P1: Academy Awards (Oscar)

Abgabetermin: 18. April 2021

Punktzahl: 29 + 3 (Einhaltung der Coding Richtlinien)

In diesem Projekt wird primär das Thema Collections behandelt. Sie sollen hierzu eine einfache Anwendung implementieren, die Folgendes ermöglicht:

- Parsen eines Datensatzes mit Oscar-Nominierungen sowie eines Datensatzes mit Geburts- und Sterbedaten von SchauspielerInnen, wobei die Daten in zwei (dynamischen) Mengen von Objekten gespeichert werden sollen.
- Filtern der Daten mit dem Ziel, verschiedene Informationen daraus zu extahieren.

Importieren Sie zunächst die Archivdatei `P1-vorgaben.zip` in Eclipse (als existierendes Projekt aus einem Archivfile). Ändern Sie dann den Namen des Projekts in <IhrNachname.IhrVorname.P1>

Vorgaben

Das Projektarchivfile `P1-vorgaben.zip` definiert bereits eine Grundstruktur, die Sie für Ihre Implementierung verwenden sollen. In den Vorgaben sind mehrere Packages vorgegeben, die nachfolgend kurz erläutert werden.

Darüberhinaus gibt es in dem Projekt ein Unterverzeichnis `resources`, in dem sich die beiden CSV-Dateien mit den Daten zu den Nominierungen und den SchauspielerInnen befinden. Die Datei `nominees.csv` beinhaltet die Nominierungen für die Oscars für den besten Schauspieler / die beste Schauspielerin in einer Haupt- oder einer Nebenrolle in den Jahren 1927 - 2010. Die Datei `actors.csv` enthält für jede(n) genannte(n) SchauspielerIn das Geburtsdatum sowie ggf. das Todesdatum. Dabei bilden die in dieser Datei genannten SchauspielerInnen nur eine Teilmenge der in den Nominierungen genannten SchauspielerInnen.

general

In diesem Package befindet sich lediglich eine Klasse `Parameters`, in der zwei konstante Strings definiert sind, die die Pfade zu den beiden Dateien im `resources` Unterverzeichnis beinhalten.

model

Dieses Package enthält einen Aufzählungstyp `Category`, welche sechs Kategorien bei den Academy Awards definiert, von denen allerdings hier nur die ersten vier verwendet werden. Für diesen Aufzählungstyp ist die `toString` Methode überschrieben. Die beiden Klassen `Actor` und `Nominee` repräsentieren eine Schauspielerin / einen Schauspieler bzw. eine Nominierung einer Schauspielerin / eines Schauspieles in der o.g.Kategorien in einem bestimmten Jahr. Beide Klassen beinhalten aktuell neben einem Konstruktor diverse Getter-Methoden. Dazu ist jeweils die `toString` Methode überschrieben.

gui

In diesem Package gibt es zwei Klassen: Die Klasse `P1GUI` dient dazu, eine Grafische Oberfläche bereitzustellen, die es ermöglicht, die Applikation am Ende interaktiv zu testen und dynamisch verschiedene Informationen aus den Daten zu filtern.

Die `Main`-Klasse muss zum Start der Applikation ausgeführt werden.

io

Das Package `io` umfasst zwei Klassen: Die Klasse `TextReader` bietet eine Klassenmethode `getText` an, die den Inhalt einer Textdatei einliest, wobei der Pfad der Datei als Parameter übergeben werden muss. Der Inhalt der Textdatei wird in einem `StringBuffer` zurückgegeben.

Die Klasse `TextParser` dient dazu, den Inhalt der beiden o.g. CSV Dateien zu parsen und daraus Objekte der Klasse `Actor` und `Nominee` zu generieren sowie diese in entsprechenden dynamischen Mengenstrukturen zu speichern.

filter

In diesem Package gibt es nur die Klasse `FilterAcademyAwards`, in der Sie später verschiedene Methoden entwickeln bzw. implementieren müssen.

Aufgabenstellung

1 (11P) Parsen von Daten und Erzeugen von Collections

Zunächst müssen die Daten in den beiden o.g. CSV-Dateien geparsed und auf Elemente der Klassen `Actor` und `Nominee` abgebildet sowie in zwei Sammlungen gespeichert werden.

(1a) (4 P) Vergleichbarkeit von Objekten

In diesem Teil sollen Sie die Klassen `model.Actor` und `model.Nominee` anpassen und erweitern. Verbessern Sie hierzu zunächst zwei Methoden, die bestimmte Werte für ein Objekt dieser Klasse liefern. Sorgen Sie dann dafür, dass zwei Elemente dieser Klassen jeweils vergleichbar sind. Damit können Sie später in sortierten Collections wie z.B. der Klasse `TreeSet` gespeichert werden.

- In der CSV Datei `actors.csv` werden die Namen der SchauspielerInnen in der Reihenfolge "Vorname(n) Nachname" angegeben. Ändern Sie die Methode `getName` so, dass diese den Namen in der Form "Nachname, Vorname(n)" zurückgibt. Sie dürfen davon ausgehen, dass der Nachname immer am Ende steht und mit (mindestens) einem Leerzeichen von dem/den Vorname(n) getrennt ist.

- Eine `String`-Repräsentation einer Instanz der Klasse `Actor` beinhaltet neben dem Namen noch das Geburts- und das Sterbedatum, bzw. nur den Namen und das Geburtsdatum, falls der/die betreffende SchausspielerIn noch lebt. Ergänzen Sie die `toString` Methode der Klasse `Actor` so, dass in letzterem Fall noch das aktuelle Alter in Jahren berechnet und dieses dann mit ausgegeben wird. Das Ergebnis sollte wie im folgenden Beispiel aussehen:

  ```
  Michael Caine (*14.03.1933, age: 88)
  ```

  Hinweis: Berücksichtigen Sie bei Ihrer Berechnung, ob der jeweilige Geburtstag im aktuellen Jahr bereits stattgefunden hat oder noch nicht.
  Anmerkung: Dieser Teil hat natürlich nichts mit der Sortierung zu tun, sondern nur mit der Darstellung der Daten.

- Überschreiben Sie in beiden Klassen die `equals` Methode. Zwei Objekte der Klassen `Actor` bzw. `Nominee` werden dann als gleich angesehen, wenn die Namen der zugehörigen Schauspieler identisch sind. Überprüfen Sie vor dem Vergleich der Namen jeweils, ob das als Parameter übergebene Objekt der gleichen Klasse angehört. Verwenden Sie hierzu den `instanceof` Operator.

- Implementieren Sie in beiden Klassen das `Comparable<E>` Interface. Instanzen der Klasse `Actor` sollen lexikalisch nach dem Nachnamen der jeweiligen Schauspieler verglichen werden (und wenn diese identisch sind, nach den Vornamen). Instanzen

der Klasse `Nominee` sollen lexikalisch nach den Namen der zugehörigen Filme verglichen werden. Sind diese identisch, soll der Name der Schauspieler zum Vergleich herangezogen werden.

(1b) (7 P) Parsen der Daten

In der Klasse `io.TextParser` sollen der Inhalt der beiden o.g. Dateien geparsed und die Ergebnisse genutzt werden, um je eine sortierte Sammlung von Objekten der Klasse `model.Actor` bzw. `model.Nominee` zu füllen. Hierzu ist in der Klasse Folgendes vorgegeben:

(1) Zwei Konstanten `ACTORS` and `AWARDS`, die beim Aufruf der Methode `parseData` (s.u.) anzeigen, welche der Dateien geparsed werden sollen

(2) Eine `HashMap aMap`, in der Instanzen der Klasse `Actor` gespeichert werden sollen. Dabei soll der Name des Schauspielers / der Schauspielerin als Schlüssel verwendet werden.

(3) Drei, noch leere Methoden, die von Ihnen zu vervollständigen sind.

(4) Eine `main`-Methode, die Sie ggf. zum Testen der Implementierung nutzen können.

Ergänzen Sie die Klasse `TextParser` wie im Folgenden beschrieben:

- Vervollständigen Sie die Methode `parseData`, so das in den beiden Fällen in dem vorgegebenen `switch`-Statement der aus der jeweiligen Datei eingelesene Text in ein Feld von Zeilen aufgeteilt wird, die dann jeweils der Methode `parseActorsLine` bzw. `parseNomineesLine` übergeben werden.
  Hinweis: Sie müssen den Parameter `set` beim Aufruf der Methoden jeweils passend casten.

- Implementieren Sie die Methode `parseActorsLine`, in der jeweils eine, als Parameter übergebene Zeile aus dem Text der Datei `actors.csv` geparsed werden soll. Daraus soll dann eine Instanz der Klasse `Actor` generiert werden, die in der als Parameter übergebenen `TreeSet` Instanz und zusätzlich in der HashMap `aMap` gespeichert werden soll.

  Jede Zeile in der Datei besteht aus zwei oder drei Spalten, die jeweils durch ein Semikolon getrennt sind. Die erste Spalte enthält den Namen, die zweite das Geburtsdatum und die dritte ggf. das Todesdatum. Tag und Monat sind dabei jeweils zweistellig angegeben und das Jahr vierstellig. Verwenden Sie eine passende Instanz der Klasse `java.time.format.DateTimeFormatter`, um die Daten zu parsen. Entfernen Sie immer die Leerzeichen am Ende des Namens.

- Implementieren Sie die Methode `parseNomineesLine`, in der jeweils eine, als Parameter übergebene Zeile aus dem Text der Datei `nominees.csv` geparsed werden soll. Daraus soll dann eine Instanz der Klasse `Nominee` generiert werden, die in der

als Parameter übergebenen `TreeSet` Instanz gespeichert werden soll. Jede Zeile in der Datei besteht aus fünf Spalten, die durch Kommata getrennt sind und in denen das Jahr der Nominierung, die Kategorie, sowie der Name des Schauspielers / der Schauspielerin, der Film und die Information gegeben ist, ob der/die Nominierte den Oscar gewonnen hat oder nicht (YES/NO). In den Text der Spalte mit dem Filmnamen ist dabei der Name der Rolle wie in folgendem Beispiel zu sehen eingebettet und muss von Ihnen extrahiert werden.

```
The King's Speech {'King George VI'}
```

Prüfen Sie beim Aufteilen einer Zeile in Spalten (Tokens), wieviele Elemente Sie gefunden haben und brechen Sie die Verarbeitung ab, wenn Sie nicht genau fünf Spalten gefunden haben. Hinweis: Der Konstruktor der Klasse `Nominee` benötigt als Parameter u.A. eine Referenz auf eine Instanz der Klasse `Actor`. Damit bei mehreren Nominierungen für denselben Schauspieler / dieselbe Schauspielerin immer das gleiche Objekt verwendet wird, müssen Sie hier immer erst anhand des Namens prüfen, ob die betreffende Person bereits in der `HashMap aMap` gespeichert ist. Ist dies der Fall, verwenden Sie die Referenz auf das dort gespeicherte Objekt. Anderenfalls müssen Sie zunächst eine neue Instanz der Klasse `Actor` generieren und diese in der HashMap speichern.

- In den Methoden `parseActorsLine` und `parseNomineesLine` bestehen die Zeilen jeweils aus 2-3 bzw. 5 Spalteneinträgen. In letzterem Fall kommte es allerdings aufgrund zusätzlicher Kommata in einer der Spalten bei einigen Zeilen dazu, dass beim Splitten der Zeile mehr Tokens generiert werden. Ergänzen Sie die beiden genannten Methoden so, dass eine Zeile nur dann nach dem Aufsplitten weiterverarbeitet wird, wenn mindestens zwei Tokens (Methode `parseActorsLine`) bzw. genau fünf Tokens (Methode `parseNomineesLine`) gefunden wurden. Anderenfalls werfen Sie jeweils eine eigene Exception mit einem passenden Text. Legen Sie hierzu eine neue Exception-Klasse an. Diese und weitere mögliche Exceptions (wie. z.B. `NumberFormatException`) sollen in der Methode `parseData` aufgefangen und, zusammen mit der betreffenden Zeile ausgegeben werden. Dabei soll das Programm anschließend nicht abgebrochen werden, sondern es soll die dann folgende Zeile ervarbeitet werden. Passen Sie die genannten Methode sowie die Methode `parseData` dementsprechend an.

2 (18P) Filterung von Daten in Collections

Die Klasse `FilterAcademyAwards` enthält zwei `TreeSet` Objekte `actors` und `nominees`, die in der vorgegebenen `init`-Methode mit Hilfe der in Teil (1b) implementierten Methoden mit Instanzen der Klasse `Actor` bzw. `Nominee` gefüllt werden. Die `init`-Methode wird hierbei durch den Konstruktor aufgerufen.

Sie sollen nun verschiedene Methoden in der Klasse `FilterAcademyAwards` implementieren, die beim Aufruf in der Klasse `gui.P1GUI` Informationen aus den vorgegebenen Collections filtern und diese zurückgeben, so dass sie in der GUI dargestellt werden können.

(2a) (4 P) Hilfsmethoden

Implementieren Sie zunächst ein paar Hilfsmethoden, mit denen u.A. die Comboboxen in der GUI gefüllt werden können.

- Implementieren Sie die Methode `getActorNames`, die eine `Vector`-Instanz aller Namen von SchauspielerInnen generieren und zurückgeben soll.
- Implementieren Sie die Methode `getYears`, die eine `Vector`-Instanz aller Jahre generieren und zurückgeben soll, die in den Nominierungen genannt werden. Dabei soll in dem resultierenden Vektor kein Jahr doppelt vorkommen und die Jahre sollen aufsteigend sortiert sein.
- Implementieren Sie die Methode `getActor`, die in dem Attribut `actors` nach einem `Actor` Objekt sucht, dessen Namen dem Parameter `aName` entspricht. Wenn es ein solches Objekt gibt, soll dieses zurückgegeben werden, ansonsten der Wert `null`. Sorgen Sie dafür, dass die Suche beendet wird, sobald ein passendes Objekt gefunden wird.

(2b) (3 P) Oscar Gewinne und Nominierungen

Implementieren Sie die Methode `getAwardsActor`, die für eine(n) SchauspielerIn, dessen/deren Name als Parameter übergeben wird, alle Gewinne eines Oscars sowie alle weiteren Nominierungen zusammenträgt und einen String mit diesen Informationen sowie den vorhandenen Informationen über den/die SchauspielerIn zurückgibt. Für die Schauspielerin Katerine Hepburn sollte beispielsweise der Rückgabestring wie folgt aussehen:

```
Katharine Hepburn :  *12.05.1907, †29.06.2003
 Awards:
 1967: Guess Who's Coming to Dinner (Christina Drayton),  Category: Actress -- Leading
 1932: Morning Glory (Eva Lovelace),  Category: Actress -- Leading Role
 1981: On Golden Pond (Ethel Thayer),  Category: Actress -- Leading Role
 1968: The Lion in Winter (Queen Eleanor of Aquitaine),  Category: Actress -- Leading

 Nominations:
 1935: Alice Adams (Alice Adams´ [came in 2n),  Category: Actress -- Leading Role
 1962: Long Day's Journey into Night (Mary Tyrone),  Category: Actress -- Leading Role
 1955: Summertime (Jane Hudson),  Category: Actress -- Leading Role
 1951: The African Queen (Rose Sayer),  Category: Actress -- Leading Role
```

```
 1940: The Philadelphia Story (Tracy Lord),  Category: Actress -- Leading Role
 1956: The Rainmaker (Lizzie Curry),  Category: Actress -- Leading Role
 1942: Woman of the Year (Tess Harding),  Category: Actress -- Leading Role
```

Hat ein(e) SchauspielerIn entweder keinen Oscar gewonnen bzw. keine weitere Nominierung erhalten, soll dies durch den String "- - -" gekennzeichnet werden wie im folgenden Beispiel für den Schauspieler F. Murray Abraham:

```
F. Murray Abraham (*24.10.1939, age: 81)
 Awards:
 1984: Amadeus (Antonio Salieri),  Category: Actor -- Leading Role

 Nominations:
 ---
```

Hinweis: Durchlaufen Sie die Menge `nominees` nur einmal und sammeln Sie die Informationen zu den gewonnenen Preisen und den Nominierungen in zwei `StringBuffer` Instanzen.

(2c) (8 P) TOP Nominierungen

Hier sollen Sie nun für ein vorgegebenes jahr die (drei) Filme mit den meisten Nominierungen herausfiltern und einen String mit dem Ergebnis zurückgeben. Gehen Sie hierzu wie folgt vor:

- Legen Sie zunächst einmal eine innere Klasse an, mit deren Hilfe Sie die Anzahl der Nominierungen für einen Film zählen können. Statten Sie die Klasse mit zwei Attributen für den Namen des Films und einer Zählvariable aus, die mit dem Wert 1 initialisiert wird. Definieren Sie einen passenden Konstruktor und überschreiben Sie die `toString` Methode, die einen String mit den Werten der beiden Attribute zurückgeben soll, der wie in folgendem Beispiel aussehen sollte:

  ```
  The King's Speech  : 3
  ```

  Damit zwei Objekte dieser Klasse vergleichbar sind, überschreiben Sie die `equals` Methode und implementieren Sie das `Comparable` Interface. ~~Vergleichen Sie zwei Objekte dieser Klasse nur über die Zählvariable für die Anzahl der Nominierungen.~~ Vergleichen Sie in letzterem Fall zunächst die Zählvariablen für die Anzahl der Nominierungen. Falls diese gleich sind, ziehen Sie auch den Namen des Films als Vergleichskriterium heran.

- Implementieren Sie nun die Methode `getTopThreeMovies` wie folgt:
  - Legen Sie eine `HashMap` an, in der Sie später für jeden Film ein Zähler-Objekt (also ein Objekt der inneren Klasse) speichern können. Nutzen Sie den Namen des Films als Schlüsselwert.

- Durchlaufen Sie dann die Menge `nominees`. Für jeden Film, der in dem betreffenden, durch den Parameter `y` der Methode spezifizierten Jahr nominiert wurde, prüfen Sie dann, ob für diesen Film bereits ein Zähler-Objekt in der `HashMap` gespeichert ist. Wenn dies nicht der Fall ist, legen Sie ein neues Zähler-Objekt an und speichern Sie es in der Map. Anderenfalls erhöhen Sie den Zähler in dem betreffenden Objekt um eins.
- Holen Sie sich anschließend alle Zähler-Objekte, die in der `HashMap` gespeichert sind und speichern Sie diese in einem Feld. Sortieren Sie den Feld-Inhalt mit Hilfe der Methode `java.utils.Arrays.sort(Object[] a)`.
- Speichern Sie die String-Repräsentationen der Zählerobjekte mit den höchsten Zählerwerten in einer `StringBuffer` Instanz. Nehmen Sie hierzu die drei Zähler-Objekte mit den höchsten Werten sowie alle weiteren Zähler-Objekte mit dem gleichen Zählerwert wie das Zähler-Objekt, das an der Stelle des drittbesten Elements steht (da nicht zwischen Objekten mit dem gleichen Zählerwert unterschieden werden soll). Sie dürfen davon ausgehen, dass das Feld mindestens die Länge 3 hat.
- Geben Sie den Inhalt des `StringBuffers` in Form eines Strings zurück.

(2d) (3 P) Geburtstage

Implementieren Sie die Methode `filterActors`, die einen Text generieren und zurückgeben soll, der alle SchauspielerInnen enthält, die in einem vorgegebenen Monat bzw. an einem vorgegebenen Tag in einem vorgegebenen Monat geboren wurden. Die Methode hat hierzu zwei Parameter vom Typ `java.time.Month` und `int`. Hat der `int` Parameter den Wert 0, soll nur nach dem Monat gefiltert werden, ansonsten nach dem Monat und dem Tag. Dazu sollen die Ergebnisse aufsteigend nach dem Geburtsjahr der SchauspielerInnen sortiert werden. Für den Monat Januar und den Wert 0 sollte dann folgendes Ergebnis generiert werden:

```
Ray Milland :  *03.01.1907, †10.03.1986
Luise Rainer :  *12.01.1910, †30.12.2014
José Ferrer :  *08.01.1912, †26.01.1992
Loretta Young :  *06.01.1913, †12.08.2000
Ernest Borgnine :  *24.01.1917, †08.07.2012
Jane Wyman :  *05.01.1917, †10.09.2007
Paul Scofield :  *21.01.1922, †19.03.2008
Paul Newman :  *26.01.1925, †26.09.2008
Patricia Neal :  *20.01.1926, †08.08.2010
Gene Hackman (*30.01.1930, age: 91)
Robert Duvall (*05.01.1931, age: 90)
Alan Alda (*28.01.1936, age: 85)
Faye Dunaway (*14.01.1941, age: 80)
Diane Keaton (*05.01.1946, age: 75)
```

```
Penelope Cruz (*04.01.1953, age: 68)
Nicolas Cage (*07.01.1964, age: 57)
Christian Bale (*30.01.1974, age: 47)
```

Filtern Sie zunächst alle SchauspielerInnen mit dem passenden Datum aus dem Attribut `actors` und sammeln Sie sie in einer dynamischen Menge. die das `List` Interface implementiert. Sortieren Sie dann die Objekte mit Hilfe der Methode `java.utils.Collections.sort(List<T> list, Comparator<? super T> c)`. Verwenden Sie als `Comparator` Objekt eine Instanz einer anonymen Klasse, die das `Comparator` Interface implementiert und die die SchauspielerInnen nach dem Geburtsjahr sortiert. Sammeln Sie dann noch die `String`-Repräsentationen der SchauspielerInnen in einer `StringBuffer` Instanz und geben Sie dessen Inhalt als `String` zurück. Hinweise:

- Denken Sie daran, dass nicht für alle Instanzen der Klasse `Actor` ein Geburtsdatum gegeben ist.
- Sie müssen nur eine Methode des `Comparator`-Interfaces implementieren.

Der folgende Screenshot zeigt Ihnen die GUI mit ein paar Beispielergebnissen: