PM1/Projekt 2

Aufgabe

Programmieren Sie eine einfache Textverarbeitung mit Ein- und Ausgabe über die Konsole. Diese soll als eigenständige Anwendung laufen und über den folgenden Befehlssatz gesteuert werden:

ADD [n]	Ruft nach der Eingabe dieses Kommandos zur Eingabe des Textes für den Absatz auf. Der Absatz wird an Position *n* eingefügt. Fehlt die Absatznummer, wird dieser am Ende angefügt.
DEL [n]	Löscht einen Absatz. Fehlt die Absatznummer, wird der letzte Absatz gelöscht.
DUMMY [n]	Fügt einen fest einprogrammierten Blindtext ein. Wird keine Absatznummer *n* angegeben, wird der Absatz am Ende angefügt.
EXIT	Beendet das Programm.

FORMAT RAW	Setzt das Ausgabeformat auf die Ausgabe der Absätze mit vorangestellter Absatznummern. Dies ist das eingestellte Standardverhalten.
	Ausgabeformat:
	`<1>: <text>`
	…
	`<n>: <text>`

FORMAT FIX <b>	Setzt das Ausgabeformat auf eine Ausgabe mit einer maximalen Spaltenbreite von *b* Zeichen ein. Das Umbruchverhalten ist wie folgt:
	• Umgebrochen werden darf grundsätzlich nur nach einem Leerzeichen.
	• Das Leerzeichen, nach welchem umgebrochen wird, zählt nicht zur Zeilenlänge dazu. Es wird noch auf der aktuellen Zeile ausgegeben, auch wenn es möglicherweise nicht mehr draufpasst.
	• Findet sich nach einem Umbruch keine Umbruchstelle innerhalb der Spaltenbreite, darf nach der Spaltenbreite umgebrochen werden.
	Beispiel für FORMAT FIX 20.
	<u>Gespeicherter Absatztext:</u>
	`Virtute praecedunt, quod fere cotidianis proeliis cum Germanis contendunt, septentr ionesimmensoslongusw ordos ionesimmensoslongusws.`
	<u>Ausgabe:</u>
	`Virtute praecedunt,`
	`quod fere cotidianis`
	`proeliis cum`
	`Germanis contendunt,`
	`septentr`
	`ionesimmensoslongusw`
	`ordos`
	`ionesimmensoslongusw`
	`s.`

INDEX	Gibt einen Index (Wortverzeichnis) aller Begriffe aus, die über alle Absätze gesehen öfter als dreimal vorkommen. Ein Begriff beginnt mit einem Grossbuchstaben. Der Index listet die Absätze, wo der jeweilige Begriff vorkommt, als Komma getrennte Zahlenfolge auf. Format: `Tigurini 2,5,10,11`
PRINT	Ausgabe des Textes gemäss dem aktuell eingestellten Ausgabeformat.
REPLACE [n]	Ruft nach der Eingabe dieses Kommandos zuerst zur Eingabe eines zu suchenden Wortes oder Textteils im Absatz *n* auf und anschliessend zur Eingabe des Textes, mit dem das Gesuchte ersetzt werden soll. Das Suchen und Ersetzten erfolgt pro Absatz, nicht über Absatzgrenzen hinweg. Wird keine Absatznummer *n* angegeben, wird der letzte Absatz geändert.

Anforderungen

- Ihre Anwendung muss eigenständig laufen, also durch Aufruf von `main()` und ohne BlueJ.
- Schreiben Sie Rückmeldungen zu Fehlern (z.B. «Unbekanntes Kommando») statt auf die Standardausgabe (`System.out`) auf die Standardfehlerausgabe (`Sytem.err`). Die Syntax ist analog zu der von System.out bekannten ("`System.err.println`").
- Sie dürfen nur die bis zur Abgabe durchgenommenen Konstrukte benutzen. Dies schliesst insbesondere **ein**: Den Stoff im BlueJ-Buch bis und mit Kapitel 9, JUnit Testing, Packages zur Verwaltung Ihres Codes und das Einbinden von Paketen aus java.util.*
- Als "Absatz" definieren wir eine über die unten beschriebene `Scanner.nextline()`-Methode eingelesenen String. Validieren Sie den erhaltenen Input und filtern Sie alle Zeichen raus, die keine Buchstaben des Alphabets (a-z), Umlaute (ä, ö, ü), Ziffern 0-9 oder ein Zeichen aus der folgenden Zeichenmenge sind: .,:;-!?'()"%@+*[]{}/\&#$
- Ihr Code ist in Javadoc dokumentiert. Mindestens alles, was public ist (Klassen, Methoden, Konstruktoren, …), jedoch ohne Getter/Setter.
- Sie haben eine sinnvolle Aufteilung in Klassen und Klassendefinitionen gefunden. Das Klassendiagramm ist mit hochzuladen. Es ist die in PROG1 vermittelte Notation zu verwenden. Abweichungen davon müssen als Legende mitgeliefert werden.
  https://moodle.zhaw.ch/pluginfile.php/397809/mod_folder/content/0/klassendiagramm.pdf
- Sie haben sinnvolle Test Cases für die Index-Funktionalität definiert und in JUnit implementiert. Die Dokumentation der Test Cases mit mindestens folgenden Elementen pro Test Case: Kurzbeschrieb, Äquivalenzklasse, ob positiver/negativer Test, Ausgangszustand, Input resp. ausgeführte Aktionen und erwarteter Output resp. Zustand laden Sie mit hoch.
- Wenden Sie die relevanten Abschnitte im Clean-Code-Handbuch an.
- Die zur Übersetzung und Start nötigen Aufrufe sind im README für Ihr Repo dokumentiert.

Hinweise

- Achten Sie auf Kohäsion und Kopplung und beachten die Hinweise im Clean-Code-Handbuch.
- Fehlervermeidung und JUnit wird in Programmieren 1 durchgenommen. Planen Sie genügend Zeit ein, um Ihre Tests zu definieren, umzusetzen und allfällige Fehler zu korrigieren.
- Für Management und Dokumentation von Arbeitsschritten empfehlen wir das Issue Tracking in GitHub. Andere Lösungen sind denkbar.
- Eine Quelle für Blindtext ist https://lipsum.com/. Andere Texte sind möglich.

Vorgehen

Ihre Dozierenden halten ein Kick-Off für das Projekt ab und geben den für Ihre Klasse gültigen Abgabetermin bekannt.

Wir empfehlen, etwa folgendermassen vorzugehen:

- Stellen Sie die grundlegenden Spezifikationen aus der Aufgabenstellung zusammen.
- Einigen Sie sich in der Gruppe auf ein Klassenmodell. Dokumentieren Sie dieses.
- Vereinbaren Sie Zuständigkeiten und einen groben Plan
- Erarbeiten Sie die Klassendefinition und setzen Sie diese um.
- Definieren Sie Ihre Test Cases und führen Sie diese aus.

Abgabe

- Legen Sie für das Projekt ein Repository gemäss der bereits aus Projekt 1 bekannten Anleitung an. Der von Ihnen erarbeitete Code muss zur Abgabe bis zur bekanntgegebenen Deadline dort hochgeladen werden. Abgaben auf anderen Services (github.com, OneDrive, etc.) werden nicht gewertet.

Bewertung

Für diese Aufgabe erhalten Sie maximal 15 Punkte. In die Bewertung fliessen folgende Kriterien ein:

Allgemeine Anforderung (all-or-nothing)

- Voraussetzung für Punkterteilung: Das Programm ist lauffähig. Ein nicht lauffähiges Programm erhält 0 Punkte.

Ein lauffähiges Programm wird in den folgenden beiden Bereichen beurteilt. Für Teil- oder Nichterfüllung werden Abzüge in Ansatz gebracht. Das Nichterfüllen einer Unterkategorie kann zum vollständigen Abzug in einem Bereich führen. Es werden keine Negativpunkte vergeben.

Bereich Entwicklung (11 P.)

- Ihr Programm besitzt die geforderte Funktionalität.
- Sie halten die Vorgaben hinsichtlich einsetzbarer Konstrukte und Clean Code ein.
- Sie haben eine gute Klassenaufteilung gefunden.
- Sie haben das Klassendiagramm auf GitHub hochgeladen. Es entspricht den Anforderungen.
- Sie haben sinnvolle Test Cases für die Index-Funktion definiert, diese in JUnit umgesetzt und erfolgreich ausgeführt.
- Sie haben die Dokumentation der Test Cases auf GitHub hochgeladen. Die Dokumentation entspricht den Anforderungen.

Bereich Vorgehen und Dokumentation (4 P.)

- Ihr Code ist mit JavaDoc dokumentiert (s.a. Hinweis unter "Anforderungen")
- Alle Gruppenmitglieder haben in nennenswertem Umfang **Code** beigetragen und auf GitHub eingecheckt. Check durch GitHub Log.

Beispielcode zur Konsoleneingabe

Die Eingabe von Text ist im BlueJ-Buch, Kapitel 6 erwähnt und in Kapitel 14.9. genauer beschrieben. Wenn Sie Probleme bei der Anwendung der Texteingabe haben, wenden Sie sich an Ihre Dozierenden. Die Anwendung der Scanner-Klasse sollte nach dem gegebenen Muster jedoch einfach gelingen.

In der Grundfunktionalität sieht sie folgendermassen aus:

```
import java.util.Scanner;
/**
 * Vereinfachte Demonstration Scanner-Funktion für Konsole
 * Modifiziert von BlueJ InputReader.java
 * https://bluej.org/objects-first/resources/projects.zip
 * @author Für die Anpassungen: berp
 * @version 2019-10-27
 */

public class InputReaderMod {
   private Scanner scanner;
/**
 * Main-Methode
 * @param args Eingabe vom Typ String
 */

   public static void main(String[] args) {
      InputReaderMod reader = new InputReaderMod();
      System.out.print("Please enter your input: ");
      System.out.print("Input was: "+reader.getInput()+"\n");
   }

/**
 * Konstruktor erzeugt Objekt vom Typ Scanner
 */
   public InputReaderMod() {
      scanner = new Scanner(System.in);
   }

/**
 * Lies Eingabezeile
 * @return Eingabezeile
 */
   public String getInput() {
      return scanner.nextLine();
   }
}
```

Quelle: Anpassung von `InputReader.java` (englischsprachige Fassung der im Buch in Kap. 6 erwähnten Klasse `EingabeLeser.java`) aus dem Unterverzeichnis `projects/chapter06/tech-support-complete` im Archiv https://bluej.org/objects-first/resources/projects.zip)