Projektblatt P12 (17 P)

Abgabe: Freitag 8. Dezember 2023, 12:00h
Entpacken Sie zunächst die Archiv-Datei `vorgaben-p12.zip`, in der sich die Rahmendateien für die zu lösenden Aufgaben befinden. Ergänzen Sie die Dateien durch Ihre Lösungen gemäß der Aufgabenstellung unten. Der hinzuzufügende Java-Sourcecode sollte **syntaktisch richtig** und **vollständig formatiert** sein. Alle Dateien sollten am Ende fehlerfrei übersetzt werden können.
Verpacken Sie die `.java` Dateien für Ihre Abgabe in einem ZIP-Archiv mit dem Namen `IhrNachname.IhrVorname.P12.zip`, welches Sie auf Ilias hochladen.

Aufgabe 1: Reguläre Ausdrücke 9 Punkte

In dieser Aufgabe sollen Sie mehrere Klassenmethoden implementieren, welche jeweils eine Zeichenkette auf eine bestimmte Eigenschaft hin untersuchen. In der Datei `Regex.java` ist hierzu eine Klasse `Regex` definiert, welche sechs Methoden `check1` - `check6` definiert, die jeweils den Default-Wert `false` zurückgeben. Darüberhinaus definiert die `main`-Methode für jede dieser Methoden mehrere Teststrings, für die die Methode aufgerufen und die Ergebnisse dann ausgegeben werden.
Implementieren Sie nun die sechs Klassenmethoden `check1` - `check6`, die jeweils einen `String` Parameter besitzen und einen boole'schen Wert zurückgeben, der angibt, ob der String eine bestimmte Eigenschaft besitzt. Ist dies der Fall, soll die Methode den Wert `true` zurückgeben, ansonsten den Wert `false`; Verwenden Sie dazu jeweils die Methode `matches` der Klasse `String` und einen passenden **Regulären Ausdruck**. In einigen Fällen benötigen Sie evtl. auch mehrere Aufrufe der `matches` Methode mit verschiedenen Regulären Ausdrücken, die Sie dann durch logische Operatoren geeignet kombinieren müssen.

`check1` Diese Methode soll überprüfen, ob der als Parameter übergebene String eine Oktalzahl darstellt (in Java-Notation).

`check2` Diese Methode soll überprüfen, ob der als Parameter übergebene String keine Whitespace-Zeichen enthält (also keine Leerzeichen, Tabulatorzeichen, Zeilenumbrüche etc.)

`check3` Diese Methode soll überprüfen, ob der als Parameter übergebene String ausschließlich aus Buchstaben und Ziffern besteht und dazu je mindestens einen Großbuchstaben, einen Kleinbuchstaben und eine Ziffer enthält.

`check4` Diese Methode soll überprüfen, ob der als Parameter übergebene String eine ganze Zahl aus dem Intervall $[-100, 100]$ repräsentiert. Führende Nullen sind dabei nicht erlaubt.

`check5` Diese Methode soll überprüfen, ob der als Parameter übergebene String mindestens einen Klein- oder Großbuchstaben enthält, der mindestens dreimal in dem String enthalten ist (immer als Klein- bzw. Großbuchstabe).

`check6` Diese Methode soll überprüfen, ob der als Parameter übergebene String aus genau vier Ziffern besteht. Darüber hinaus soll der String nur aus zwei verschiedenen Ziffern bestehen, die jeweils zweimal vorkommen.

Die `main`-Methode der Klasse `RegEx` beinhaltet mehrere Felder von Strings, mit denen die Methoden `check1` - `check6` getestet werden.
Ein Programmlauf sollte dann folgende Ausgabe produzieren:

```
Test 1:
Die Zeichenkette repräsentiert eine Oktalzahl in
Java-Notation
"01" -> true
"020" -> true
"0" -> false
"00" -> true
"100" -> false
"088" -> false
"019" -> false

Test 2:
Die Zeichenkette enthält keine Whitespace-Zeichen.
"abababa" -> true
"ab ab" -> false
" ab" -> false
"ab     ab" -> false
"ab
" -> false

Test 3:
Die Zeichenkette besteht ausschließlich aus Buchstaben
und Ziffern, enthält dabei aber je mindestens einen
Kleinbuchstaben, Großbuchstaben sowie eine Ziffer
"aB33" -> true
"1234Xy" -> true
"0XxxxX0" -> true
"a999z" -> false
"AB33" -> false
```

```
Test 4:
Die Zeichenkette repräsentiert eine ganze Zahl aus
dem Intervall [-100,100]
"-100" -> true
"-99" -> true
"42" -> true
"+12" -> true
"0" -> true
"09" -> false
"101" -> false

Test 5:
Die Zeichenkette enthält mindestens einen Buchstaben
genau dreimal (Klein- und Großbuchstaben werden
dabei als unterschiedliche Zeichen gewertet).
"axbcxdexfgh" -> true
"aabbcca" -> true
"hiiih" -> true
"zzz" -> true
"PspSPs" -> false

Test 6:
Die Zeichenkette besteht aus genau vier Ziffern.
Dabei wird die Zeichenkette aus zwei verschiedenen
Ziffern gebildet, die jeweils zweimal vorkommen.
"1212" -> true
"5566" -> true
"7447" -> true
"9999" -> false
"313" -> false
"1747" -> false
```

Aufgabe 2: Enums, Listener — 8 Punkte

In dieser Aufgabe sollen Sie einen einfachen Aufzählungstyp für Monate implementieren und testen:

(a) Definieren Sie in einer neuen Datei einen Aufzählungstyp `Monat` und statten Sie ihn wie folgt aus:

- Definieren Sie die Konstanten `JANUAR` bis `DEZEMBER`
- Legen Sie ein Attribut für die Anzahl der Tage in einem Monat an, erweitern Sie die Definition der Konstanten entsprechend (Default-Wert für die Anzahl der Tage im FEBRUAR ist hierbei 28) und schreiben Sie einen passenden Konstruktor.
- Schreiben Sie eine Getter-Methode für die Anzahl der Tage, die einen Parameter mit einem (ganzzahligen) Wert für das Jahr besitzt. Wenn die Methode für das Objekt `FEBRUAR` und ein Schaltjahr aufgerufen wird, geben Sie den Wert 29 zurück, ansonsten den Default-Wert für das jeweilige Objekt.

  Hinweis: Verwenden Sie eine passende statische Methode der Klasse `java.time.Year`, um herauszufinden, ob das als Parameter übergebene Jahr ein Schaltjahr ist.
- Überladen Sie die Getter-Methode mit einer Methode ohne Parameter, die die zuvor implementierte Methode mit dem aktuellen Jahreswert aufruft.

  Hinweis:Um diesen zu bestimmen, können Sie ebenfalls eine statische Methode der Klasse `java.time.Year` verwenden.
- Überschreiben Sie die Methode `toString`, so dass für jede Monats-Konstante (Objekt) ein String mit der "normalen" Schreibweise (z.B. `Januar` für `JANUAR`) zurückgegeben wird.

(b) Die Klasse `Aufgabe_2` gibt eine einfache GUI vor, die im oberen Teil ein Textfeld enthält, über dass Sie eine (gültige) Jahreszahl eingeben sollen, für die im unteren Teil enthaltenen Textbereich dann die Monate mit der jeweiligen Anzahl von Tagen angezeigt werden sollen.

Ergänzen Sie die Klasse durch eine innere Klasse, die einen Listener darstellt, der auf das Bestätigen einer Eingabe (durch Drücken der <Return> Taste) in einem Textfeld reagiert.

Beim Betätigen der Eingabetaste soll der Text aus dem Textfeld `jahre` (Attribut der Klasse `Aufgabe_2`) gelesen und die Leerzeichen auf beiden Seiten des String abgeschnitten werden. Wandeln Sie den String dann in einen `int`-Wert um. Gehen Sie dabei davon aus, dass der User einen String angegeben hat, der eine (korrekte) Jahreszahl repräsentiert.

Generieren Sie dann eine `StringBuffer` Instanz, die Sie mit einem Text füllen, welche als Überschrift die Jahreszahl und für alle in dem Aufzählungstyp `Monat` definierten Monate die jeweilige Anzahl der Tage angibt (siehe Screenshot unten für ein Beispiel). Bestimmen Sie die dort definierten Monate dynamisch mit Hilfe der Methode `values`.

Ersetzen Sie den aktuellen String in dem Textbereich `ausgabe` durch den Inhalt des `StringBuffer` Objekts.

Registrieren Sie eine Instanz der Klasse bei dem Textfeld `jahre` in der Methode `createListener`.

Die folgenden Screenshots zeigen die GUI bei Aufruf der Klasse `Aufgabe_2` und nach Eingabe von "2024" (plus Bestätigung durch die <Return> Taste).