Programmieren II: Java

3. Praktikum (Abgabe 7. Juni, 24 Uhr)

Auer, Hanel, Jürgensen, Lehner, Riemenschneider

Lernziele

Aufgabe 1: Password-Checker

Aufgabe 2: World Jumble

Aufgabe 3: Median

Aufgabe 4: Magische Quadrate

Inhalt

Hinweise zum Praktikum

Hinweise zum Praktikum

- Abgabetermin dieses Praktikums: **Abgabe 7. Juni, 24 Uhr**
- Sie dürfen die Aufgaben alleine oder zu zweit abgeben
- Kennzeichnen Sie Ihre Abgabe entsprechend mit Ihren Namen
- Sie müssen 4 der 5 Praktika bestehen
- Kommentieren Sie Ihren Code
  - Jede Methode (wenn nicht vorgegeben)
  - Wichtige Anweisungen/Code-Blöcke
  - Nicht kommentierter Code führt zu Nichtbestehen
- Bestehen Sie eine Abgabe nicht haben Sie einen zweiten Versuch, in dem Sie Ihre Abgabe verbessern müssen
- Wichtig
  - Sie sind einer Praktikumsgruppe zugewiesen, nur in dieser werden Ihre Abgaben akzeptiert
  - Beachten Sie dazu die Hinweise auf der Moodle-Kursseite

Inhalt

Lernziele

Willkommen zum 3. Praktikum!

Lernziele

- Strings: arbeiten mit Strings, Zeichen und `StringBuilder`
- Varargs
- Arrays: erstellen und zugreifen

Inhalt

Aufgabe 1: Password-Checker

Password-Checker

- Schwierigkeitgrad: leicht

Password-Checker

- Schwierigkeitgrad: leicht
- Implementieren Sie in einer Klasse `PasswordChecker` die Methode

```
public static boolean check(String password);
```

mit:

Password-Checker

- Schwierigkeitgrad: leicht
- Implementieren Sie in einer Klasse `PasswordChecker` die Methode

```
public static boolean check(String password);
```

mit:

  - Die Methode soll die Eignung des übergebenen Strings als Passwort prüfen:

Password-Checker

- Schwierigkeitgrad: leicht
- Implementieren Sie in einer Klasse `PasswordChecker` die Methode

```
public static boolean check(String password);
```

mit:

- Die Methode soll die Eignung des übergebenen Strings als Passwort prüfen:
  - Mindestens acht Zeichen

Password-Checker

- Schwierigkeitgrad: leicht
- Implementieren Sie in einer Klasse `PasswordChecker` die Methode

```
public static boolean check(String password);
```

mit:

- Die Methode soll die Eignung des übergebenen Strings als Passwort prüfen:
  - Mindestens acht Zeichen
  - Erlaubte Zeichen: Groß- und Kleinbuchstaben (ohne Umlaute), Ziffern und Sonderzeichen (# . ? !)

Password-Checker

- Schwierigkeitgrad: leicht
- Implementieren Sie in einer Klasse `PasswordChecker` die Methode

```
public static boolean check(String password);
```

mit:

- Die Methode soll die Eignung des übergebenen Strings als Passwort prüfen:
  - Mindestens acht Zeichen
  - Erlaubte Zeichen: Groß- und Kleinbuchstaben (ohne Umlaute), Ziffern und Sonderzeichen (# . ? !)
  - Jeweils mindestens ein/e Großbuchstabe, Kleinbuchstabe, Ziffer, Sonderzeichen

Password-Checker

- Schwierigkeitgrad: leicht
- Implementieren Sie in einer Klasse `PasswordChecker` die Methode

```
public static boolean check(String password);
```

mit:

- Die Methode soll die Eignung des übergebenen Strings als Passwort prüfen:
  - Mindestens acht Zeichen
  - Erlaubte Zeichen: Groß- und Kleinbuchstaben (ohne Umlaute), Ziffern und Sonderzeichen (# . ? !)
  - Jeweils mindestens ein/e Großbuchstabe, Kleinbuchstabe, Ziffer, Sonderzeichen
- Sind die Bedingungen erfüllt gibt die Methode **`true`** zurück, sonst **`false`**

Password-Checker

- Schwierigkeitgrad: leicht
- Implementieren Sie in einer Klasse `PasswordChecker` die Methode

```
public static boolean check(String password);
```

mit:

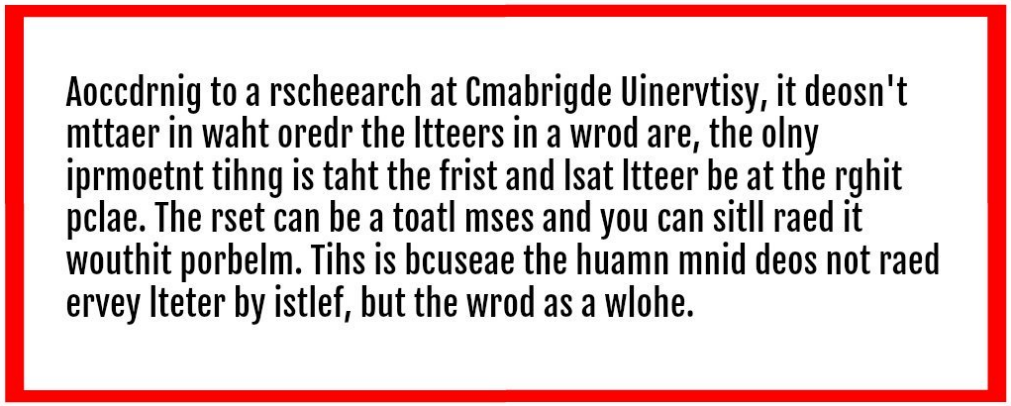
  - Die Methode soll die Eignung des übergebenen Strings als Passwort prüfen:
    - Mindestens acht Zeichen
    - Erlaubte Zeichen: Groß- und Kleinbuchstaben (ohne Umlaute), Ziffern und Sonderzeichen (# . ? !)
    - Jeweils mindestens ein/e Großbuchstabe, Kleinbuchstabe, Ziffer, Sonderzeichen
  - Sind die Bedingungen erfüllt gibt die Methode **`true`** zurück, sonst **`false`**
  - Sie können zum Testen Ihrer Implementierungen die JUnit-Tests in password-checker/PasswordCheckerTest.java verwenden und/oder ein eigenes kleines Java-Programm implementieren

Inhalt

Aufgabe 2: World Jumble

Word Jumble

- Vor ein paar Jahren geisterte folgendes „Science-Meme“ durch das Netz



- „Übersetzt“: Ncah enier Sutide an der Cmarbdige Uinevsrtiät, ist die Rieehfnloge der Bcushatebn nciht wcithig snoedrn nur dsas der esrte und ltetze Bcushatbe an der rcithgien Setlle snid. Der Rset knan ein ttolaes Druhcieandner sien and man knan es immer ncoh onhe Porlbmee lseen. Der Gurnd ist dsas das Gherin nciht enieznle Bcushatebn efrsast, snoedrn gnaze Wröetr.

Aufgabenstellung

- Schwierigkeitsgrad: mittel
- Wir wollen eine Java-Anwendung schreiben, die einen Text einliest und die Buchstaben der Worte wie oben gesehen vertauscht
- Sie können mit dem Grundgerüst aus der Datei word-jumble/WordJumble.java starten
  - `void jumbleWord(String word, StringBuilder builder)` vertauscht Buchstaben des Wortes `word` und hängt das Ergebnis an `StringBuilder` an (wird von Ihnen implementiert)
  - `String jumbleText(Scanner scanner)` liest Wort für Wort aus `scanner` und übergibt es an `jumbleWord` (bereits vollständig)
  - `main` ruft `jumbleText` auf der Standardeingabe auf (bereits vollständig)
- Hinweis: Wir gehen davon aus, dass der zu transformierende Text keine Satzzeichen enthält

Aufgabenstellung

- `WordJumble` ist bereits lauffähig, übersetzen Sie es.
- Das Programm liest von der Standardeingabe und gibt das Ergebnis auf die Standardausgabe aus
- Führen Sie das Programm aus und geben Sie einen Text ein. Sie können die Eingabe mit Ctrl-Z (Windows) oder Ctrl-D (Linux/macOS) in einer neuen Zeile beenden.

```
java WordJumble
Das ist ein Test
<Ctrl-D> oder <Ctrl-Z>
Das ist ein Test
```

- Alternativ können Sie über < den Inhalt einer Datei als Eingabe verwenden

```
java WordJumble < WordJumbleTest.txt
Nach einer Studie...
```

Sie können dazu die Datei word-jumble/WordJumbleTest.txt verwenden

Aufgabenstellung

- Die aktuelle Implementierung von `jumbleWord` hängt das Wort unverändert an den `StringBuilder` an
- Implementieren Sie `jumbleWord` nach folgenden Regeln
  - Worte der Länge bis drei bleiben unverändert
  - Worte der Länge vier oder länger werden wie folgt bearbeitet
    - Erster und letzter Buchstabe bleiben wie sie sind
    - Alle Buchstaben innerhalb des Wortes werden von links nach rechts paarweise vertauscht
    - Beispiel: T<u>es</u>t → Tset
    - Noch ein Beispiel: P<u>ro</u>grammieren → Por<u>gr</u>ammieren → Porr<u>gam</u>mieren → Porrgma<u>mi</u>eren → Porrgmaim<u>er</u>en → Porrgmaimreen
- Hinweise:
  - Was passiert mit dem vorletzten Buchstaben bei Worten ungerader Länge größer als drei?
  - Sehen Sie sich die Dokumentationen von String und StringBuilder an. Wie greift man auf einzelne Zeichen eines `String`s zu? Wie hängt man Elemente an `StringBuilder` an?
- Zum Testen können Sie word-jumble/WordJumbleTest.java verwenden

Inhalt

Aufgabe 3: Median

Median

- Schwierigkeitsgrad: mittel
- Den Median einer Auflistung von **`int`**s berechnet man wie folgt
  1. Sortiere die Auflistung
  2. Hat die Auflistung ungerade viele Einträge, gib das mittlere Element zurück
  3. Hat die Auflistung gerade viele Einträge, gib das Mittel der beiden mittleren Elemente zurück
- Beispiel
  - Eingabe: `[4,7,1,9,2]`
  - Sortiert: `[1,2,4,7,9]`
  - Es sind ungerade viele Elemente, also ist der Median das mittlere Element 4
- Noch ein Beispiel
  - Eingabe: `[4,7,1,9,2,5]`
  - Sortiert: `[1,2,4,5,7,9]`
  - Es sind gerade viele Elemente, also ist der Median das Mittel der beiden mittleren Element 4 und 5 und damit $(4+5)/2 = 4.5$

Aufgabenstellung

- Implementieren Sie in einer Klasse `Median` die Methode `computeMedian`
  - Die Methode ist statisch
  - Sie akzeptiert als Varargs eine Auflistung von **`int`**s
  - Sie gibt den Median der Auflistung als **`double`** zurück
  - Sollte `computeMedian` mit ungültigen Parametern aufgerufen werden, so generieren Sie eine `IllegalArgumentException` mit entsprechender Nachricht
- Hinweise
  - Ein beispielhafter Aufruf der Methoden sieht wie folgt aus:

```
Median.computeMedian(4,7,1,9,2); // liefert 4
```

  - Oder:

```
Median.computeMedian(4,7,1,9,2,5); // liefert 4.5
```

  - Sie können zum Sortieren die Methoden `Arrays.sort` verwenden
  - Verwenden Sie zum Testen Ihrer Implementierung die JUnit-Tests in median/MedianTest.java

Inhalt

Aufgabe 4: Magische Quadrate

- Schwierigkeitsgrad: mittel bis schwer
- In einem `magischen Quadrat` sind die Summen aller Zeilen, Spalten und Diagonalen gleich
- Beispiel (Summe jeweils 15)

8	1	6
3	5	7
4	9	2

- Die Anzahl der Zeilen und Spalten nennt man Größe des magischen Quadrats
- In dem Stich „Melencolia I“ von Albrecht Dürer (s. rechts) ist rechts oben ein magisches Quadrat der Größe vier zu sehen

Aufgabenstellung

- Definieren Sie eine Klasse `MagicSquare`
- Implementieren Sie eine statische Methode `int[][] getExample()`, die das Beispiel der vorherigen Folie als zweidimensionalen Array zurückgibt
  - Definieren den Array als Array-Literal
  - Der erste Index des Arrays entspricht der `Zeile`, der zweite Index der Spalte
- Implementieren Sie die statische Methode `isMagic`
  - `isMagic` bekommt als Parameter einen zweidimensionalen **`int`**-Array und gibt **`boolean`** zurück
  - `isMagic` gibt **`true`** zurück wenn es sich beim zweidimensionalen Array um ein magisches Quadrat handelt
  - Bei ungültigem Parameter generieren Sie eine `IllegalArgumentException` mit entsprechender Nachricht:
    - **`null`** ist nicht erlaubt
    - zweidimensionale Arrays mit ungleicher Spalten- und Zeilenzahl sind nicht erlaubt
- Verwenden zum Testen die JUnit-Tests in

  magic/MagicSquareTest.java

Bonus-Aufgabe

- Diese Aufgabe ist optional (aber eine tolle Übung!)
- Schwierigkeitsgrad: schwer
- Unter `https://en.wikipedia.org/wiki/Siamese_method` finden Sie eine Methode um magische Quadrate ungerader Größe zu generieren
- Aufgaben
  - Implementieren Sie dieses Verfahren
  - Testen Sie die Korrektheit mit Hilfe Ihrer Methode `isMagic`
  - Implementieren Sie eine Methode, die ein magisches Quadrat auf der Konsole ausgibt
  - Schreiben Sie ein Hauptprogramm, das über die Kommandozeile die Größe einliest und das magische Quadrat dazu ausgibt
- Die Ausgabe könnte bspw. so aussehen:

```
java MagicSquare 3
8    1    6
3    5    7
4    9    2
```