hochschule mannheim

Techniken der Programmentwicklung (TPE)
Sommersemester 2014

FAKULTÄT FÜR INFORMATIK

Pflichtübung 4

Ausgabe: 26.05.2014
Abgabe: 19.06.2014 (23:50 Uhr)
Testat (IMB): 23.06.2014 (10:30–12:45 Uhr)
Testat (UIB): 20.06.2014 (15:20–17:30 Uhr)

Aufgabe 1: Kinoprogramm — 70 Punkte

Nachdem ein ehemaliger amerikanischer Geheimagent ausgeplaudert hat, welche Überwachungsmaßnahmen dank Ihrer Graphensuche möglich sind, entlässt Sie der Geheimdienst fristlos. Wieder auf der Suche nach einem neuen Job treffen Sie einen ehemaligen Chemielehrer, der inzwischen zum Serienstar aufgestiegen ist. Er vermittelt Ihnen ein Projekt bei einer großen Kinokette. Ihre Aufgabe ist, Ihr unbändiges Wissen über Collections in eine Verwaltungssoftware für Kinos zu entwickeln.

(a) **Problem und Implementierung**

Ihre Software soll die Programme einer beliebigen Anzahl von Kinos verwalten. Jedes `Kino` hat einen *Namen*, liegt in einer *Stadt* und besitzt einen oder mehrere Säle, in denen Filme gespielt werden können. Jeder `Saal` hat einen *Namen* und eine feste Anzahl von *Sitzplätzen*.

Ein `Film` zeichnet sich durch einen *Titel*, eine *Laufzeit* und eine *Altersfreigabe* aus. Die Altersfreigabe eines Films kann die Stufen *ohne Altersbeschränkung*, *ab 6 Jahre*, *ab 12 Jahre*, *ab 16 Jahre* und *ohne Jugendfreigabe* haben.

Ein Film kann zu verschiedenen Zeiten in verschiedenen Sälen laufen. Hierbei sind die Zeiten frei wählbar zwischen 0:00 Uhr und 23:59 Uhr. Es kann auch passieren, dass ein Film gleichzeitig oder zu sich überschneidenden Zeiten in mehreren Sälen läuft, wenn es ein echter Blockbuster ist – oder der Ersteller dieser Aufgabe den Film ganz besonders gut findet.

Der Verwender Ihrer Software kann

- Kinos mit entsprechenden Sälen erzeugen. Ein Kino ohne Namen, Stadt und Saal soll nicht angelegt werden können.
- Filme mit Startzeiten den Sälen zuordnen. Bei einer ungültigen Startzeit (außerhalb des möglichen Bereiches soll eine `IllegalTimeException` geworfen werden).
- Auslesen aller Filme mit Ihren Anfangszeiten als Array sortiert nach der Anfangszeit. Filme können doppelt vorkommen, wenn sie unterschiedliche Anfangszeiten haben `getAlleFilmeMitZeiten`.
- Auslesen aller Filme in einem bestimmten Saal mit den entsprechenden Anfangszeiten als Array (`getFilmeFuerSaalMitZeiten`). Die Filme sind nach der Startzeit sortiert.
- Auslesen aller Filme, die im Kino laufen als Array. Hierbei soll jeder Film nur einmal enthalten sein, auch wenn er zu mehreren Zeiten und in unterschiedliche Sälen läuft (`getAlleFilme`). Die Methode ist überladen und erlaubt in einer Variante die Angabe eines Sortierkriteriums als Enumeration. Wird kein Kriterium angegeben, werden die Filme nach dem Namen sortiert.

Zusätzlich soll das Kino noch die Möglichkeit bieten, über die `toString`-Methode den gesamten Spielplan auszugeben. Beispielsweise sieht die Ausgabe dann so aus:

```
Cinemaxx in Mannheim
Saal 'Blauer Saal' (250 Plaetze)
  14:00 -- Batman Begins [ab 12 Jahre] 134 min
  17:00 -- Batman Begins [ab 12 Jahre] 134 min
  20:00 -- Batman Begins [ab 12 Jahre] 134 min
  23:00 -- Batman Begins [ab 12 Jahre] 134 min

Saal 'Grüner Saal' (200 Plaetze)
```

```
  15:00 -- Barbie - Die Prinzessinnen-Akademie [ohne Altersbeschränkung] 81 min
  17:00 -- Ice Age 3 [ohne Altersbeschränkung] 90 min
  19:00 -- Ice Age 3 [ohne Altersbeschränkung] 90 min
  21:00 -- Machete [keine Jugendfreigabe] 100 min

Saal 'Studio' (150 Plaetze)
  15:00 -- Ice Age 3 [ohne Altersbeschränkung] 90 min
  17:00 -- Trainspotting [keine Jugendfreigabe] 89 min
  20:00 -- Pulp Fiction [ab 16 Jahre] 148 min
  23:00 -- From Dusk till Dawn [ab 16 Jahre] 87 min

Saal 'Kellerloch' (30 Plaetze)
  20:00 -- Chocolat [ab 6 Jahre] 121 min
  23:00 -- Trainspotting [keine Jugendfreigabe] 89 min
```

Implementieren Sie die Klasse `Kino` so, dass Objekte von ihr in einer foreach-Schleife verwendet werden können, um über das gesamte Programm zu iterieren (Film und Anfangszeit). Bei jedem Schleifendurchlauf soll ein Objekt zurückgegeben werden, dass die Startzeit und den Film enthält. Filme können doppelt vorkommen, wenn sie unterschiedliche Anfangszeiten haben.

Benutzen Sie für die Darstellung der Startzeit eine eigene Klasse `Zeit`, da die von Java gelieferten Klassen `Date` und `Calendar` sehr unhandlich in der Benutzung sind. Die Zeit soll auch aus einem String im Format *HH:MM* erzeugt werden können (`parse`) und wieder in einen solchen konvertiert werden können.

Filme sollen nach allen drei Attributen (Name, Altersfreigabe, Laufzeit) sortiert werden können. Die natürlich Sortierreihenfolge von Filmen ist nach dem Namen. Implementieren Sie die Klassen, die für diese weiteren Sortierkriterien nötig sind als statische geschachtelte Klassen im Film.

Denken Sie daran, bei allen Klassen die von `Object` geerbten Methoden sinnvoll zu überschreiben.

Um Ihre Collection-Fähigkeiten auf eine besondere Probe zu stellen, verlangt der Auftraggeber, dass die interne Speicherung im Kino eine *Map* mit dem Saal als Schlüssel verwendet.

Implementieren Sie die hier beschrieben Funktionalität mit Hilfe von Collections. Führen Sie weitere Klassen ein, wenn dies nötig ist, um eine elegante Implementierung zu erreichen.

(b) **Test**

Testen Sie Ihre Implementierung mit Unit-Tests. Sie können das oben als Beispiel gegebene Kino als einen Ihrer Testfälle benutzen.

Aufgabe 2: Collatz-Folge **30 Punkte**

(a) Die iterative Collatz-Folge ist für natürliche Zahlen wie folgt definiert:

$$f : \mathbb{N} \to \mathbb{N}, f(n) = \begin{cases} \frac{n}{2} & \textit{wenn } n \textit{ gerade} \\ 3n+1 & \textit{wenn } n \textit{ ungerade} \end{cases}$$

Die angegebene Formel erzeugt z. B. für die Startzahl 13 die Folge:

```
13 -> 40 -> 20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1
```

Man sieht, dass die Folge (mit 13 beginnend und mit 1 endend) 10 Werte enthält. Obwohl immer noch nicht bewiesen ist, dass die Collatz-Folge bei beliebigen Startwerten immer mit 1 endet, dürfen Sie dies für Ihre Lösung einfach annehmen.

Schreiben Sie eine Klasse `Collatz`, die man mit einem Startwert parametrieren kann und aus der man dann über einen `Iterator` die Elemente der Collatz-Folge nacheinander auslesen kann. Den Iterator implementieren Sie bitte als anonyme innere Klasse in `Collatz`.

(b) Testen Sie Ihre Collatz-Implementierung mit JUnit-Tests.

(c) Schreiben Sie ein Programm, mit dem Sie herausfinden können, bei welcher Startzahl unterhalb von einer Millionen die längste Collatz-Folge entsteht. Dieses Programm soll mit mehreren Threads gleichzeitig arbeiten, um möglichst schnell zu einer Lösung zu kommen. Testen Sie das Programm mit mindestens vier Threads.

Geben Sie an, zu welchem Ergebnis Ihr Programm gekommen ist, d. h. bei welcher Startzahl die längste Folge entsteht und wie viele Elemente diese längste Folge hat.

Achtung
Bitte beachten Sie folgendes, damit es nicht zu unnötigen Punktabzügen in der Bewertung kommt:

- Benennen Sie Klassen und Methoden konsistent und verständlich. Klassen sollten Nomen als Namen, Methoden Verben haben.
- Dokumentieren Sie alle Klassen, Interfaces, Konstruktoren und Methoden mit JavaDoc. Dokumentieren Sie auch alle Parameter, Rückgabewerte und Ausnahmen.
- Formatieren Sie Ihren Code konsistent. Ein guter Standard sind 4 Spaces Einrückung pro Ebene ohne Verwendung von Tabulatoren.
- Halten Sie sich an die Sun Java-Code-Convention
  http://www.oracle.com/technetwork/java/codeconventions-150003.pdf
- Programmieren Sie defensiv und testen Sie Eingabewerte auf deren Gültigkeit. Ihr Programm darf auch mit Daten nicht abstürzen, die außerhalb der Aufgabenstellung liegen.
- Machen Sie keine Konsoleneingaben oder -Ausgaben, es sei denn die Aufgabe fordert dies explizit.
- Halten Sie Daten und Methoden zusammen. Trennen Sie diese nicht unnötig auf.
- Kopieren Sie keinen Code sondern versuchen Sie mit den bekannten Mitteln der Objektorientierung Code-Duplikate zu vermeiden.