hochschule mannheim

FAKULTÄT FÜR INFORMATIK

Prof. Dr. Peter Knauber

PR1 – Programmieren 1 Bachelor-Studiengang Informatik WS2019/2020

Übungsblatt 9: Interfaces, Klassen und Objekte

Ausgabe: 11.12.19
Abgabe: 08.01.20 10:00 Uhr elektronisch mittels Git

Aufgabe 1

Überblick:
Teilaufgabe a beschreibt die Klasse `Song`, weil Sie in den folgenden Teilaufgaben `Song`s verwalten sollen.
Teilaufgabe b beschreibt eine (einfachere) Implementierung einer linearen Liste, die `Song`s aufnehmen kann.
Teilaufgabe c beschreibt eine schwierigere Implementierung einer linearen Liste, die ebenfalls `Song`s aufnehmen kann.
Teilaufgabe d beschreibt eine `main`-Methode, in der Songs eingelesen und in die gewählte Listenimplementierung eingefügt werden; ein Hauptmenü ermöglicht das Bearbeiten dieser Liste.
Bearbeiten Sie auf jeden Fall die Teilaufgaben a und d. Danach können Sie *eine* (beliebige) der beiden Teilaufgaben b oder c bearbeiten. Mit Teilaufgabe b können Sie maximal 8 Punkte erreichen, mit Teilaufgabe c maximal 10 Punkte. Wenn Sie beide Teilaufgaben abgeben wird nur eine davon bewertet!

Gegeben ist das folgende Interface:

```
package linkedArrayListOfSongs;

public interface SongList {

	/**
	 * Fuegt den neuen Song an das Ende der Liste an.
	 */
	void addLast(Song s);

	/**
	 * Loescht den Knoten an der angegebenen Position aus der Liste;
	 * dabei hat die erste Listenposition die Positionsnummer 0.
	 * @throws PRException wenn die Positionsangabe ungueltig ist.
	 */
	void remove(int position);

	/**
	 * Liefert die Anzahl der gespeicherten Songs.
	 */
	int size();

	/**
	 * Liefert den Song an der angegebenen Position der Liste;
	 * dabei hat die erste Listenposition die Positionsnummer 0.
	 * @throws PRException wenn die Positionsangabe ungueltig ist.
	 */
	Song get(int position);

	/**
	 * Entfernt alle Knoten aus der Liste
	 */
	void clear();
```

```
    /**
     * Liefert genau dann true, wenn mindestens ein Song mit dem
     * angegebenen Namen in der Liste enthalten ist, sonst false
     */
    boolean contains(String songName);

    /**
     * Liefert die erste Position in der Liste, an der ein Song
     * mit dem angegebenen Namen enthalten ist;
     * dabei hat die erste Listenposition die Positionsnummer 0.
     * Wird kein Song mit dem angegebenen Namen gefunden, liefert
     * die Methode -1 als Ergebnis.
     */
    int indexOf(String songName);

}
```

a) Erstellen Sie eine Klasse `Song`, um den Titel eines Musiksongs, den Namen des zugehörigen Albums sowie einen oder mehrere zugehörige Musiker/Bands zu speichern. Die Klasse soll einen geeigneten Konstruktor besitzen (dieser wird in der Teilaufgabe d beschrieben) und die folgenden Methoden zur Verfügung stellen:

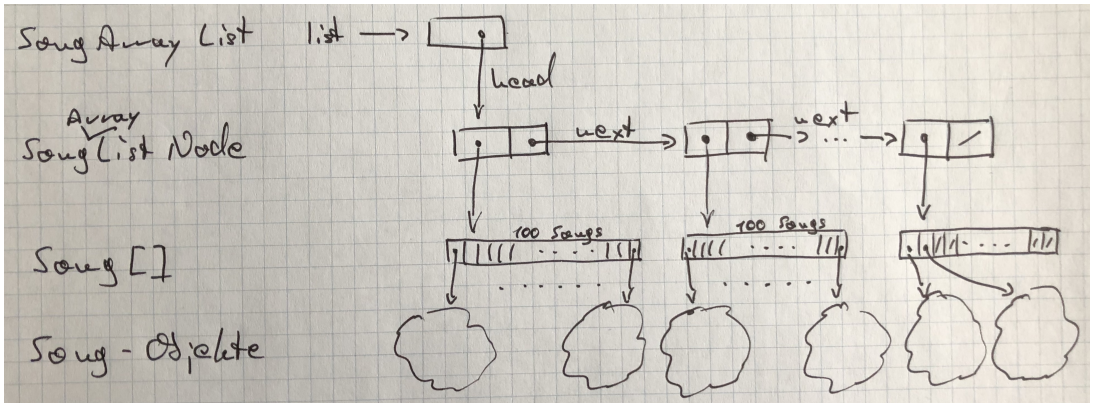
- `String getSongName()` Liefert den Namen des Songs
- `String getAlbumName()` Liefert den Namen des zum Song gehörigen Albums
- `String[] getArtists()` Liefert ein passend dimensioniertes Array mit einem oder mehreren Namen der zugehörigen Künstler oder Bands
- `String toString()` Liefert eine für Menschen gut lesbare Darstellung aller Daten des Songs

b) Sie sollen eine einfach verkettete Liste implementieren, in die `Song`-Objekte aufgenommen werden können. Im Einzelnen:

Erstellen Sie eine Klasse `SongLinkedList`, die das Interface `SongList` implementiert.
Die Klasse `SongLinkedList` soll als Hilfsklasse für die Darstellung der verketten Knoten eine Klasse `SongArrayListNode` benutzen, die jeweils einen `Song` und einen Verweis auf einen Folge-`SongArrayListNode` enthält. In dieser Klasse dürfen Sie folgende Methoden vereinbaren: `getNext`, `setNext` und `getSong`. Es gibt aber auch eine Lösung, in der diese Methoden nicht benötigt werden.

c) Implementieren Sie das Interface `SongList` als verkettete Array-Liste. Das ist wie folgt zu verstehen:

Es soll eine Klasse `SongArrayListNode` geben, die ein Array mit dem Element-Typ `Song` enthält, um maximal 100 Alben aufzunehmen. (Vermutlich benötigen Sie zu Ihrem Array auch die Information, wie viele `Song`s aktuell im Array eingetragen sind.) Außerdem enthält die Klasse `SongArrayListNode` einen Verweis auf einen Folge-`SongArrayListNode`, der angehängt wird, wenn das Array nicht mehr ausreicht, um alle Alben zu speichern. Damit stellen die Objekte der Klasse `SongArrayListNode` die Knoten einer verketteten Liste dar. Beim Anfügen eines neuen `Song`s wird aber nicht jedes Mal ein neuer Knoten erzeugt und angehängt, sondern erst dann, wenn im letzten Knoten alle Array-Positionen mit `Song`s belegt sind. Beispiel-Darstellung:



Das Interface `SongList` soll von einer Klasse `SongArrayList` implementiert werden, die (wie bei einfach verketteten Listen) einen Verweis auf den ersten `SongArrayListNode` enthält und so quasi den Zugang zur Liste ermöglicht.
Anmerkung:
Die `ArraySongList` ist flexibler als ein `Song[]` in Java, weil Sie sich nicht darum kümmern müssen, wann das Array nicht mehr genug Platz für neue Songs bietet: Sie rufen einfach nur `addLast` auf. Gleichzeitig ist sie performanter als eine "normale" verkettete Liste, bei der für jedes neue Element ein neuer Listenknoten erzeugt werden muss.

d) Erstellen Sie eine Klasse `Start` mit einer `main`-Methode wie üblich, um Ihre Listen-Implementierung zu testen. Ihre Klasse sollte ein Menü anbieten, um die Listen-Methoden mit verschiedenen Parametern testen zu können.

Es soll auch die Möglichkeit geben, mit folgenden Testdaten zu arbeiten:
Die Datei "songs.txt" enthält eine Menge von Song-Testdaten, die in jeder Zeile durch Semikolon getrennt jeweils den Titel eines Songs, den Namen des zugehörigen Albums sowie einen oder mehrere Musiker enthalten. Sie können die Datei ganz einfach mit der `MakeItSimple`-Methode `readStringArray` in ein Array von Strings lesen (jede Zeile aus der Datei ergibt einen String). Den Dateinamen geben Sie relativ zu Ihrem Projekt an.
Versehen Sie Ihre Klasse `Song` mit einem Konstruktor, der einen solchen String als Parameter akzeptiert, analysiert und für die Verwendung in der Klasse aufteilt. So können Sie in `main` einfach die Datei lesen, dann nacheinander aus jeder Zeile ein `Song`-Objekt erzeugen und es in die Liste einfügen. Der Konstruktor soll für alle fehlerhaften Strings eine `PRException` mit einer geeigneten Fehlermeldung auslösen. (Die Datei enthält keine fehlerhaften Zeilen.)
Für den Konstruktor dürfen Sie die Methode `split` aus der Klasse `String` benutzen; Sie müssen aber erklären können, was die Methode tut.

Allgemeines

- Legen Sie für die Bearbeitung dieses Übungsblattes ein Paket namens `uebung09` an, in dem Sie Ihre Klassen anlegen.
- In Moodle finden Sie Testklassen `...1stTest`, mit der Sie prüfen können, ob Ihre Methoden korrekt definiert sind und zumindest die einfachsten Eingaben korrekt behandelt werden.
- Erlaubt sind `MakeItSimple`-Funktionen (keine nicht besprochene Funktionalität aus der Java-Standard-Bibliothek) und das bisher erworbene Wissen aus den PR1-Vorlesungen. Sie müssen alle(!) von Ihnen verwendeten Konstrukte der Sprache sowie alle verwendeten Methoden, die nicht aus der Hilfsbibliothek `MakeItSimple` stammen, gut erklären und nötigenfalls im Testat selbst programmieren können!
- Sie geben ab, indem Sie vor Ende der Abgabefrist Ihr Projekt mit den lauffähigen Programmen des Übungsblattes in das Repository auf *wilma* pushen. Achten Sie darauf, ob Sie die Kontroll-E-Mail bekommen! Die letzte hochgeladene Version Ihres Projekts wird gewertet. Andere Abgaben, ob elektronisch oder auf Papier, zählen als **nicht abgegeben**!
- Die Aufgaben sind in Eclipse zu bearbeiten und beim Testat vorzuführen.
- Nutzen Sie die Übungsstunde, um eventuelle Fragen zur Vorlesung, zur Übung oder allgemein zum Studium zu stellen! Die Betreuer sind da, um Ihnen solche Fragen zu beantworten und Ihnen bei Bedarf zu helfen!