Universität Leipzig
Institut für Informatik

WS 20/21

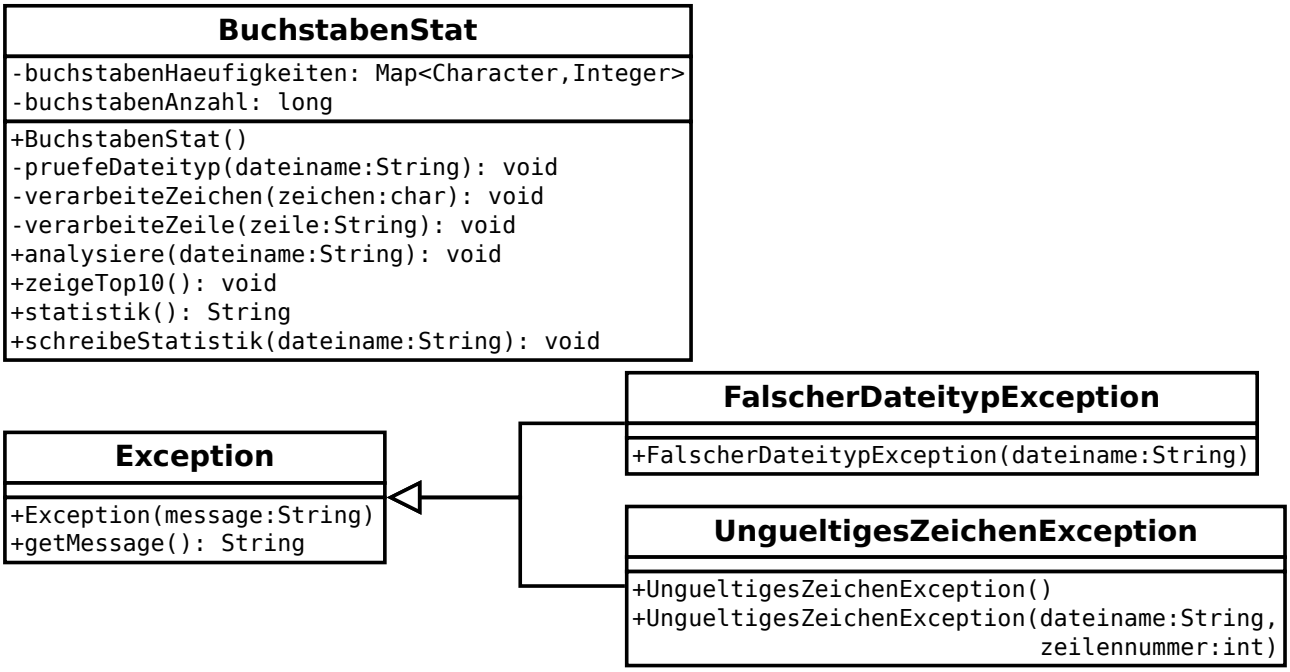
Modellierung und Programmierung 1 – Übungsserie 4

Abgabetermin: 17.01.2021, 22:00 Uhr
Abgabeformat: 1 ZIP-Datei
Max. Punkte: 23 + 3 Zusatzpunkte

Dateien lesen und schreiben, Exceptions

Je nach Sprache treten Buchstaben in unterschiedlicher Häufigkeit auf. Ziel dieser Aufgabe ist ein Programm zur statistischen Auswertung der Buchstabenhäufigkeiten.

Hinweis: Bei der Buchstabenstatistik soll nicht zwischen Groß- und Kleinschreibung unterschiedenen werden. Mittels der Methode `char Character.toUpperCase(char c)` kann ein Kleinbuchstabe in einen Großbuchstaben umgewandelt werden.



1. **Klassen FalscherDateitypException und UngueltigesZeichenException** *(4 Punkte)*

   - *(2 Punkte)* **Klasse FalscherDateitypException**
     Die Ausnahme (Exception) `FalscherDateitypException` wird geworfen, wenn eine Datei mit falscher Dateiendung analysiert werden soll (siehe Methode `pruefeDateityp` der Klasse `BuchstabenStat`). Für eine Datei `Faust.pdf` soll die Message der Ausnahme (Exception) wie folgt lauten:
     `Faust.pdf hat eine nicht kompatible Dateiendung.`

   - *(2 Punkte)* **Klasse UngueltigesZeichenException**
     Die Ausnahme (Exception) `UngueltigesZeichenException` wird geworfen, wenn beim Analysieren einer Datei ein ungültiges Zeichen auftritt (siehe Methode `verarbeite` der Klasse `BuchstabenStat`).

     Für den Standardkonstruktor (Konstruktor ohne Parameter) soll die Message der Exception wie folgt lauten:
     `Ungültiges Zeichen`

Für den zweiten Konstruktor mit den Parametern `dateiname` und `zeilennummer`, soll die Message, bei einer Beispielbelegung der Parameter mit `Faust.txt` und Zeilennumer 200, wie folgt lauten:
`Ungültiges Zeichen in Faust.txt Zeile 200`

2. **Klasse BuchstabenStat** *(17 Punkte)*

   - *(1 Punkt)* Übernehmen Sie die Klasse `BuchstabenStat` wie im Klassendiagramm vorgegeben. Die Attribute der Klasse haben dabei folgende Bedeutung:
     - `buchstabenHaeufigkeiten` speichert die Buchstaben und wie oft sie auftreten.
     - `buchstabenAnzahl` speichert wieviele Buchstaben insgesamt gefunden wurden.
   - *(1 Punkt)* Der Konstruktor initialisiert die Attribute.
   - *(3 Punkte)* `void pruefeDateityp(String dateiname)`
     Die Methode `pruefeDateityp` prüft für den als `String` übergebenen Dateinamen, ob er auf `.txt` endet. Dabei soll Groß-/Kleinschreibung keine Rolle spielen. Sollte der Dateityp nicht passen, wird die Ausnahme (Exception) `FalscherDateitypException` geworfen.
   - *(3 Punkte)* `void verarbeiteZeichen(char zeichen )`
     Die Methode `verarbeiteZeichen` prüft mittels der Methode `boolean Character.isLetter(char c)` ob es sich bei dem übergebenen Zeichen um einen Buchstaben handelt. Wenn dies der Fall ist, wird das Attribut `buchstabenAnzahl` und der Zähler für den entsprechenden Buchstaben im Attribut `buchstabenHaeufigkeiten` um eins erhöht.

     Sollte es sich nicht um einen Buchstaben handeln, dann wird mittels `boolean Character.isDefined(char c)` getestet ob es ein gültiger Unicode ist. Falls der Test fehlschlägt, wird eine `UngueltigesZeichenException` (mit Standardkonstruktor) erstellt und geworfen.
   - *(1 Punkt)* `void verarbeiteZeile(String zeile)`
     Die Methode `verarbeiteZeile` ruft für jedes Zeichen der gegebenen Zeile die Methode `verarbeiteZeichen` auf. Falls die `verarbeiteZeichen`-Methode eine `UngueltigesZeichenException` geworfen hat, wird diese von der `verarbeiteZeile`-Methode ungeprüft weitergereicht.
   - *(4 Punkte)* `void analysiere( String dateiname )`
     Die Methode `analysiere` soll die Buchstabenstatistik für die übergebene Datei erstellen. Sie geht dabei wie folgt vor:
     - Der Dateiname wird mithilfe der Methode `pruefeDateityp` geprüft. Eine geworfene `FalscherDateitypException` wird nach außen weitergereicht.
     - Die Datei wird mithilfe der Klassen `FileReader` und `BufferedReader` geöffnet und Zeile für Zeile eingelesen. Für jede Zeile wird die Methode `verarbeiteZeile` aufgerufen.

       Kann die Datei nicht gefunden werden (FileNotFoundException) oder treten Ein-/Ausgabe-Fehler (IOException) auf, so sollen diese gefangen werden und jeweils eine passende Fehlermeldung mittels `System.err.println` ausgegeben werden. Anschließend wird das Programm mittels `System.exit(1);` beendet.

       Eine Ausnahme `UngueltigesZeichenException` der Methode `verarbeiteZeile` wird gefangen. Anschließend wird eine neue `UngueltigesZeichenException` mit Dateiname und Zeilennummer geworfen.

       Hinweis: Bei Textdateien kann die Zuordnung zwischen Bitfolge in der Datei und dem zu lesenden Zeichen mit unterschiedlichen Codierungen erfolgen. Die mitgegebenen Dateien sind UTF-8 codiert und müssen entsprechend geöffnet werden. Dazu geben Sie die Codierung als Parameter des Konstruktor von `FileReader` an. Sei `dateiname` der String mit dem Namen der zu lesenden Datei, dann erzeugen Sie die `FileReader`-Instanz mit `new FileReader( dateiname, java.nio.charset.StandardCharsets.UTF_8 )` (statt `new FileReader( dateiname )`).

- *(3 Zusatzpunkte)* `void zeigeTop10()`
  Die Methode `zeigeTop10` gibt die 10 häufigsten Buchstaben mit ihrer Anzahl auf der Konsole in folgender Form aus:

  ```
  Buchstaben-Top 10:
  E: 22818
  N: 14065
  I: 11906
  H: 9792
  R: 9764
  S: 9684
  T: 9132
  A: 7902
  D: 7080
  U: 6808
  ```

  Hinweise:

  - Die Methode `entrySet` einer `Map` liefert ein `Set` mit Elementen vom Typ `Map.Entry` (Paaren von Schlüssel und Wert).
  - Der statischen Methode `Collections.sort` kann neben einer Liste, ein `Comparator` zur Ordnung der Listeneinträge als Parameter übergeben werden.
  - Die statische Methode `Map.Entry.comparingByValue()` liefert einen Comparator, welcher `Map.Entry` Instanzen nach dem Wert ordnet.

- *(2 Punkte)* `String statistik()`
  Die Methode `statistik` gibt für jeden Buchstaben der Analyse seinen Anteil an allen Buchstaben aus. Für die Datei `Faust.txt` sieht das wie folgt aus (gekürztes Beispiel):

  ```
  A: 0,05
  B: 0,02
  C: 0,04
  D: 0,05
  E: 0,15
  F: 0,02
  ...
  ```

  Es kann zum Beispiel abgelesen werden, dass der Buchstabe E einen Anteil von 15 Prozent an allen Buchstaben hat.

  Hinweise:

  - Ein Beispiel für ein komplettes Resultat finden Sie in der Datei `FaustStat.txt`
  - Achten Sie darauf die Ausgabe auf zwei Nachkommastellen zu formatieren.

- *(2 Punkte)* `void schreibeStatistik(String dateiname)`
  Die Methode `schreibeStatistik` schreibt die mittels der Methode `statistik` erstellte Tabelle in die Datei mit dem als Parameter übergebenen Dateinamen.

  Tritt ein Ein-/Ausgabe-Ausnahme (IOException) auf, so soll diese gefangen werden und eine passende Fehlermeldung mittels `System.err.println` ausgegeben werden. Anschließend wird das Programm mittels `System.exit(1);` beendet.

3. **Klasse Main** *(2 Punkte)*

- *(2 Punkte)* `static void main(String[] args)`
  In der `main`-Methode soll folgendes passieren.
  - Zunächst wird eine Instanz der Klasse `BuchstabenStat` erzeugt.
  - Anschließend wird die `analysiere`-Methode der erzeugten Instanz mit einer `.txt`-Datei aufgerufen. Sollten die Ausnahmen `FalscherDateitypException` oder `UngueltigesZeichenException` von `analysiere` geworfen werden, so werden diese ausgegeben und die `main`-Methode beendet

- Wenn von Ihnen bearbeitet, wird als nächstes die Methode `zeigeTop10` ausgeführt.
- Letztendlich wird mit der Methode `schreibeStatistik` die Statistik in eine Datei, z.B. `FaustStat.txt`, geschrieben.

Hinweis: Zum Prüfen Ihrer Implementierung finden Sie Dateikombinationen aus Text und Statistik in der `Mitgabe.zip`. Außerdem ist eine Datei `Faust_kaputt.txt` enthalten. Diese Datei enthält ein ungültiges Zeichen und sollte bei Ihren Programm die Ausnahme `UngueltigesZeichenException` auslösen.

- deutsch: `Faust.txt` und `FaustStat.txt`
- deutsch: `Faust_kaputt.txt` enthält ein ungültiges Zeichen
- deutsch: `DeutschlandEinWintermaerchen.txt` und `DeutschlandEinWintermaerchenStat.txt`
- französisch: `lesMiserables.txt` und `lesMiserablesStat.txt`
- englisch: `PrideAndPrejudice.txt` und `PrideAndPrejudiceStat.txt`