Die Hausaufgaben müssen von den Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog `https://seblog.cs.uni-kassel.de/ws2223/einfuhrung-in-die-informatik/` zu berücksichtigen.

Abgabefrist ist der 19.01.2023 - 23:59 Uhr

Vorbereitung

Zur Abgabe dieser Hausaufgabe muss zunächst ein neues Respository angelegt werden. Mit dem folgenden Link kannst du das Respository für Hausaufgabe 9 anlegen bzw. einsehen.

`https://classroom.github.com/a/hfad23RN`

Achte bei jeder Aufgabe auf das verlangte Dateiformat. Andere Formate werden nicht akzeptiert und folglich mit **0** Punkten gewertet. Handschriftliche Abgaben werden nicht akzeptiert.

Es sollen nur die bis zur jeweiligen Übung vermittelten Konzepte verwendet werden.

Nicht oder zu spät abgegebene (Teil-)Aufgaben werden mit 0 Punkten bewertet.

Code-Formatierung

Formatiere deinen Code vor dem Abgeben wie in Vorlesung und Übung gezeigt. Verstöße gegen Formatierungs- und Namenskonventionen führen zu Punktabzug.

Lauffähigkeit

Nicht ausführbare Abgaben werden mit 0 Punkten bewertet.

3-Credit Abschnitt

Diese Hausaufgabe zählt zu dem in Übung 1 erläuterten „3-Credit Abschnitt" der Veranstaltung. Wenn du dir unsicher bist, welche Prüfung du in dieser Veranstaltung ablegen musst, informiere dich bitte bei deinem Studienservice.

Aufgabe 1 - Einkaufsliste (30P)

Projekt-Setup

Zunächst muss per IntelliJ ein neues Java-Projekt angelegt werden, wie es in Vorlesung / Übung gezeigt wurde. Lege die Klasse `GroceryList` an, diese wird das komplette Programm beinhalten.

Logik

Es soll eine Einkaufsliste programmiert werden, die mit Hilfe von Kommandozeilenargumenten befüllt und modifiziert werden kann. Die Kommandozeilenargumente werden der `main`-Funktion als String-Array übergeben. Das erste Kommandozeilenargument nennen wir in dieser Hausaufgabe „Command". Je nach Command werden weitere Argumente erwartet, um Einträge zur Einkaufsliste hinzuzufügen, zu löschen, zu verändern oder die Liste nur anzuzeigen.

Es soll folgende „Commands" geben:

- `add <text>` - fügt <text> ans Ende der Einkaufsliste hinzu und zeigt die veränderte Liste an.
- `del <number>` - löscht den <number>. Eintrag von der Liste und zeigt die veränderte Liste an.
- `alt <number> <text>` - überschreibt den <number>. Eintrag der Liste mit <text> und zeigt die veränderte Liste an.
- `show` - zeigt die Liste an.

Wobei mit `<text>` einfache Wörter ohne Whitespace gemeint sind und mit `<number>` eine Zahl, die auf den entsprechenden Eintrag verweist. Bei der Ausgabe der Liste sollen die einzelnen Items mit Indizes versehen sein, sodass der User diese beim Löschen oder Modifizieren eingeben kann. Achte darauf, dass das erste angezeigte Element der Einkaufsliste mit „1:" ausgegeben werden muss. Intern muss auf den korrekten Array-Index umgerechnet werden. Siehe unten: Beispiel-Ablauf.

Die Einträge der Einkaufsliste sollen in der Datei `groceries.txt` gespeichert werden. Wenn die Datei nicht gefunden werden kann, muss das Programm die Datei zunächst anlegen. Dann kann der Inhalt der Datei gelesen und gegebenenfalls modifiziert werden. Die Liste muss nach der Modifikation wieder in die Datei geschrieben werden. Hierbei sollen nur die Items abgespeichert werden, die Nummerierung ist nur für die Ausgabe auf der Konsole gedacht. Beim Anlegen, Lesen und Schreiben der Datei sollen mögliche `IOException`s durch `try-catch`-Anweisungen gefangen werden und im Falle der Exception das Programm durch `System.exit()` beendet werden.

Nutze

- Exit Code 1 beim fehlerhaften Anlegen der Datei,
- Exit Code 2 beim fehlerhaften Lesen der Datei und
- Exit Code 3 beim fehlerhaften Schreiben der Datei.

Hinweis: Es sind keine `throws` an Methoden erlaubt und es sind keine weiteren Klassen erlaubt.

Beispiel-Ablauf

Folgende Gegenüberstellung zeigt links die eingegebenen Kommandozeilenargumente mit zugehöriger Konsolenausgabe und rechts den Inhalt der Datei `groceries.txt` an. Die Datei ist zu Beginn leer bzw. existiert noch nicht.

Kommandozeilenargumente: `add Brot`

```
1: Brot
```

`groceries.txt` nach Programmausführung

```
Brot
```

Kommandozeilenargumente: `add Käse`

```
1: Brot
2: Käse
```

`groceries.txt` nach Programmausführung

```
Brot
Käse
```

Kommandozeilenargumente: `add Tomaten`

```
1: Brot
2: Käse
3: Tomaten
```

`groceries.txt` nach Programmausführung

```
Brot
Käse
Tomaten
```

Kommandozeilenargumente: `alt 2 Cola`

```
1: Brot
2: Cola
3: Tomaten
```

`groceries.txt` nach Programmausführung

```
Brot
Cola
Tomaten
```

Kommandozeilenargumente: `del 1`

```
1: Cola
2: Tomaten
```

`groceries.txt` nach Programmausführung

```
Cola
Tomaten
```

Kommandozeilenargumente: `show`

```
1: Cola
2: Tomaten
```

`groceries.txt` nach Programmausführung

```
Cola
Tomaten
```

Fehlerhafte Eingaben

Da der User alle möglichen Eingaben tätigen kann, können z. B. unbekannte Commands oder negative Zahlen eingegeben werden. Das Programm soll bei fehlerhaften Eingaben nicht abstürzen, sondern diese Fälle entsprechend abfangen und eine passende Fehlermeldung auf der Konsole ausgeben. Anschließend soll `System.exit()` mit dem jeweils gegebenen Fehlercode verwendet werden:

- Wird kein Command übergeben, soll eine entsprechende Fehlermeldung ausgegeben werden. Anschließend sollen die verfügbaren Commands und eine Erklärung ausgegeben werden. Exit Code 4
- Wird ein unbekanntes Command übergeben, soll eine entsprechende Fehlermeldung ausgegeben werden. Anschließend sollen die verfügbaren Commands und eine Erklärung ausgegeben werden. Exit Code 5
- Werden bei einem Command zu wenig Argumente mitgegeben, soll eine Fehlermeldung ausgegeben werden, in der steht, wie der Command korrekt benutzt wird. Exit Code 6
- Erwartet ein Command ein Argument mit einer Zahl und bekommt einen String, der

nicht entsprechend umgewandelt werden kann, soll eine Fehlermeldung ausgegeben werden, in der steht, wie der Command korrekt benutzt wird. Nutze hierbei eine `try-catch`-Anweisung, um die `NumberFormatException` zu fangen. Exit Code 7

- Erwartet ein Command ein Argument mit einer Zahl, muss diese ein gültiger Index auf der angezeigten Liste sein. Beispielsweise negative oder zu hohe Zahlen außerhalb der Liste sind nicht zulässig. Solche Fälle müssen abgefangen und eine entsprechende Fehlermeldung ausgegeben werden. Exit Code 8

Abgabe

Gib deine Lösung in einer Datei mit dem Namen `GroceryList.java` im GitHub-Repository zur aktuellen Hausaufgabe ab. Die Datei kannst du aus der Projektstruktur in IntelliJ direkt in dein GitHub-Repository per Drag and Drop einfügen, wie in der Übung gezeigt.