Hochschule Darmstadt
Prof. Dr. Benjamin Meyer

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

27. April 2021

Programmieren, Algorithmen und Datenstrukturen II, SoSe 2021
Aufgabenblatt 2

Die Aufgaben dieses Blattes bauen auf dem im letzten Praktikum entstandenen Framework auf. Sollten Sie daher noch Anmerkungen aus der Abnahme offen haben, nehmen Sie sich zuerst die Zeit, Ihr Programm auf den aktuellen Stand zu bringen.

2.1 Menüsteuerung

Als letzte Wiederholung von PAD1-Kenntnissen soll zuallererst eine Benutzersteuerung für das Streaming-Programm der letzten Aufgabe erstellt werden. Bei Programmstart soll der Benutzer daher per Tastatureingabe aus dem folgenden Menü wählen können:

```
(1) Ausgeben aller Videos (nach Durchschnittsbewertung sortiert)
(2) Abspielen eines Videos
(3) Hinzufuegen einer Bewertung zu einem Video
(4) Video-Datenbank aus File laden
(5) Video-Datenbank speichern
(6) Programm beenden
```

Kapseln Sie jeden Menüeintrag in eine entsprechende eigene Funktion. Die Inhalte sind aus dem letzten Aufgabenblatt bereits weitestgehend implementiert. Ergänzen Sie, was noch fehlt. Nummerieren Sie zudem die Videos bei der Auflistung und sprechen Sie einzelne Videos dann über die jeweilige Nummer an.
Achten Sie zudem auf inkorrekte Eingaben des Nutzers. Mittlerweile können Sie sämtliche ungültigen Eingaben abfangen (z.B. auch einen String, wenn ein Integer gefordert war). Testen Sie hierfür die Funktionen des `istream`.

2.2 Fehlerbehandlung

Die Funktionsaufrufe (4) und (5) aus dem Menü benötigen eine manuelle Eingabe des gewünschten Dateipfades. Sollte die gewünschte Datei nicht geöffnet/geschrieben werden können, werfen Sie einen Fehler in der entsprechenden Methode.
Die Menü-Routine soll nun entsprechend den potentiell entstehenden Fehler abfangen und sinnvoll reagieren können: Neben der Ausgabe einer Fehlermeldung und der erneuten Aufforderung zur Eingabe eines Dateipfades soll ein Fehlerzähler speichern, wie oft der Dateiname falsch eingegeben wurde. Bei der dritten falschen Eingabe (hintereinander) wird statt dessen ein Standardparameter als Dateipfad genutzt, welcher dem Programm beim Start als Argument mitgeteilt wurde.

2.3 Vererbung

Bisher verwaltet die Datenbank nur Einträge vom Typ `Movie`. Dies soll nun geändert und um die Speicherung von Serien ergänzt werden.
Damit dies funktioniert, gehen Sie wie folgt vor: Benennen Sie die `Movie`-Klasse um in das allgemeinere `MediaFile`. Üben Sie sich hierfür im Umgang mit der Refactor-Funktion Ihrer IDE, damit beispielsweise auch der Vektor der `Database` nun `Mediafiles` speichert. Nun können Sie zwei neue Klassen erstellen, die beide vom Mediafile erben und somit die bis jetzt implementierte Funktionalität übernehmen: Das `Movie` und die `Series`.

Eine `Serie` zeichnet sich vor allem dadurch aus, dass sie eine Anzahl an Folgen besitzt. Speichern Sie sich zusätzlich dazu auch die aktuelle Folge, die als nächstes geschaut werden müsste (und aktualisieren Sie jeweils, wenn eine Folge abgespielt wird - ist die Serie durch, wird von vorne begonnen). Die übergeordnete `Dauer` der Oberklasse soll hierbei als (durchschnittliche) Dauer einer Episode interpretiert werden.
Fügen Sie manuell in der `main()` einige Serien der Datenbank hinzu. Sie werden merken, Sie müssen hierfür den Dateityp des Vektors der `Database` nicht ändern. Allerdings sollten Sie, falls noch nicht geschehen, im Vektor der Datenbank nur Pointer auf die einzelnen Mediafiles halten, nicht die Objekte selbst. Achten Sie hierfür darauf, in Ihren Methoden lokale Objekte per `new` zu erzeugen, ansonsten reicht die Sichtbarkeit der Objekte nur bis zum Methodenende!

Das `Movie` wird stattdessen durch ein Erscheinungsjahr erweitert (inklusive Möglichkeiten zum Auslesen und Setzen des Wertes). Die ursprüngliche Sammlung an Filmen (also das Textfile) soll weiterhin `Movies` gespeichert haben. Sie können entweder das Textfile sowie die Auslese-Methode um das Erscheinungsjahr erweitern, oder Sie generieren zufällig einen Wert für jedes eingelesene `Movie`.
Um die beiden neuen Klassen mit zusätzlichem Leben zu füllen, schreiben Sie für beide Klassen jeweils eine neue Abspielmethode (die kenntlich macht, welches Dateiformat gerade abgespielt wird).
Erweitern Sie danach die Funktionen des Eingabe-Menüs, um bei Option 2 auch mit Serien agieren zu können. Was passiert, wenn Sie nun einen Film oder eine Serie abspielen wollen? Wie können Sie das Programmverhalten ändern?
Implementieren Sie zu guter Letzt für die Klasse `Movie` eine Methode, die als Parameter ein zweites Movie als Referenz übergeben bekommt und entscheidet, welcher Film älter ist. Erzeugen Sie manuell ein Movie und testen Sie die Methode mit einem beliebigen File aus der `Database`, was müssen Sie tun, damit das funktioniert?

2.4 Bonusaufgabe: Erstellen einer grafischen Benutzeroberfläche

Erweitern Sie das Eingabemenü um einen 7. Punkt: Umschalten auf grafische Benutzeroberfläche. Diese Benutzeroberfläche gilt es jetzt in Qt zu erstellen! Sie haben hierfür keine expliziten Vorgaben, halten Sie sich grob an das inhaltliche Design der Menüführung. Achten Sie weiterhin auf Falscheingaben des Benutzers.
Tipp: Für ein einfaches Feedback an den Nutzer können Sie ein QLabel platzieren und den initialen Text einfach löschen. Falls es zu einer Falscheingabe kommt, können Sie dies dem Nutzer mitteilen, indem Sie den Text des QLabels anpassen. Beispiel für das QLabel `label`:

```
ui->label->setText("neuer Text");
```

Alternativ können Sie auch separate Warnfenster erzeugen.

2.5 Bonusaufgabe: Anpassen der Dateiverwaltung

Das Lesen und Schreiben von Dateien ist die Grundlage vieler Programme, seien es Savegames, Datenbanken, Programmeinstellungen oder Ähnliches. Passen Sie daher zur Übung Ihr Dateiformat an, um bei dem Schreiben den Unterschied zwischen Serien und Filmen kenntlich zu machen. Im zweiten Schritt muss danach natürlich noch die Lese-Routine angepasst werden, um das neue Dateiformat zu verstehen. Zum Abschluss können Sie versuchen, die Dateien in binärer Form zu schreiben und zu lesen.