ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Science

Programmierung 2 - Sommersemester 2024

Prof. Dr. Markus Esch

Übung Nr. 8
Abgabe KW 27

1. Aufgabe (Codeopolis)

   Die Beispiellösung für die vorangegangene Aufgabe erweitert die Klasse `Harvest` um das Attribut `durability`. Dieses drückt die Haltbarkeit einer Ernte aus und hängt von den Wetterbedingungen im Erntejahr ab. Getreide mit geringerer Haltbarkeit verrottet schneller. Daher ist es sinnvoll, die Klasse `Silo` so anzupassen, dass bei der Entnahme von Getreide das Getreide mit der geringsten Haltbarkeit zuerst entnommen wird. Passen Sie daher die Klasse `Silo` so an, dass die `Harvest`-Objekte in einer Datenstruktur gespeichert werden, die es ermöglicht, das Getreide mit der geringsten Haltbarkeit zuerst zu entnehmen. Wählen Sie dazu eine geeignete Datenstruktur aus dem Java Collections Framework.

   Passen Sie die Methoden `Silo.decay` und `Silo.getFillLevel` so an, dass Streams für den Zugriff auf die Harvest-Objekte verwendet werden.

2. Aufgabe

   Erstellen Sie ein Programm zur Verwaltung der Bücher in einer Bibliothek, verwenden Sie dazu Java Collections und wo immer es möglich ist Streams. Setzen Sie folgende Anforderungen um.

   (a) Erstellen Sie eine Klasse `Book` mit den folgenden Attributen:

   - `title` (String)
   - `author` (String)
   - `year` (int)
   - `pages` (int)
   - `genre` (String)
   - `rating` (double) - Die Bewertung des Buches (1.0 bis 5.0)
   - `borrowed` (boolean) - Gibt an, ob das Buch ausgeliehen ist
   - `returnDate` (LocalDate) - Das Rückgabedatum des Buches, wenn es ausgeliehen ist

   Implementieren Sie die `Book`-Klasse so, dass Bücher anhand des Titels sortiert werden können.

   (b) Erstellen Sie eine Klasse `User` mit den folgenden Attributen:

   - `name` (String)
   - `readerID` (String)
   - eine Collection der ausgeliehenen Bücher.

Verwenden Sie eine geeignete Collection, um das Ausleihkonto eines Nutzers zu verwalten. Die Collection sollte die folgenden Eigenschaften haben:

- Sie soll die ausgeliehenen Bücher eines Benutzers speichern.
- Es sollen keine Duplikate erlaubt sein.
- Die Bücher sollen nach Rückgabedatum sortiert sein.

Fügen Sie der Klasse Methoden hinzu, um Bücher auszuleihen und zurückzugeben.

(c) Erstellen Sie eine Klasse `LibraryManagementSystem`. Diese Klasse verwaltet alle Bücher und alle Nutzer. Dazu soll die Klasse folgende Datenstrukturen besitzen:

- Verwenden Sie eine geeignete Datenstruktur, um alle Bücher der Bibliothek sortiert zu speichern.
- Verwenden Sie eine geeignete Datenstruktur, um Nutzer anhand ihrer `readerID` schnell zu finden.
- Verwenden Sie eine geeignete Datenstruktur, um alle verliehenen Bücher effizient nach Rückgabedatum zu verwalten.

Außerdem soll die Klasse Methoden bereitstellen, um:

- ein Buch zur Sammlung der Bücher hinzuzufügen.
- einem Nutzer ein Buch auszuleihen.
- alle Bücher anzuzeigen, die von einem bestimmten Benutzer ausgeliehen wurden.
- alle ausgeliehenen Bücher anzuzeigen, sortiert nach dem Rückgabedatum.
- die Bücher zu filtern, die nach einem bestimmten Jahr veröffentlicht wurden.
- die Bücher nach der Anzahl der Seiten zu sortieren und als Liste zurückzugeben.
- die Gesamtanzahl der Seiten aller Bücher in der Bibliothek zu berechnen.
- die Bücher nach ihrem Genre zu filtern.
- die durchschnittliche Bewertung pro Genre zu berechnen.
- die drei am besten bewerteten Bücher zurückzugeben.
- die Autoren mit den meisten Büchern zurückzugeben.
- die Bücher nach ihrer Bewertung zu sortieren.
- eine gefilterte und sortierte Liste der Bücher zurückzugeben. Der Filter und das Sortierkriterium sollen als funktionale Interfaces an die Methode übergeben werden.

Verwenden Sie bei der Implementierung Streams. Fügen Sie weitere Methoden nach Bedarf hinzu.

(d) Testen Sie ihre Library Management System mithilfe eines Command Line Interfaces. Verwenden Sie als Basis das in Moodle bereitgestellte Template `LibraryCLI.java` nutzen. Implementieren Sie die Methode `loadBooksFromCSV`, um Bücher aus einer CSV-Datei[1] zu lesen. Zum Testen können Sie die ebenfalls in Moodle bereitgestellte Datei `books.csv` nutzen.

[1] CSV steht für Comma-separated values. Jede Zeile der Datei enthält einen Datensatz. Die einzelnen Felder des Datensatzes sind durch Komma getrennt.

Lernziele:

Nach der Bearbeitung des Übungsblattes sollten Sie ...

- ... in der Lage sein für ein gegebenes Problem eine geeignete Collection auszuwählen.
- ... Collections in einer Anwendung nutzen können, um Objekte zu verwalten.
- ... die Methoden von Collections effektiv nutzen können.
- ... die Vorteile von Streams benennen können.
- ... in der Lage sein, Streams zu nutzen, um Daten aus Collections zu filtern, zu transformieren und zu sammeln.
- ... parallele und serielle Streams, unterscheiden und anwenden können.
- ... komplexe Operationen wie Mapping, Reduzierung und Gruppierung mithilfe von Streams durchführen können.
- ... benutzerdefinierte Stream-Operationen implementieren können.

Zusatzaufgaben

Die folgenden Aufgaben sind Zusatzaufgaben und nicht Teil des Peer Reviews! Sie können die Aufgaben freiwillig bearbeiten, um die Inhalte weiter zu üben. Selbstverständlich können Sie auch Unterstützung und Beratung zu den Zusatzaufgaben erhalten.

1. Aufgabe (Codeopolis)

   (a) In den Klassen `City`, `CityState` und `TurnResult` werden verschiedene Arrays verwendet, um z.B. die Menge des gepflanzten Getreides zu speichern. Dies hat den Nachteil, dass die Getreidetypen des Enums `Game.GrainType` umständlich auf den Array-Index abgebildet werden müssen. Passen Sie die Implementierung daher so an, dass geeignete Datenstrukturen aus dem Java Collection Framework genutzt werden.

   Hinweise:

   - In den meisten Fällen ist es sinnvoll, eine Map-Implementierung zu nutzen, da diese eine einfache Zuordnung des Getreidetyps (`Game.GrainType`) ermöglicht.
   - Beachten Sie, dass diese Änderung auch Änderungen in weiteren Klassen nach sich zieht.

   (b) Passen Sie ihre Implementierung an, so dass Sie Streams in `CityState` und `TurnResult` verwenden, wo immer dies möglich ist.

2. Aufgabe

   Es soll eine Anwendung für einen Musik-Streaming-Dienst implementiert werden. Der Dienst bietet die Möglichkeit verschiedene Abfragen auf die Sammlung aller Songs auszuführen. Nutzen Sie bei der Impelemntierung Java Collections und Streams. Setzen Sie folgende Anforderungen um:

   (a) Erstellen Sie eine Klasse `Song` mit den folgenden Attributen:

   - `title`: Der Titel des Songs
   - `artist`: Der Künstler des Songs
   - `album`: Das Album, zu dem der Song gehört
   - `genre`: Das Genre des Songs
   - `length`: Die Länge des Songs in Sekunden
   - `rating`: Die Bewertung des Songs (1.0 bis 5.0)
   - `playCount`: Die Anzahl der Male, die der Song abgespielt wurde
   - `year`: Das Erscheinungsjahr des Songs

   Implementieren Sie die `Song`-Klasse so, dass die Songs nacheinander nach den Kriterien Künstler, Album und Titel sortiert werden.

   (b) Erstellen Sie eine Klasse `MusicLibrary`, welche die Musiksammlung in einer Java Collection verwaltet. Wählen Sie dazu eine Collection, die keine Dopplungen erlaubt und die Songs sortiert speichert.

   Die Klasse soll folgende Methoden bereitstellen:

   - Songs hinzufügen und entfernen:
     - Eine Methode, um einen Song hinzuzufügen.
     - Eine Methode, um einen Song zu entfernen.

- Top-gespielte Songs pro Genre:
  - Eine Methode, die die Top 2 Songs pro Genre basierend auf der Anzahl der Wiedergaben zurückgibt.
- Durchschnittliche Bewertung pro Künstler:
  - Eine Methode, die die durchschnittliche Bewertung der Songs für jeden Künstler berechnet und zurückgibt.
- Längste Songs pro Album:
  - Eine Methode, die für jedes Album den längsten Song zurückgibt.
- Gesamtlänge aller Songs eines Genres:
  - Eine Methode, die die Gesamtlänge aller Songs eines bestimmten Genres in Minuten berechnet.
- Künstler mit mindestens einer Bewertung über 4.5:
  - Eine Methode, die alle Künstler zurückgibt, die mindestens einen Song mit einer Bewertung über 4.5 haben.
- Top-N Songs basierend auf Bewertung und Wiedergaben:
  - Eine Methode, die die Top-N Songs zurückgibt, basierend auf einer Kombination aus Bewertung und Anzahl der Wiedergaben.
- Songs nach Jahr filtern:
  - Eine Methode, die alle Songs eines bestimmten Jahres zurückgibt.
- Songs nach Jahr sortieren:
  - Eine Methode, die alle Songs nach ihrem Erscheinungsjahr sortiert zurückgibt.
- Songs nach benutzerdefinierten Kriterien sortieren:
  - Eine Methode, die Songs nach benutzerdefinierten Kriterien sortiert zurückgibt.
- Songs nach benutzerdefinierten Kriterien filtern:
  - Eine Methode, die Songs nach benutzerdefinierten Kriterien filtert und zurückgibt.
- Bereichsabfragen:
  - Eine Methode, die die Songs in einem bestimmten Bereich zurückgibt (zwischen zwei gegebenen Songs).
  - Eine Methode, die den nächsten Song nach einem bestimmten Song zurückgibt.
  - Eine Methode, die den vorherigen Song vor einem bestimmten Song zurückgibt.

Verwenden Sie bei der Implementierung Streams. Fügen Sie weitere Methoden nach Bedarf hinzu.

(c) Testen Sie ihre Music Library mithilfe eines Command Line Interfaces. Verwenden dazu das in Moodle bereitgestellte Template `MusicLibraryCLI.java`.

(d) Schreiben Sie JUnit-Tests für die Music Library.

3. Aufgabe

(a) Implementieren Sie folgende Methode:

```
public List<String> readTextFile(String file,
  String prefix,
  int minLength,
  int limit)
```

Die Methode soll die gegebene Textdatei `file` einlesen. Filtern Sie anschließend die ersten `limit` unterschiedlichen Wörter heraus, die mit dem Substring `prefix` beginnen und mindestens eine Länge von `minLength` haben. Sortieren Sie die resultierenden Wörter alphabetisch und geben Sie das Ergebnis als Liste zurück.

Wichtig: Implementieren Sie Ihre Lösung mithilfe eines Streams!

Testen Sie Ihre Lösung mit dem RFC 791 als Input: https://tools.ietf.org/rfc/rfc791.txt

(b) Implementieren Sie eine zweite Methode analog zur Methode `readTextFile` mit dem Unterschied, dass Sie eine Map zurückgeben, die die resultierenden Wörter gruppiert nach ihrer Länge enthält.

4. Aufgabe

Implementieren Sie einen MinHeap basierend auf einem Array fester Größe. In der Vorlesung Informatik 1 wurde dargestellt, wie dies sehr einfach zu realisieren ist. Beachten Sie dabei folgendes:

(a) Ihr MinHeap sollte das Interface `java.util.Queue<E>` implementieren. Sie müssen jedoch nur folgende Interface-Methoden implementieren, bei allen anderen können Sie eine `java.lang.UnsupportedOperationException` werfen:

i `public boolean offer(E e)`

ii `public E poll()`

iii `public E peek()`

(b) Verwenden Sie nicht die abstrakte Klasse `java.util.AbstractQueue<E>`

(c) Implementieren Sie den MinHeap als generische Klasse, welcher Objekte speichern kann, die das Interface `java.lang.Comparable` implementieren.

(d) Implementieren Sie eine Testklasse, um die Methoden Ihres MinHeaps zu testen.