable auch außerhalb der Methode zugegriffen werden?
5. Haben die beiden Aufrufe `func(i+1);` und `func(i++);` semantisch die gleiche Bedeutung?

Aufgabe 4

Aufgabenstellung:

- Implementieren Sie ein Spiel zum *Wörterraten*, bei dem ein zufällig gewähltes Wort unter Eingabe einzelner Buchstaben erraten werden soll und eine vorgegebene Anzahl an Rateversuchen nicht überstiegen werden darf. Dazu werden folgende vier Methoden benötigt:

  1. Eine Methode `getRandomWord` für die Generierung des Zufallswortes[1]. Diese Methode wählt aus 10 Wörtern eines zufällig aus und gibt dieses zurück (bereits vorgegeben).
  2. Eine Methode `initBackground`, die den Hintergrund zu Beginn einmal zeichnet und alle Spielkomponenten des Spieles platziert (bereits vorgegeben). Sie dürfen die Methode inhaltlich anpassen, falls Sie die Spielkomponenten anders formatieren möchten.
  3. Eine Methode `getUserInput` für das Einlesen der Eingaben. Verwenden Sie den Scanner, um die Daten von der Konsole einzulesen (muss implementiert werden).
  4. Eine Methode `runGame`, die den Spielablauf durchführt, solange bis das Spiel gewonnen bzw. verloren wurde (muss implementiert werden).

(!) Für die Realisierung dieser Aufgabe dürfen **keine** Arrays verwendet werden.

- Implementieren Sie eine Methode `getUserInput`:

  ```
  char getUserInput(Scanner gameScanner)
  ```

  Diese Methode liest die Eingaben der spielenden Person mit Hilfe eines Scanners (`gameScanner`) ein. Die Methode prüft die Eingaben, ob es sich um Buchstaben im Intervall von [a, z] bzw. [A, Z] handelt. Großbuchstaben werden in Kleinbuchstaben umgewandelt. Sollte mehr als ein Buchstabe, oder nicht im Intervall liegende Zeichen eingegeben werden, dann wird die Eingabe verworfen und die spielende Person auf der Konsole darauf hingewiesen. Es wird in der Methode so oft eine neue Eingabe entgegengenommen, bis eine gültige Eingabe erfolgt. Danach wird dieser einzelne gültige Buchstabe als `char` zurückgegeben. Vorbedingung(en): `gameScanner != null`

- Implementieren Sie eine Methode `runGame`:

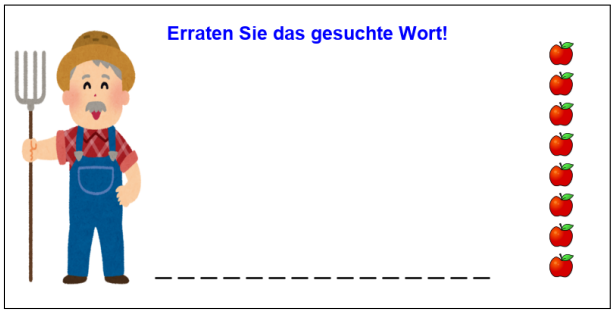
  ```
  void runGame(String guessingWord, int numGuesses)
  ```

  Diese Methode realisiert den Spielablauf. Das zu erratende Wort wird in `guessingWord` übergeben. Die maximale Anzahl der Rateversuche wird in `numGuesses` übergeben. Diese Methode beinhaltet eine Schleife, in welcher solange Buchstaben eingelesen werden, bis entweder alle Buchstaben aufgedeckt (Gewonnen) oder alle Rateversuche aufgebraucht sind (Verloren). Beim Einlesen eines Buchstabens gibt es mehrere Möglichkeiten:

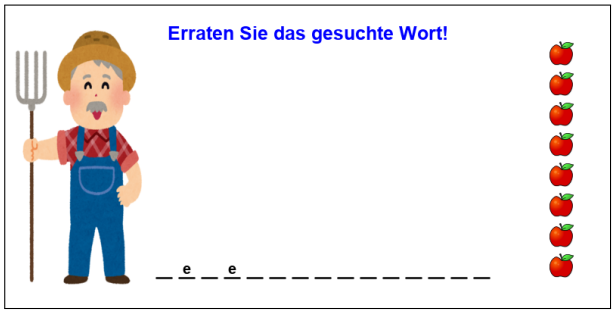
[1] Bitte beachten Sie, dass diese Methode nur mit Java-Konstrukten implementiert wurde, die wir bisher in der Vorlesung kennengelernt haben. Wir werden im Laufe des Semesters noch andere Java-Konstrukte kennenlernen, die eine bessere Implementierung dieser Methode ermöglich.

1. Kommt der eingelesene Buchstabe im gesuchten Wort vor, dann wird überall dort, wo der Buchstabe im Wort steht, der Buchstabe über den entsprechenden Unterstrich geschrieben. In diesem Fall geht kein Rateversuch verloren. Beispiel: Abbildung 2b zeigt, wie im 1. Rateversuch der Buchstabe `'e'` eingelesen wird. Danach wird `'n'` eingelesen, der ebenfalls im gesuchten Wort vorkommt (siehe Abbildung 2c).
2. Wird ein bereits erratener Buchstabe erneut eingegeben, geht ein Rateversuch verloren. Das bedeutet, dass ein Apfel weggenommen wird (siehe Abbildung 2d). Um einen Apfel wegzunehmen, muss der Apfel mit einem weißen Quadrat überzeichnet werden. (Hinweis: Um die Position der Äpfel zu berechnen, sehen Sie sich an, wie die Äpfel in `initBackground` gezeichnet werden.)
3. Der eingegebene Buchstabe kommt nicht im Wort vor. Dadurch geht ein Rateversuch verloren. Das bedeutet, dass ebenfalls ein Apfel weggenommen wird.

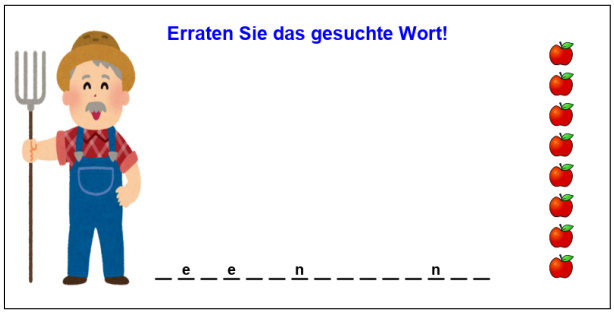
Werden alle Rateversuche aufgebraucht, ohne das Wort zu erraten, dann ist das Spiel „Verloren“ (siehe Abbildung 2f). Werden korrekte Buchstaben eingegeben, gehen keine Rateversuche verloren. Wurden alle Buchstaben eines gesuchten Wortes mit den vorhandenen Rateversuchen gefunden, ist das Spiel „Gewonnen“ (siehe Abbildung 2g). Achten Sie bei der Implementierung darauf, dass der erste Buchstabe groß geschrieben wird, auch wenn die Überprüfung über Kleinbuchstaben erfolgt!
Vorbedingung(en): `guessingWord.length > 1`, `guessingWord.length < 16`, `numGuesses > 0` und `numGuesses < 9`



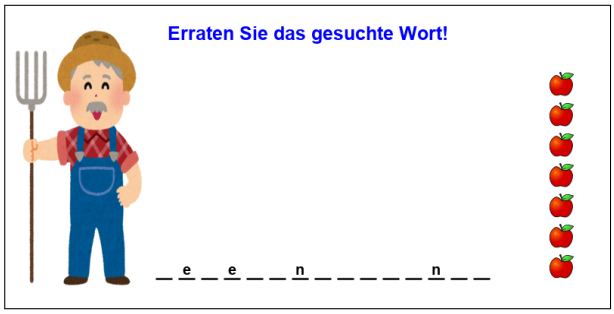
(a)



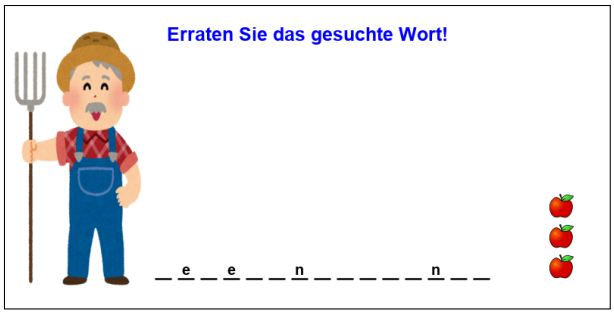
(b)



(c)

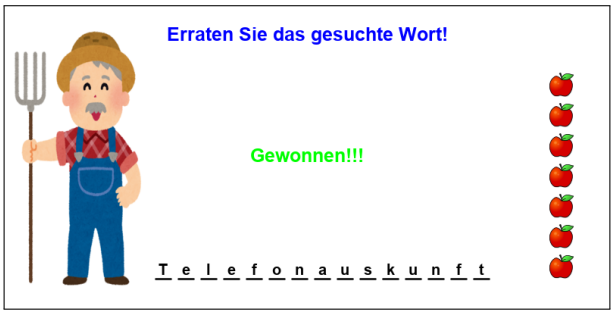


(d)



(e)

(f)



(g)

Abbildung 2: Beispielausgaben von verschiedenen Spielzuständen.