Programmierung 2

Aufgabenzettel 3

Dieser Aufgabenzettel muss bis zum **28.04, 12:00 Uhr**, elektronisch in Moodle als Gruppe von 2 Studierenden abgegeben werden. Spätere Abgaben sind nicht möglich.

Die Abgabe erfolgt als ZIP-Datei, welche die folgenden Konventionen einhalten **muss**:

- Dateiname: p2-assignment-3-group-*Gruppennummer*.zip
  (Beispiel: Gruppe 15 gibt eine Datei mit dem Namen p2-assignment-3-group-15.zip ab)
- Die folgenden Dateien müssen sich in der Ablage befinden (Ordner beachten!):
  - `src/de/hrw/progra2/assignment3/MiddleCharacter.java`
  - `src/de/hrw/progra2/assignment3/TemperatureConverter.java`
  - `src/de/hrw/progra2/assignment3/PasswordCharacteristicValidator.java`
- Alle Java-Dateien **müssen in UTF-8 kodiert sein und** sich im Ordner `src/de/hrw/progra2/assignment3/` befinden und kompilieren.
- Die ZIP-Datei darf keine Dateien mit Endung `.class` oder `.jar` enthalten.

Abgaben, die diese Konventionen nicht einhalten, werden mit 0 Punkten bewertet.

Tipp: Benutzen Sie das folgende Online-Tool, um Ihre Datei vor der Abgabe formal zu prüfen:

https://codingprof.hs-rw.de/csf/?module=p2&assignment=3

Aufgabe 1: Methoden überladen (3 Punkte)

Die Methode `middleCharacter` bestimmt den mittleren Buchstaben eines Wortes. Wenn die Länge des Wortes gerade ist, gibt es zwei mittlere Buchstaben.

a) Implementieren Sie die erste Variante der Methode `middleCharacter`. Sie erhält als Übergabeparameter einen String vom Typ `String`. Als Rückgabewert gibt die Funktion die oder den Buchstabe/n als `String` zurück.
   Testen Sie die Methode in einem Hauptprogramm.

   (1.5 Punkte)

b) Durch Überladung erhält die zweite Variante der Methode `middleCharacter` ein `Character-Array` als Übergabeparameter.
   Testen Sie die Methode in einem Hauptprogramm.

   (1.5 Punkte)

Hinweis zur Abgabe:

Der Quelltext für die Lösung der Aufgabe muss in einer Datei `MiddleCharacter.java` gespeichert sein.

Aufgabe 2: Kommandozeilenparameter (4 Punkte)

Schreiben Sie das folgende Programm, welches über die Kommandozeile gesteuert werden kann.

Das Programm `TemperatureConverter` rechnet eine Temperatur von Grad Celsius entweder in Kelvin oder in Fahrenheit um und gibt diese auf dem Bildschirm aus.

Die Methode erhält die folgenden Kommandozeilenparameter (in dieser Reihenfolge):

- Ein **optionaler** Parameter --useInteger. Ist der optionale Parameter gesetzt, soll das Ergebnis als Ganzzahl ausgegeben werden.
- Eine Auswahl, in welche Einheit umgerechnet werden soll (0: Kelvin, 1: Fahrenheit).
- Eine Temperatur in Grad Celsius (Fließkommazahl).

Beispielhafte Aufrufe:

java TemperatureConverter 0 22.5
ergibt
295.65 Kelvin

java TemperatureConverter --useInteger 1 18.37
ergibt
65 F

java TemperatureConverter
ergibt
"Sie müssen mindestens eine Auswahl und eine Temperatur angeben."

Hinweise:

- Recherchieren Sie die Formeln für die Umrechnungen selbstständig.
- Auf fehlende nicht-optionale Übergabeparameter muss durch eine Fehlermeldung reagiert werden.

Hinweis zur Abgabe:

Der Quelltext für die Lösung der Aufgabe muss in einer Datei `TemperatureConverter.java` gespeichert sein.

Aufgabe 3: Methoden (4 Punkte)

In dieser Aufgabe schreiben Sie Methoden, um die Sicherheitscharakteristiken eines Passworts zu überprüfen.

Die Methode `isStrongPassword` erhält als Übergabeparameter ein Passwort als `String` und gibt `true` zurück, wenn das Passwort sicher ist und `false`, wenn das Passwort unsicher ist.

Um ein Passwort auf Sicherheit zu prüfen, ruft isStrongPassword weitere Methoden auf, welche folgende 5 Charakteristiken prüfen:

- Das Passwort hat mindestens eine Länge von 12 Buchstaben oder Zahlen.
- Das Passwort enthält Groß- und Kleinbuchstaben.
- Das Passwort ist eine Mixtur aus Buchstaben und Zahlen.
- Das Passwort enthält mindestens einmal eines der folgenden Sonderzeichen: ! @ # ?
- Das Passwort enthält keines der beiden folgenden Sonderzeichen: < >

Die 5 Methoden zur Prüfung der Passwort-Charakteristiken geben jeweils `true` zurück, wenn die jeweilige Charakteristik erfüllt ist.

Implementieren Sie alle 6 Methoden und testen Sie die Funktionalität in einem Hauptprogramm.

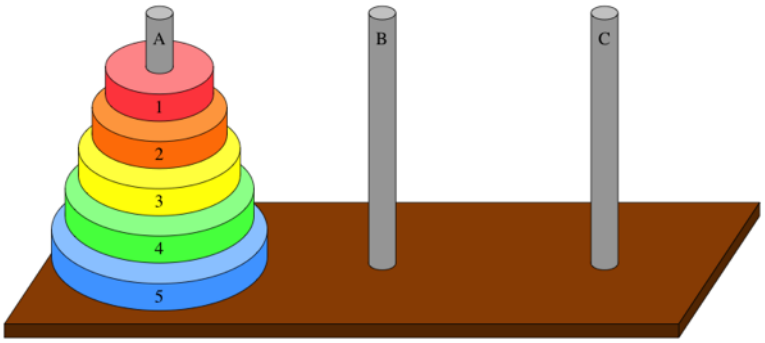
Hinweis zur Abgabe:

Der Quelltext für die Lösung der Aufgabe muss in einer Datei `PasswordCharacteristicValidator.java` gespeichert sein.

Freiwillige Zusatzaufgabe: Türme von Hanoi (*keine* Punkte)

Die im Folgenden dargestellte Aufgabe ist *freiwillig*. Dies bedeutet, dass es *keine* Punkte für die Lösung dieser Aufgabe gibt. Eine Bearbeitung (und auch nur der Versuch) lohnt sich dennoch.

Das Knobelspiel *Türme von Hanoi* besteht aus drei Stäben a, b und c. Auf den Ausgangsstab a werden n Scheiben gelegt. Die Scheiben unterscheiden sich in ihrer Größe. In der Ausgangsposition sind die auf a liegenden Scheiben von der größten Scheibe ganz unten hin zur kleinsten Scheibe ganz oben angeordnet.
Ziel des Spiels ist es, alle Scheiben von Stab a auf Stab c zu bewegen.
Als *Regeln* darf bei jedem Zug nur die oberste Scheibe eines Stapels bewegt werden.
Dies ist nur erlaubt, sofern sich am neuen Stab keine kleinere Scheibe befindet.
Daraus folgt, dass zu jedem Spielzug auf den Stäben a, b und c immer alle Scheiben von groß (unten) nach klein (oben) gestapelt sind.



Türme von Hanoi mit 3 Stäben A, B, C und n = 5 in der Ausgangsposition. Bildquelle

Ihre Aufgabe ist es, einen rekursiven (sich selbst aufrufenden) Algorithmus in der Methode `calculateHanoiTowers.java` zu implementieren, der das Spiel mit n = 3 Scheiben und 3 Stäben (a, b, c) simuliert.
Tipps:

- Lösungen für diese Aufgabe gibt es zahlreich im Internet. Versuchen Sie deshalb, nicht nach konkreten Implementierungen zu schauen, sondern selbst auf die Lösung zu kommen.
- Lesen Sie ggfs. die beiden verlinkten Artikel zum Verständnis.
- Sie können auch vorerst ohne Rekursion – sondern mit Iteration – arbeiten.
- Artikel zu Türmen von Hanoi: Wikipedia

- Betrachten Sie den Pseudocode vom Artikel zu den Türmen von Hanoi auf Wikipedia (wobei hier als n der Buchstabe i verwendet wurde):

```
funktion bewege (Zahl i, Stab a, Stab b, Stab c) {
    falls (i > 0) {
       bewege(i-1, a, c, b);
       verschiebe oberste Scheibe von a nach c;
       bewege(i-1, b, a, c);
    }
}
```