Übung 01: Klassen

Abgabetermin: 12. 3. 2020, 8:15

Name: Florian Daniel **Matrikelnummer**: k1557830

Für die Lösung spannende Codestellen wurden kommentiert.

Gruppen Informatik (bitte ankreuzen):

- ☐ G1 (Krismayer)
- ☐ G2 (Prähofer)
- ☐ G3 (Weninger)
- ☐ G4 (Weninger)
- ☐ G5 (Schörgenhumer)
- ☐ G6 (Löberbauer)

Gruppen WIN (bitte ankreuzen): G1, G2, G3 -> nicht sicher

- ☒ G1 (Khalil/Hummer)
- ☐ G2 (Khalil/Hummer)
- ☐ G3 (Khalil/Hummer)
- ☐ G4 (Weißenbek)

Aufgabe	Punkte	abzugeben über Moodle	Punkte
Übung 1	24	Java Programmcode in Zip-Datei Java Programmcode als pdf-Datei	

Übung 01: Todo-Liste (24 Punkte)

In dieser Übung soll ein Programmsystem zur Verwaltung einer Liste von „Todos“ realisiert werden.

Todos haben folgende Eigenschaften:

- eine Beschreibung
- ein Fälligkeitsdatum
- einen Status (`Open`, `Done`)
- eine eindeutige Id (die automatisch durch das Programm vergeben werden soll)

Das Programmsystem soll das Anlegen, Abfragen und die Verwaltung der Todo-Einträge unterstützen und dazu eine entsprechende Programmschnittstelle zur Verfügung stellen.

Klasse `Todo`:

Implementieren Sie eine Klasse `Todo` für Todo-Einträge und eine Enumerations-Klasse `Status` für den Status von Todos. Implementieren Sie entsprechende Felder, Konstruktoren, Zugriffsmethoden und eine Methode zum Abschließen des Todos (Status auf `Done` setzen). Achten Sie besonders auf die Verwendung der richtigen Access-Modifier und die Verwendung von `final`.

Verwenden Sie die Klasse `java.time.LocalDate` für die Repräsentation des Datums (siehe kurze Anleitung zur Verwendung findet sich im Anhang).

Klasse `TodoManager`:

Implementieren Sie dann eine Klasse `TodoManager` zur Verwaltung einer Liste von Todos. Der `TodoManager` soll folgende Operationen zur Verfügung stellen:

- Anfügen eines neuen Todos mit Beschreibung und Datum
- Suche nach einem Todo mit gegebener Id
- Zugriff auf alle Todos
- Zugriff auf alle Todos bis zu einem bestimmten Datum

Alle relevanten Code-Stellen wurden kommentiert, um den Lösungsweg zu erl

- Zugriff auf alle offenen Todos
- Zugriff auf offene Todos bis zu einem bestimmten Datum
- Zugriff auf alle erledigten Todos
- Zugriff auf erledigte Todos bis zu einem bestimmten Datum

Achten Sie besonders auf die Gestaltung der Methodensignaturen der Klasse `TodoManager`. Die Rückgabewerte der Zugriffsmethoden sollen Arrays von Todos sein, z.B.,

```
public Todo[] getUntil(LocalDate until)
```

wobei das Array **keine Nullwerte** haben darf und die Todos im Array nach dem Datum **aufsteigend sortiert** sein müssen.

Anforderung verkettete lineare Liste:

Die Verwaltung der Todos in der Klasse `TodoManager` sollen Sie eine verkettete lineare Liste implementieren (Sie dürfen **keine Klassen** der und auch **keine Methoden der Java-Collection-Bibliothek** verwenden). Die `Todo`-Einträge sollen in der linearen Liste nach Fälligkeitsdatum aufsteigend sortiert eingefügt werden.

Hinweise:

Orientieren Sie sich bei der Implementierung der linearen Liste am Musterbeispiel aus der Lehrveranstaltung.

Um über die Knoten der Liste zu iterieren kann man folgende Schleifenform verwenden:

```
Node node = head;
while (node != null) {
    …
    node = node.next;
}
```

Zur Unterstützung der geforderten Methoden, entwickeln Sie am besten folgende Methode `count`

```
int count(LocalDate until, Status status)
```

die alle Todos bis zum gegebenen Datum `until` und mit gegebenen Status `status` zählt. Dabei bedeutet ein Parameterwert `null`, dass das Kriterium nicht berücksichtigt wird. Zum Beispiel zählt

```
int n = count(null, null);
```

alle Todos und

```
int nOpen = count(null, Status.OPEN);
```

alle offenen Todos.

Implementieren Sie des Weiteren zur Unterstützung der geforderten Methoden eine Methode `get`

```
Todo[] get(LocalDate until, Status status)
```

die alle Todos bis zum gegebenen Datum `until` und mit gegebenen Status `status` in einem Array liefert. Zum Beispiel, soll

```
Todo[] openTodos = get(null, Status.OPEN);
```

alle offenen Todos in einem Array liefern. Bei der Implementierung von `get` können Sie mit `count` die Anzahl der Elemente bestimmen und damit ein Array entsprechender Größe erzeugen.

Verwenden Sie die Methode `get` bei der Definition der Methode des TodoManagers.

Hauptprogramm:

Zu Testzwecken ist im Download ein File `todos.txt` mit einigen Testdaten und eine Klasse `todos.app.TodosMain` enthalten. Die Klasse `TodosMain` zeigt, wie die Daten unter Verwendung der Klasse `inout.In` eingelesen werden können. Testen Sie Ihr Programm indem Sie

- aus den Daten ihren `TodoManager` befüllen
- alle Todos ausgeben

- die Todos bis zu einem bestimmten Datum ausgeben
- einige Todos abschließen
- die abgeschlossenen Todos ausgeben
- die noch offenen Todos ausgeben
- die offenen zu einem bestimmten Datum ausgeben
- die abgeschlossenen bis einem Datum löschen
- und wiederrum alle Todos ausgeben

Beispielausgabe:

```
All Todos:
===========
   2: 2017-03-02 - Attend SW2 class              : OPEN
   4: 2017-03-03 - Study packages in Java        : OPEN
   0: 2017-03-08 - Program SW2 assignment 1      : OPEN
   1: 2017-03-09 - Attend SW2 class              : OPEN
   5: 2017-03-09 - Attend SW2 lecture            : OPEN
   7: 2017-03-09 - Submit SW2 assignment 1       : OPEN
   3: 2017-03-10 - Study inheritance in Java     : OPEN
   6: 2017-03-14 - Program SW2 assignment 2      : OPEN

Until March 9:
==============
   2: 2017-03-02 - Attend SW2 class              : OPEN
   4: 2017-03-03 - Study packages in Java        : OPEN
   0: 2017-03-08 - Program SW2 assignment 1      : OPEN
   1: 2017-03-09 - Attend SW2 class              : OPEN
   5: 2017-03-09 - Attend SW2 lecture            : OPEN
   7: 2017-03-09 - Submit SW2 assignment 1       : OPEN
```

- Abschließen von 2, 4, 1, 5

```
Done:
=====
   2: 2017-03-02 - Attend SW2 class              : DONE
   4: 2017-03-03 - Study packages in Java        : DONE
   1: 2017-03-09 - Attend SW2 class              : DONE
   5: 2017-03-09 - Attend SW2 lecture            : DONE

Still open:
===========
   0: 2017-03-08 - Program SW2 assignment 1      : OPEN
   7: 2017-03-09 - Submit SW2 assignment 1       : OPEN
   3: 2017-03-10 - Study inheritance in Java     : OPEN
   6: 2017-03-14 - Program SW2 assignment 2      : OPEN

Still open until Until March 9:
===============================
   0: 2017-03-08 - Program SW2 assignment 1      : OPEN
   7: 2017-03-09 - Submit SW2 assignment 1       : OPEN
```

Anhang: Hinweise zur Verwendung von `LocalDate` und `In` und `Out`

Verwendung der Klasse `java.time.LocalDate`

Erzeugen von `LocalDate`-Objekten mit `LocalDate.of`, z.B.:

```
LocalDate march5 = LocalDate.of(2020, 3, 5);
```

Vergleich von `LocalDate`-Objekten mit `isBefore`, `isEqual`, `isAfter`, z.B.:

```
if (march1.isBefore(march2))
if (march1.isAfter(march2) || march1.isEqual(march2))
```

Klasse `inout.In` und `inout.Out`

Im Download zur Übung finden Sie die Klassen `inout.In` und `inout.Out`, die eine einfache Ein- und Ausgabe von unterschiedlichen Werten von der Konsole oder von Files erlauben. Die Klasse `todos.app.TodosMain`, die ebenso im Download enthalten ist, zeigt, wie man mit der Klasse `inout.In` arbeiten kann.