- *Debug Debug*

Inhaltsverzeichnis

Dokumentation

- Java API
- Javainsel
- javabuch.de

Status

Noch 0 Minuten

Weg damit, das will ich gar nicht wissen!

Alles Speichern

Grundlagen

Implementieren Sie alle Aufgaben in den dafür vorgesehenen Dateien im Ordner "impl".

- Bitte lesen Sie die Angabe **komplett** durch, bevor Sie mit der Umsetzung beginnen.
- Halten Sie sich genau an die Angabe und geben Sie nur die geforderten Daten aus.
- Haben Sie **Schwierigkeiten** bei der Umsetzung einer der Teilaufgaben, **versuchen Sie andere Teile zuerst umzusetzen** und wenden Sie sich zuletzt nochmal der ungelösten Teilaufgabe zu. Sie erhalten auch Punkte für nicht vollständig implementierte Teilaufgaben

Programmieraufgabe

(30 Punkte)

Sie haben die Aufgabe, eine einfache Kalenderverwaltung zu implementieren.

Aufgabenstellung

Die Programmfunktion besteht aus einer einfachen Kalenderverwaltung. Der verwaltete Zeitraum ist ein Monat mit 31 Tagen. Die Zeit nach dem 31. Tag wird nicht betrachtet.

An jedem Tag kann maximal eine Aufgabe eingetragen werden. Es wird keine Uhrzeit berücksichtigt. Eine Aufgabe hat:

- einen Namen vom Typ String, der keine Leerzeichen enthält, z. B. (nehmen wir an, es geht um Kundenbesuche) `"Meier"`, sowie
- eine Priorität A, B oder C vom Typ char, z. B. `'A'` (A ist die höchste Priorität).

Ein Kalendereintrag ist somit durch die Nummer des Tags (zwischen 1 und 31) sowie Aufgabe und Priorität bestimmt.

Eine Aufgabe gleichen Namens kann auch an mehreren Tagen im Monat geplant sein.

Der Kalender wird dargestellt durch ein Array (der Länge 31) von Strings, wobei sich ein Element (d.h. ein String) aus der Priorität und dem Namen der Aufgabe zusammensetzt, z. B. `"A Meier"`.

Ein Tag, für den keine Aufgabe eingetragen ist, ist auf *keine Aufgabe* gesetzt: dies wird durch die Priorität '/' und einen leeren Namen der Aufgabe dargestellt, das entsprechende Array-Element hat also den Wert `"/"`.

Einlesen der Befehle

Das Programm wird zeilenweise bedient. Jede Zeile enthält zuerst als Befehlskennung ein Zeichen, das den Befehl identifiziert, und danach die Parameter des Befehls.

In der `main`-Methode der Klasse `Kalender` wird von der Standardeingabe (`System.in`) mittels Scanner eingelesen.

Das Programm erwartet bis zum Ende der Eingabe (EOF durch Ctrl+D) beliebig viele der folgenden Befehle:

- `k` **`tag`** zeigt den Kalender ab (inklusive) Tag *ab* an (ist teilweise in der Methode `tagToString` bereits vorgegeben): es wird die Zeile "Aufgaben:" ausgegeben, danach wird in einer Zeile pro Tag dessen Nummer und der entsprechende Eintrag ausgegeben, z. B. `5: A Meier`
- `e` **`tag prio aufgabe`** tragt die Aufgabe *aufgabe* mit Priorität *prio* am Tag *tag* des Monats ein. Der Eintrag erfolgt nur, wenn der Tag frei ist, oder eine Aufgabe niedrigerer Priorität eingetragen ist. Sonst passiert nichts.
- `w` **`tag prio aufgabe`** tragt die Aufgabe wöchentlich ein. Wie *eintragen*, jedoch wöchentlich, d.h. am Tag *tag*, 7 Tage danach, usw. bis maximal am Tag 31. Jeder Eintrag erfolgt nur, wenn der Tag frei ist oder eine Aufgabe niedrigerer Priorität eingetragen ist. Die Einträge können also ganz, teilweise oder gar nicht erfolgen.
- `z` **`prio`** gibt die Anzahl der Einträge mit Priorität *prio* aus.
- `a` **`aufgabe prio`** ändert für jeden Eintrag der Aufgabe *aufgabe* die Priorität auf *prio*.
- `v` **`von bis um`** verschiebt alle Aufgaben im Zeitraum von Tag *von* bis Tag *bis* (jeweils inklusive) um *um* Tage nach hinten. Für einen Tag, auf den ein Eintrag neu zu liegen käme, darf es allerdings noch keine Aufgabe gleicher oder höherer Priorität geben. Ein eventueller Eintrag mit niedriger Priorität wird ersetzt (überschrieben). Falls der Tag, auf den der Eintrag neu zu liegen käme, nach dem 31. ist, soll die Verschiebung für diesen Eintrag nicht durchgeführt werden (es kann somit vorkommen, dass nur ein vorderer Teil der (durch *von* und *bis* bestimmten) Einträge verschoben wird). Ein Tag, für den durch das Verschieben ein Eintrag entfernt wird und der dadurch keine Aufgabe mehr enthält, wird auf *keine Aufgabe* gesetzt (s.o.).

Die angeführten Befehle sollen die zu implementierenden Methoden `anzeigen, eintragen, eintragenWoechentlich, anzahl, aendern` und `verschieben` nutzen.

Hinweis: Sie können davon ausgehen, dass alle Eingaben korrekt sind. Sie brauchen also die einzulesenden Befehle und Werte nicht auf Gültigkeit zu überprüfen und keine Fehlerbehandlung implementieren.

Gefragte Methoden

`public static void anzeigen(int tag, String[] kalender)`

> Gibt den Kalender ab einem bestimmten Tag (inklusive) zeilenweise aus. Davor wird in einer Zeile "Aufgaben:" ausgegeben.

`public static void eintragen(String[] kalender, int tag, char prio, String aufgabe)`

> Tragt eine Aufgabe mit ihrer Priorität an einen bestimmten Tag in einen Kalender ein, wenn der bisherige Eintrag an diesem Tag leer ist oder eine niedrigere Priorität besitzt.

`public static void eintragenWoechentlich(String[] kalender, int tag, char prio, String aufgabe)`

> Wöchentliches Eintragen einer Aufgabe mit ihrer Priorität in einen Kalender. Jeder Eintrag geschieht nur, wenn der bisherige Eintrag an diesem Tag leer ist oder eine niedrigere Priorität besitzt.
>
> *Achtung: Nutzen Sie für die Implementierung dieser Methode die Methode eintragen!*

`public static int anzahl(String[] kalender, char prio)`

> Berechnet die Anzahl der Kalendereinträge mit einer bestimmten Priorität.

`public static void aendern(String[] kalender, String aufgabe, char prio)`

> Ändert die Priorität für jeden Eintrag einer Aufgabe in einem Kalender.

`public static void verschieben(String[] kalender, int von, int bis, int um)`

> Verschieben einer Reihe von Einträge eines Kalenders.
>
> *Beachten Sie die genaue Definition in der Angabe!*

Herangehensweise

Folgende Vorgehensweise ist bei der Umsetzung der Aufgabe empfehlenswert:

- Fügen Sie eine geeignete Deklaration des Arrays ein
- Initialisieren Sie den Kalender, d.h. setzen sie jeden Tag auf *keine Aufgabe*
- Implementieren Sie die gefragten Methoden (sowie entsprechend das Einlesen und Ausführen des jeweiligen Befehls) in der `main-Methode`.
- Implementieren Sie die Methode `anzeigen`
- Implementieren die den weiteren Befehlen entsprechenden Methoden. Empfehlenswert: in der obigen (und im Codegerüst) vorgegebenen Reihenfolge!

Testfälle

Kompilieren Sie früh und oft und versuchen Sie Fehler sofort zu beheben. Testen Sie nach jedem Kompilieren Ihre Implementierung. Zum Testen Ihrer Implementierung können Sie die mitgelieferten Ein- und Ausgabe-Paare im Ordner `io` nutzen. Rufen Sie Ihr Programm (z.B. für `spec.$01`) wie folgt auf:

```
java Kalender < ../io/spec.i01
```

Vergleichen Sie die Ausgabe des Programms anschließend mit der geforderten Ausgabe in den entsprechenden Dateien (z.B.: `spec.o01`). Wenn Sie direkt auf der Konsole Eingaben vornehmen, beenden Sie die Eingabe mit Strg+D.

Automatisiertes Testen

Um alle vorhandenen Testfälle auf einmal auszuführen und so Ihre Implementierung ausführlich zu prüfen, können Sie mit dem aus der Übung bekannten Skript `batchrun` automatisch alle Testfälle ausführen. Führen Sie dazu im Terminal im Testverzeichnis folgenden Befehl aus: `batchrun`
Dieses Skript nutzt, so vorhanden, auch weitere Testfälle aus dem Verzeichnis `batch`.

spec.$1

INPUT

```
e 25 A Fischer
e 26 B Huber
e 28 C Maier
e 30 A Schmidt
k 20
v 25 30 1
k 20
```

OUTPUT

```
Aufgaben:
20: /
21: /
22: /
23: /
24: /
25: A Fischer
26: B Huber
27: /
28: C Maier
29: /
30: A Schmidt
31: /
Aufgaben:
20: /
21: /
22: /
23: /
24: /
25: /
26: A Fischer
27: B Huber
28: /
29: C Maier
30: /
31: A Schmidt
```

spec.$2

INPUT

```
e 8 A Schulze
e 15 C Hauser
e 29 B Fischer
k 1
w 1 B Meier
k 1
```

OUTPUT

```
Aufgaben:
1: /
2: /
3: /
4: /
5: /
6: /
```

7: /
8: A Schulze
9: /
10: /
11: /
12: /
13: /
14: /
15: C Hauser
16: /
17: /
18: /
19: /
20: /
21: /
22: /
23: /
24: /
25: /
26: /
27: /
28: /
29: B Fischer
30: /
31: /
Aufgaben:
1: B Meier
2: /
3: /
4: /
5: /
6: /
7: /
8: A Schulze
9: /
10: /
11: /
12: /
13: /
14: /
15: B Meier
16: /
17: /
18: /
19: /
20: /
21: /
22: B Meier
23: /
24: /
25: /
26: /

```
27: /
28: /
29: B Fischer
30: /
31: /
```

spec.$3

INPUT

```
e 31 A Meier
k 31
```

OUTPUT

```
Aufgaben:
31: A Meier
```

spec.$4

INPUT

```
w 1 B Schulze
w 3 A Meier
w 6 C Reindl
e 2 B Gerstl
a Meier B
k 1
```

OUTPUT

```
Aufgaben:
1: B Schulze
2: B Gerstl
3: B Meier
4: /
5: /
6: C Reindl
7: /
8: B Schulze
9: /
10: B Meier
11: /
12: /
13: C Reindl
14: /
15: B Schulze
16: /
17: B Meier
18: /
19: /
20: C Reindl
21: /
22: B Schulze
23: /
24: B Meier
25: /
26: /
27: C Reindl
28: /
29: B Schulze
30: /
31: B Meier
```