# Übung 01: Klassen

Abgabetermin: 9. 3. 2017, 8:15

**Name**: _____ **Matrikelnummer**: _____

**Informatik:** ☐ G1 (Marr) ☐ G2 (Prähofer) ☐ G3 (Prähofer) ☐ G3 (Löberbauer)

**WIN:** ☐ G1 (Khalil) ☐ G2 (Hummel) ☐ G3 (Khalil)

| Aufgabe | Punkte | abzugeben schriftlich | abzugeben elektronisch | korr. | Punkte |
|---|---|---|---|---|---|
| Übung 1 | 24 | Java-Programm,<br>Ausgabe eines Testlaufs | Java-Programm | ☐ | |

## 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)

Klasse `Todo`:

Implementieren Sie eine Klasse `Todo` und eine Enumeration `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 eine Liste von Todos. Der `TodoManager` soll (zumindest) 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
- 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
- Löschen aller abgeschlossenen 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.

Hinweise:

Die Verwaltung der Todos in der Klasse `TodoManager` ist Ihnen überlassen. Sie dürfen aber **keine Klassen** der und auch **keine Methoden der Java-Collection-Bibliothek** verwenden. Es bietet sich entweder eine verkettete Liste oder ein Array (das bei Bedarf vergrößert wird) an. Die Todos sollten **aufsteigend sortiert** gespeichert werden.

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
```

- Löschen bis 9. 3. 2017

```
All Todos:
===========
   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
```

Abzugeben sind:

schriftlich im Postkasten:

- Java-Programm schön formatiert ausgedruckt
- Ausgabe eines Testlaufs

elektronisch über Upload von der LVA Homepage:

- Java-Programm (alle ihre Klassen) in einem Zip verpackt.

## 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 march1 = LocalDate.of(2017, 3, 1);
```

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.