Übungsblatt 7: Strings, Klassen aus der API, Reguläre Ausdrücke

Aufgabe 1: Verarbeitung von Zeichenketten

Schreiben Sie eine Klasse `StringWorker` mit einigen nützlichen Klassen-Methoden zur Verarbeitung von Strings. Benutzen Sie hierzu geeignete Methoden der Klassen `String` (z.B. die Methoden `charAt`, `indexOf`, `replace`) und `StringBuffer` (z.B. die Methode `append`).

- eine Methode, die eine Zeichenkette der Länge n aus zufällig gewählten Buchstaben erzeugt.
- eine Methode, die eine als Parameter übergebenen Zeichenkette in umgekehrter Reihenfolge zurückgibt.
  Beispiel: `reverse(‘‘abcdef‘‘)` → `‘‘fedcba‘‘`
- eine Methode, die überprüft, ob eine als Parameter übergebenen Zeichenkette ein Palindrom darstellt (d.h. von hinten und vorne gelesen identisch ist).
  Beispiel: `palindromP(‘‘abcdcba‘‘)` → `true`
- eine Methode, die einen Teil einer als Parameter übergebenen Zeichenkette umkehrt. Der umzukehrende Teil soll dabei durch einen Index i für die erste Position des Teilstrings in der Zeichenkette und einen Index j für die erste Position in der Zeichenkette, die nicht mehr Bestandteil des Teilsstrings ist, spezifiziert werden.
  Beispiel: `reverseSubstring(‘‘aaaxyzbbb‘‘,3,6)` → `‘‘aaazyxbbb‘‘`
- eine Methode, die in einer als Parameter übergebenen Zeichenkette ein einzelnes Zeichen an einer Position k durch ein anderes Zeichen (alle Werte als Parameter übergeben) ersetzt.
  Beispiel: `setChar(‘‘aaaXaaa‘‘,3,’Y’)` → `‘‘aaaYaaa‘‘`
- eine Methode, die in einer als Parameter übergebenen Zeichenkette einen Teil der Zeichenkette ab einer bestimmten Position durch eine weitere Zeichenkette (beide Werte ebenfalls als Parameter übergeben) ersetzt. Dies soll nur dann gemacht werden, wenn die ursprüngliche Zeichenkette hierfür lang genug ist.
  Beispiel: `setSubstring(‘‘aaaxyzbbb‘‘,3,‘‘ccc‘‘)` → `‘‘aaacccbbb‘‘`

- Eine Methode, die die Anzahl der Vorkommen eines Strings in einem anderen zählt. Hierbei sollen überlappende Vorkommen entsprechend mehrfach gezählt werden.
  Beispiel: `countPositions1(‘‘aaabbbaaa‘‘,‘‘aa‘‘)` $\rightarrow$ `4`

- Eine Methode, die die Anzahl der Vorkommen eines Strings in einem anderen zählt. Hierbei sollen überlappende Vorkommen nur einfach gezählt werden.
  Beispiel: `countPositions1(‘‘aaabbbaaa‘‘,‘‘aa‘‘)` $\rightarrow$ `2`

- eine Methode, die die zwei Zeichenketten zeichenweise mischt. Ist eine der beiden Zeichenketten länger als die andere, soll der Überhang hintendran gehängt werden.
  Beispiel: `mixStrings(‘‘abc‘‘, ‘‘xyz00‘‘)` $\rightarrow$ `‘‘axbycz00‘‘`

Ergänzen Sie die Klasse durch eine `main`-Methode mit Aufrufen der von Ihnen geschriebenen Methoden um diese zu testen.

Aufgabe 2: Hüllklassen, Kommandozeilenargumente

Schreiben Sie ein Programm, welches alle Argumente, die von der Kommandozeile übergeben werden, als `double`-Zahlen interpretiert, diese zunächst in zwei verschiedenen Feldern speichert und dann die Summe der Zahlen in den beiden Feldern berechnet. Erweitern Sie hierzu die Klasse `Aufgabe1`, in der bereits zwei Instanzattribute vorgegeben sind: je ein Feld mit Elementen vom Typ `double` bzw. `Double`. Entwickeln Sie dann

- eine Methode `initZahlen`, die die beiden Felder mit einer Größe von n Elementen initialisiert. Der Wert von n soll dabei als Parameter an die Methode übergeben werden.

- eine Methode `setZahl`, die einen `String`-Parameter in einen `double` sowie einen `Double` Wert umwandelt und die Resultate in den Feldern `zahlen` bzw. `zahlObjekte` am Index i speichert. Der umzuwandelnde `String` sowie der Indexwert i sollen dabei als Parameter an die Methode übergeben werden.

- eine Methode `add1`, die die Werte im Feld `zahlen` addiert und die Summe als Ergebnis zurückgibt.

- eine Methode `add2`, die die Werte im Feld `zahlObjekte` addiert und die Summe als Ergebnis zurückgibt. Achten Sie hierbei darauf, dass Sie kein Auto-(Un-)Boxing verwenden.

Ergänzen Sie nun die `main`-Methode so, dass alle Argumente, die an diese Methode (über den Aufruf von der Kommandozeile) übergeben werden, als Gleitkomma-Zahlen interpretiert und in den (geeignet initialisierten) Instanzvariablen einer Instanz der Klasse `Aufgabe1` gespeichert werden. Geben Sie die Summe dieser Zahlen als Ergebnis der Methoden `add1` und `add2` aus.

Dann sollte das Ergebnis des Aufrufs

```
java Aufgabe1 12 13.0 1.5e2
```

wie folgt aussehen:

```
Summe 1: 175.0
Summe 2: 175.0
```

Aufgabe 3: Terminverwaltung, Calendar-Klasse

In der Klasse `TerminverwaltungJ8` ist Folgendes vorgegeben:

- eine private Instanzvariable `termine`, die ein Feld vom Typ `LocalDateTime` beinhaltet
- eine Klassenmethode `randomMinMax`, die eine ganzzahlige Zufallszahl in einem Bereich $[a, b]$ generiert und zurückgibt
- Eine Klassenmethode `createRandomTermin`, die einen zufälligen Termin im aktuellen Jahr generiert und zurückgibt. (in Schaltjahren allerdings nicht den 31.12.) Die Uhrzeit für den Termin liegt dabei zwischen 9 und 16 Uhr und beginnt jeweils zu einer vollen Viertelstunde)
- ein Konstruktor, der die Instanzvariable `termine` mit einem Feld der Länge n initialisiert und mit Zufallsterminen füllt.
- eine Instanzmethode `getKW`, die einen `int`-Wert für die Kalenderwoche eines Datums vpm Typ LocalDateberechnet und zurückgibt.

Vervollständigen Sie nun die Klasse `TerminverwaltungJ8` wie folgt:

- Implementieren Sie zunächst eine Klassenmethode `createTermin` mit vier Parametern für den Monat, den Tag, die Stunde und die Minute (jeweils als int-Wert) und generieren mit deren Hilfe eine Kalenderinstanz (Klasse `LocalDateTime`) für das aktuelle Jahr.

- Schreiben Sie eine Methode `formatTermin`, die für ein als Parameter übergebenes Datum vom Typ `LocalDate` mit Hilfe der `DateTimeFormatter`-Klasse einen formatierten String mit den wichtigsten Informationen zu dem Termin erzeugt und zurückgibt.

- Schreiben Sie eine Methode, die in der zu dieser Instanz gehörigen Terminliste alle Termine sucht, die in einer bestimmten Kalenderwoche liegen, und diese formatiert als String zurückgibt. Die Kalenderwoche soll dabei als `int`-Wert an die Methode übergeben werden.

- Implementieren Sie eine Methode, die alle Termine aus der zu dieser Instanz gehörigen Terminliste sammelt, die innerhalb eines Zeitraums zwischen zwei, der Methode als Parameter übergebenen Zeitpunkten liegt und, als `String` formatiert, zurückgibt.

- Fügen Sie nun noch eine Methode hinzu, die alle Termine in den vom heutigen Datum an gerechneten nächsten k Tagen ausfiltert und diese in Form eines formatierten Strings zurückgibt.

- Ergänzen Sie die `main`-Methode der Klasse um Testaufrufe für die von Ihnen implementierten Methoden. Geben Sie zunächst alle Termine aus dem Feld `termine` aus, danach z.B. alle Termine in der Kalenderwoche 12, anschließend alle Termine zwischen dem 1. Mai und dem 5. Juni und schließlich noch alle Termine in den nächsten sieben Tagen.

  Eine Ausgabe des Programms könnte dann beispielsweise wie folgt aussehen:

```
>>>>>>>>Alle Termine:
17. Mai, 13:15, (KW 20)
05. März, 15:30, (KW 10)
10. Februar, 14:30, (KW 6)
10. September, 12:15, (KW 37)
02. Juni, 10:00, (KW 22)
12. August, 10:30, (KW 33)
```

```
12. November, 11:00, (KW 46)
07. Juli, 10:45, (KW 27)
23. November, 16:45, (KW 47)
23. August, 09:45, (KW 34)
11. Dezember, 10:15, (KW 50)
06. Juli, 09:00, (KW 27)
14. Mai, 16:15, (KW 20)
23. April, 15:45, (KW 17)
13. Januar, 14:30, (KW 2)
29. Oktober, 11:15, (KW 44)
20. Juli, 16:15, (KW 29)
28. Dezember, 16:00, (KW 52)
08. Januar, 14:45, (KW 2)
19. Mai, 16:15, (KW 20)
29. Mai, 14:45, (KW 22)
20. März, 16:30, (KW 12)
10. September, 09:00, (KW 37)
23. Januar, 09:45, (KW 4)
01. Dezember, 10:45, (KW 48)

>>>>>>>>Termine in KW 12:
Alle Termine in KW12:
20. März, 16:30, (KW 12)


>>>>>>>>Termine zwischen 12.5 und 5.6.:
Alle Termine im Zeitraum vom 12. Mai, 00:00, (KW 19) bis 05. Juni, 00:00, (KW
17. Mai, 13:15, (KW 20)
02. Juni, 10:00, (KW 22)
14. Mai, 16:15, (KW 20)
19. Mai, 16:15, (KW 20)
29. Mai, 14:45, (KW 22)


>>>>>>>>Termine in den nächsten 7 Tagen:
Keine Termine im Zeitraum vom 02. November, 12:00, (KW 44) und 09. November,
```