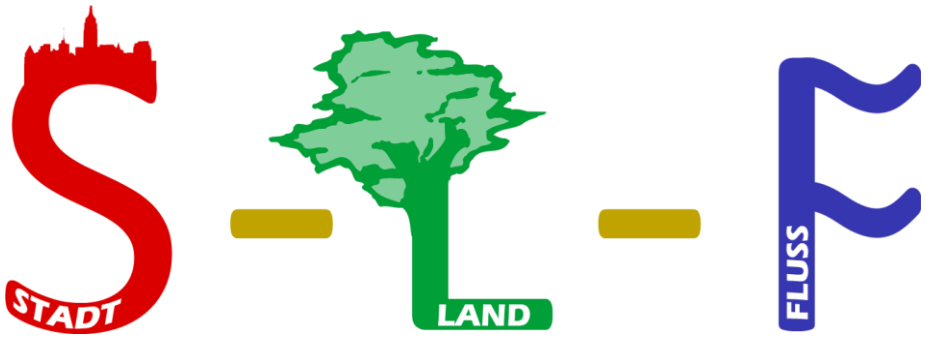

Initial _richtig _ _wichtig

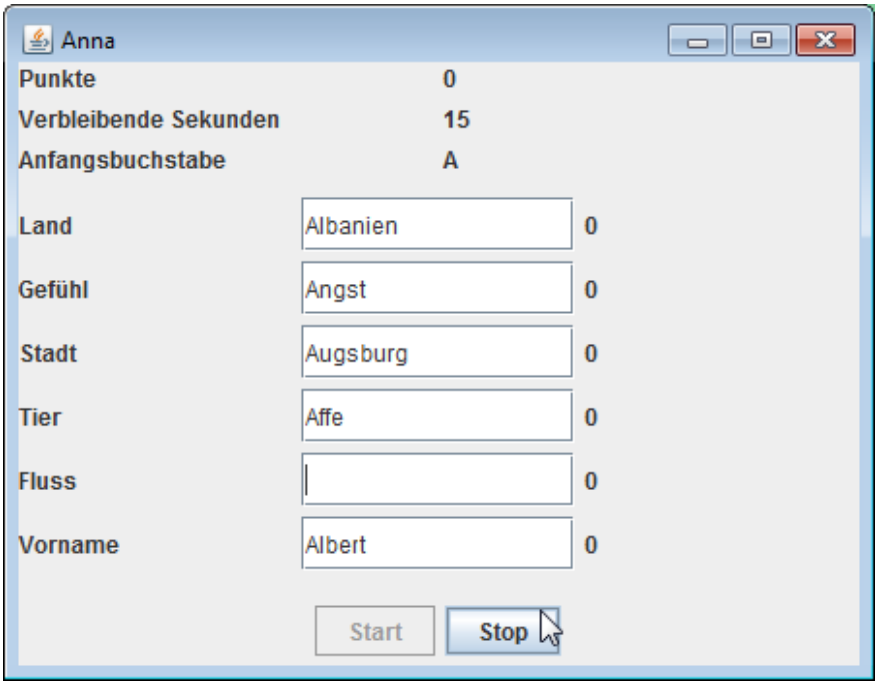

Abbildung 1

Hinweis zur Bewertung:

- 1/4 der Punkte (25%) wird nach Funktionstests Ihrer Lösung vergeben.
- 3/4 der Punkte (75%) werden entsprechend des in den Teilaufgaben angegebenen Schlüssels auf Basis des Quellcodes vergeben.

DHBW Karlsruhe, Vorlesung Programmieren I+II, Klausur

14.10.2016

Bearbeitungszeit: 120 Minuten

Aufgabe

Schreiben Sie eine Java-Anwendung **`StadtLandFluss`**, die das beliebte Wissens- und Quizspiel implementiert!

Diese Variante wird jedoch nicht mit Papier und Bleistift gespielt, sondern die Spieler treten jeweils am Rechner gegeneinander an. Jeder Spieler bekommt hierfür ein eigenes Fenster zur Verfügung gestellt.
Hinweis: Zur Vereinfachung der Aufgabe spielen alle Spieler gleichzeitig an einem Rechner.

Teilaufgabe a) **[15%]**

Ein Spieler wird repräsentiert durch die Klasse **`Player`** mit Attributen zum Speichern des Namens und der erreichten (Gesamt-)Punktzahl, zunächst 0.

Die (möglichen) Spalten des Spieles (Spiel-Spalten) sollen durch einen Aufzählungstyp **`ColumnType`** repräsentiert werden, der als einziges Attribut den Spaltentitel (Titel) kennt.

Gegeben sind die `ColumnTypes` `CITY` (Stadt), `COUNTRY` (Land), `RIVER` (Fluss), `PROFESSION` (Beruf), `ANIMAL` (Tier), `NAME` (Vorname), `SPORT` (Sportart), `FOOD` (Lebensmittel), `BEVERAGE` (Getränk) und `GAME` (Spiel).

Ein Stadt-Land-Fluss-Spiel (eine Spielrunde) soll durch die Klasse **`Game`** repräsentiert werden. Ein Spiel wird gekennzeichnet durch den gültigen Anfangsbuchstaben, eine (prinzipiell beliebige) Auswahl von Spalten, die (verbleibende) Spieldauer in Sekunden und den Zustand des Spiels, d.h. ob es bereits läuft (`boolean running`).

Die minimale bzw. maximale Anzahl von Spalten sowie die Gesamtdauer des Spieles (in Sekunden) sollen per Konstruktor festgelegt werden können:

```
public Game(int min, int max, int seconds)
```

Dabei soll überprüft werden, ob `min>=3` bzw. `max>=min` gilt. Falls nicht sollen `min=3` bzw. `max=min` gelten.

Die Methode `char createFirstCharacter()` soll einen zufälligen Buchstaben zwischen ‘A’ und ‘Z’ ermitteln (den gültigen Anfangsbuchstaben).

Ausgehend von diesem Anfangsbuchstaben soll die Methode `<return-type> createColumns()` eine zufällige Auswahl von Spalten erzeugen. Die Anzahl der Spalten soll innerhalb der Grenzen `min` und `max` (s. Konstruktor) zufällig sein. `<return-type>` ist passend zu wählen!
Jeder Spaltentyp darf nur einmal vorkommen. Die Spalten `CITY`, `COUNTRY` und `RIVER` sollen immer enthalten sein! Die Reihenfolge aller Spalten ist beliebig *(zur Vereinfachung!)*.

Die Klasse `Game` wird später erweitert!

Teilaufgabe b) **[15%]**

Die Benutzeroberfläche soll für jeden Spieler als elektronisches Spielblatt („Sheet“) realisiert werden. Entwickeln Sie eine Klasse **`Sheet`** mit zunächst folgenden Eigenschaften:

Ein Konstruktor `public Sheet(Player player, Game game)` erzeugt ein neues Spielblatt für einen bestimmten Spieler und ein bestimmtes Stadt-Land-Fluss-Spiel (`Game`).

Ein Spielblatt erzeugt eine grafische Benutzeroberfläche in einem eigenen Fenster (*s. Abbildung 2*). In der Titelzeile des Fensters soll der Name des Spielers angezeigt werden. Im oberen Bereich sollen die (Gesamt-)Punktzahl des Spielers, die verbleibenden Sekunden (anfangs leer) sowie der Anfangsbuchstabe des Spiels (anfangs leer) angezeigt werden.

Im unteren Bereich sollen zwei Buttons („Start“ und „Stop“) angezeigt werden. Zunächst ist nur der Button „Start“ aktiviert. *Hinweis*: `<JButton-Instanz>.setEnabled(...)`.

Im mittleren Bereich wird zunächst die Meldung „Kein Spiel aktiv.“ angezeigt. Diese wird später ersetzt sobald das Spiel startet, *s. Teilaufgabe e)*.

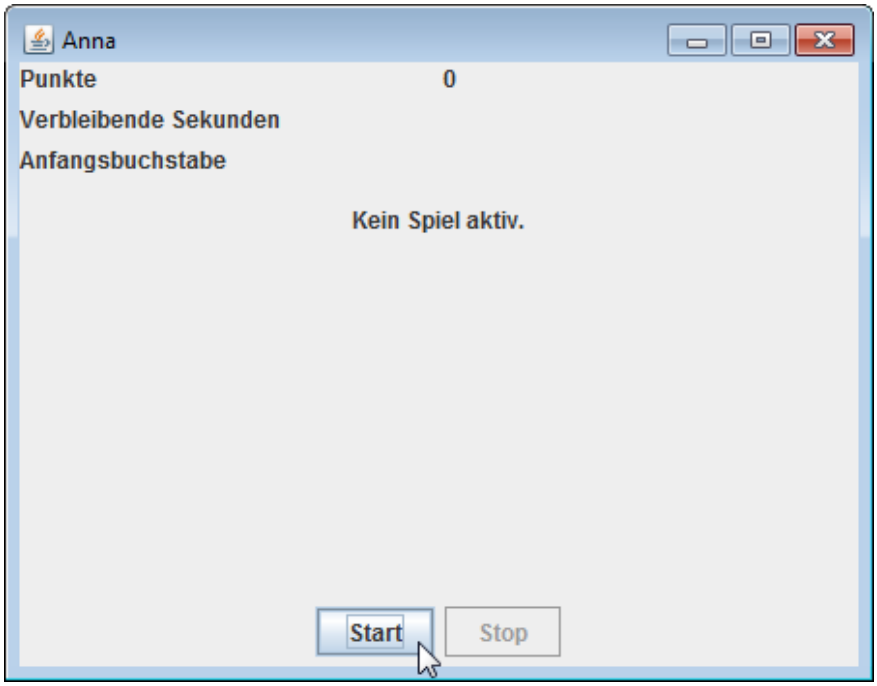

Abbildung 2

Teilaufgabe c) **[10%]**

Erweitern Sie die Klasse `Game` um eine Methode `public void register(Sheet sheet)`, mit der ein neues Spielblatt (und damit ein Spieler) für dieses Spiel registriert werden kann!

Die Klasse `Game` soll weiterhin folgende Methoden implementieren:

`public void startGame()` soll, sofern das Spiel nicht bereits läuft, ein neues Spiel starten, d.h. einen neuen Anfangsbuchstaben und neue Spiel-Spalten erzeugen. *Hinweis: Wird in Teilaufgabe e) erweitert.*

`public void stopGame()` soll ein laufendes Spiel beenden. Alle registrierten Spielblätter sollen benachrichtigt werden, z.B. um die Buttons „Start“ und „Stop“ zu aktivieren bzw. zu deaktivieren, *s. Teilaufgabe e)*, und die erreichten Ergebnisse darzustellen, *s. Teilaufgabe d)*.

Teilaufgabe d) **[10%]**

Erweitern Sie die Klasse `Game` um eine Methode zur Auswertung von eingegebenen Begriffen! Das Ergebnis ist später für jedes Spielblatt (`Sheet`) und jede Spiel-Spalte zu ermitteln. Pro Spiel-Spalte gibt es entweder 0, 5, 10 oder 20 Punkte.

Ein Spieler erhält

- 20 Punkte, wenn er als einziger Spieler einen korrekten Begriff für diese Spalte gefunden hat,
- 10 Punkte, wenn er einen korrekten Begriff gefunden hat, den kein anderer Spieler gefunden hat,
- 5 Punkte, wenn er einen korrekten Begriff gefunden hat, den auch mindestens ein anderer Spieler hat,
- 0 Punkte, wenn er keinen korrekten Begriff gefunden hat.

Hinweis: „korrekt“ sind an dieser Stelle alle Begriffe mit mehr als einem Buchstaben und dem richtigen Anfangsbuchstaben. Dies wird in Teilaufgabe f) erweitert. Groß- und. Kleinschreibung sollen keine Rolle spielen!

Teilaufgabe e) **[10%]**

Erweitern Sie die Klasse `Sheet` so, dass sie Spiel-Spalten und Ergebnisse darstellen kann, *s. Abbildung 1 auf dem Deckblatt*, und die Buttons „Start“ und „Stop“ sollen nun interaktiv sein.

Eine Spiel-Spalte soll jeweils in einer Zeile dargestellt werden: links deren Titel, rechts daneben ein Textfeld zur Eingabe des Begriffs und rechts daneben eine Möglichkeit zur Anzeige der für den angegebenen Begriff erreichten Punktzahl, zunächst 0, *s. Abbildung 1*.

Sobald ein Spiel startet (Benachrichtigung durch `Game`, *s. Teilaufgabe c)*) sollen im oberen Teil die verbleibenden Sekunden (*s. Teilaufgabe g)*) sowie der ermittelte Anfangsbuchstabe dargestellt werden.

Der Button „Start“ soll nur aktiv (`enabled`) sein, wenn das Spiel nicht läuft. Der Button „Stop“ soll dagegen nur bei einem laufenden Spiel aktiv (`enabled`) sein.

Sobald ein (beliebiger) Spieler auf seinem Spielblatt den Button „Start“ drückt soll das Spiel gestartet, d.h. die Methode `startGame()` der zugehörigen Instanz von `Game` aufgerufen werden. Entsprechend soll das Spiel beendet werden sobald ein (beliebiger) Spieler den Button „Stop“ drückt (Aufruf der Methode `stopGame()`).

Erweitern Sie die Klasse `Sheet` so, dass bei der Benachrichtigung durch `stopGame()` die erreichten Punkte, *s. Teilaufgabe d)*, neben jeder Spiel-Spalte angezeigt werden! Zusätzlich soll auch die Gesamtpunktzahl des Spielers („Punkte“) entsprechend aktualisiert und angezeigt werden. (Sobald ein neues Spiel gestartet wird sollen die Punktezahlen neben den Textfeldern jeweils wieder auf 0 stehen.)

Beim Aufruf von `startGame()` bzw. `stopGame()` sollen alle registrierten Spielblätter benachrichtigt werden, z.B. um die Spalten des Spiels darzustellen und die Buttons „Start“ und „Stop“ zu deaktivieren bzw. zu aktivieren.

Teilaufgabe f) **[10%]**

In der auf dem USB-Stick vorhandenen Datei „`validwords.txt`“ sind gültige Wörter (pro Zeile ein Wort) gespeichert.

Lesen Sie diese Datei in eine geeignete Datenstruktur ein!

Erweitern Sie Ihren Mechanismus für die Erkennung von korrekten Worten aus *Teilaufgabe d)* so, dass nur Wörter akzeptiert werden, deren Anfangsbuchstabe passt und die in der Datei stehen. Falls ein Wort nicht in der Datei steht, soll eine Dialogbox erscheinen, in der nachgefragt wird, ob es sich um eine korrekte Antwort handelt, *s. Abbildung 3*.

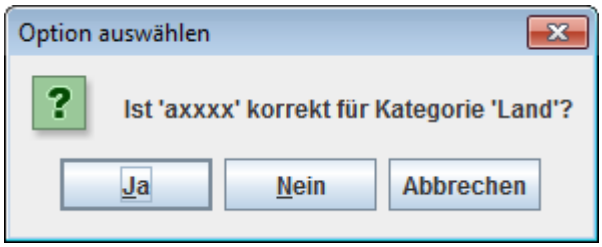

Abbildung 3

Falls ja, soll die Antwort akzeptiert werden und das Wort an die Datei „`validwords.txt`“ angehängt werden.

Hinweis: Zur Vereinfachung wird nicht geprüft ob ein gültiges Wort zu einem bestimmten `ColumnType` *passt.*

Teilaufgabe g) **[5%]**

Erweitern Sie die Klasse `Game` um einen Thread, der jeweils zu Beginn eines Spiels startet und die verbleibende Zeit bis zum Spielende herunterzählt. Sollte diese Zeit auf 0 Sekunden fallen wird das Spiel durch Aufruf der Methode `stopGame()` beendet. Der Thread soll ebenfalls beendet werden wenn sich der Zustand des Spiels auf „nicht laufend“ ändert.

Sorgen Sie dafür, dass die verbleinende Zeit bis um Spielende auf jedem Spielblatt angezeigt wird.

Hinweis: Zur Vereinfachung dürfen Sie davon ausgehen, dass ein `Thread.sleep(1000)` *genau eine Sekunde dauert. Sie können danach die verbleibende Zeit um jeweils eine Sekunde herunterzählen.*

Hinweis: Starten Sie die Anwendung mit einer Klasse `StadtLandFluss` *(s. USB-Stick), etwa:*

```
public class StadtLandFluss {
	public static void main(String[] args) {
		Game slf = new Game(4, 6, 60);
		slf.register(new Sheet(new Player("Otto"), slf));
		slf.register(new Sheet(new Player("Anna"), slf));
	}

}
```

Hinweis: Zur Vereinfachung soll beim Drücken des Schließen-Buttons eines Fensters die gesamte Anwendung beendet werden.