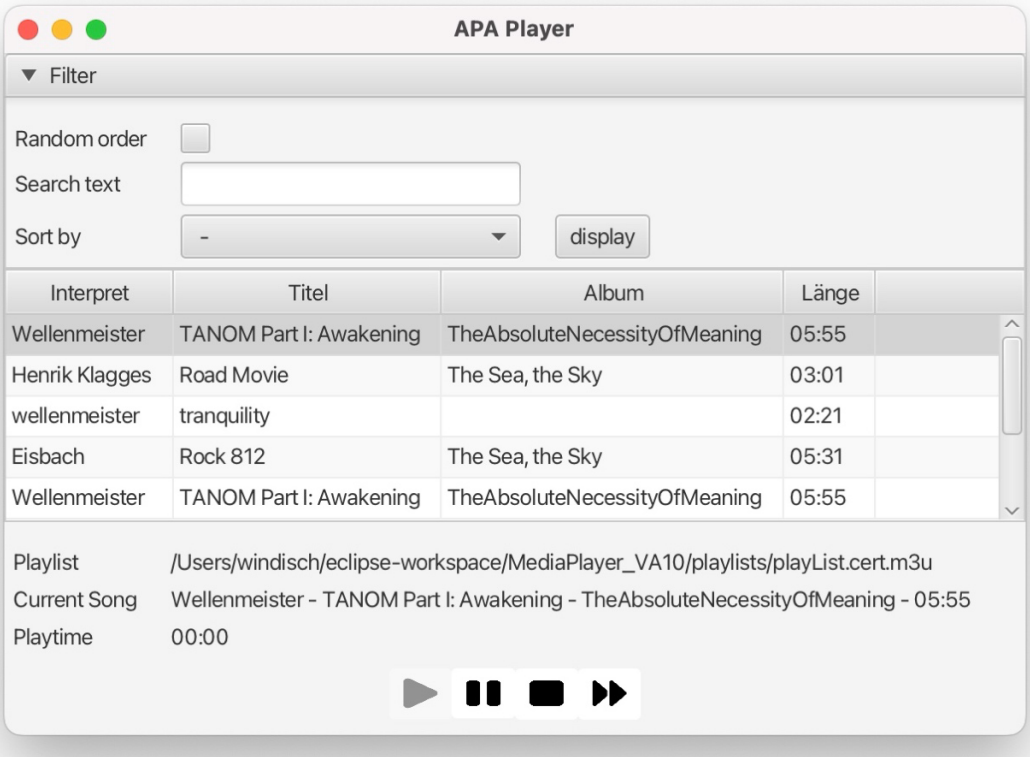
Technische Hochschule Ingolstadt

Vorführaufgabe 6: Zerlegen von Dateipfaden

Ziel des Praktikums ist die Entwicklung eines Mediaplayers zum Abspielen von Audiodateien. Die folgende Abbildung zeigt das finale User Interface (UI) der Anwendung, welches aus dem Player mit einer such- und sortierbaren Abspielliste und den Schaltflächen zur Steuerung besteht.



Vorbereitung: Aufsetzen der Entwicklungsumgebung

Legen Sie in Ihrer IDE ein neues Projekt an, z.B. „MediaPlayer_VA06“.

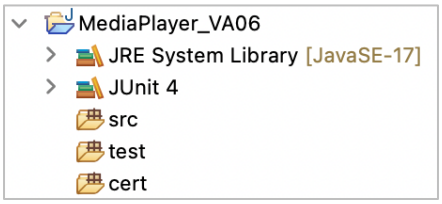
Hinweis: Für das Weitere wird angenommen, dass Sie Eclipse für Java (siehe https://www.eclipse.org/downloads/) als IDE (Integrated Development Environment) nutzen. Selbstverständlich sind Sie frei in der Wahl ihrer Entwicklungsumgebung, den optimalen Support erhalten Sie im Rahmen des Praktikums jedoch nur für Eclipse.

Achtung: legen Sie keine Module-Info an! (in Eclipse ist das eine Option im ersten Fenster des Assistenten)

Sorgen Sie durch geeignete Einstellungen in Ihrer IDE dafür, dass:

- ein Source Folder[1] `src` zur Speicherung der Java-Quellen vorhanden ist
- ein Source Folder[2] `test` zur Speicherung eigener JUnit-Tests angelegt wird
- ein Source Folder `cert` zur Speicherung der Abnahme-Tests angelegt wird
- JUnit der Version 4 als „Dependency“ für Tests im Projekt eingebunden wird[3]

In Eclipse sollte ihr Projekt nun so aussehen:



Grundlegendes zu Pfad- und Dateinamen

Der im Rahmen des Praktikums entstehende Audioplayer kann Audiodaten in Form von Audiodateien (Audiofiles) verarbeiten, die über ein Dateisystem zugreifbar sind. Daher beginnen wir die Implementierung des Audioplayers mit dem Erstellen der Klasse `AudioFile`.

Jedes Objekt der Klasse `AudioFile` bezieht sich auf eine einzelne Audiodatei. Der *Pfadname* dieser Datei besteht aus dem optionalen Pfad und dem Dateinamen. Wird ein Pfad angegeben, so kann dies eine absolute oder eine relative Angabe sein. Es kann also der vollständige Pfad, ein relativer Pfad oder auch gar kein Pfad mit angegeben sein.

Hinweis: Das Zeichen `'␣'` steht hier, ebenso wie im Rest der Angabe, jeweils für ein Leerzeichen.

Hinweis: Bitte beachten Sie, dass der Pfadseparator, also das Trennzeichen für die einzelnen Verzeichnisse, je nach Betriebssystem unterschiedlich ist. Unter Unix/MacOS ist es das Zeichen `'/'`, unter Windows das Zeichen `'\'`.

Achtung: Um das `'\'`-Zeichen in Java zu nutzen, muss `'\\'` geschrieben werden, da `'\'` ein Steuerzeichen ist.

Beispiele für Unix/Linux/Mac Systeme:

/home/meier/Musik/Falco␣–␣Rock␣Me␣Amadeus.mp3 (absoluter Pfad, beginnt mit /)
../musik/Falco␣–␣Rock␣Me␣Amadeus.mp3 (relativer Pfad)

---

[1] In Eclipse existiert dieser bereits

[2] In Eclipse: Rechtsklick auf Projekt → New → Source Folder; in einem Source Folder enthaltene Java-Quellen werden automatisch übersetzt.

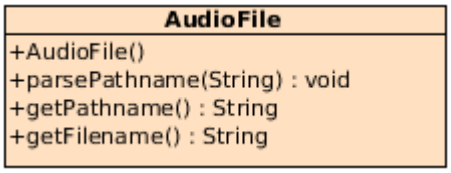
[3] In Eclipse: Rechtsklick auf Projekt → Build Path → Add Libraries → JUnit → JUnit library version: JUnit 4

Falco_–_Rock_Me_Amadeus.mp3	(ohne Pfad)

Beispiele für Windows Systeme:

D:\Daten\Musik\Falco_–_Rock_Me_Amadeus.mp	(absoluter Pfad, beginnt mit Laufwerksangabe)
..\Musik\Falco_–_Rock_Me_Amadeus.mp3	(relativer Pfad)
Falco_–_Rock_Me_Amadeus.mp3	(ohne Pfad)

Teilaufgabe a) Anlegen der Klasse AudioFile



Legen Sie eine Klasse `AudioFile` an, die die vier im Klassendiagramm gezeigten Methoden zur Verfügung stellt.

Die Aufgabe der Klasse `AudioFile` ist die Zerlegung und Betriebssystem-abhängige Normalisierung eines gegebenen Pfadnamens. Die Ergebnisse sollen in Attributen abgelegt werden, die außerhalb des `AudioFile`-Objekts durch Getter-Methoden (z.B. `getPathname()`) gelesen werden können.

Hinweis: Eine Methode `getPathname()` deutet in der Regel auf ein Attribut `pathname` hin, analog gilt dies für die Methode `getFilename()` und andere.

Teilaufgabe b) Zerlegen von Pfadnamen

Implementieren Sie die beiden parse-Methoden gemäß den nachfolgenden Vorgaben.

```
parsePathname(String path): void
```

Der Parameter `path` beinhaltet den Pfadnamen der Audiodatei. Die Methode soll den Parameter analysieren und als Ergebnis der Analyse die Attribute `pathname` und `filename` belegen. Im Einzelnen sollen folgende Anforderungen erfüllt werden, Beispiele finden sich in der Tabelle:

- **Laufwerksangabe behandeln**
  Falls das Programm nicht unter Windows läuft, soll eine Laufwerksangabe am Anfang (z.B. „C:\...“) korrigiert werden.
- **Pfadseparatoren normalisieren**
  In `path` enthaltene Pfadseparatoren sollen in die für das aktuelle Betriebssystem gültigen Separatoren umgewandelt werden. Zudem sollen Folgen von Pfadseparatoren (z.B. "medien///song.mp3") zu einem zusammenfasst werden (z.B. "medien/song.mp3").
- **Speichern der Ergebnisse in Attributen**
  Das Ergebnis der oben genannten Operationen soll in `pathname` gespeichert werden. Gibt es Pfadtrenner in `path`, so ergibt sich der Wert für `filename` aus der Zeichenfolge rechts vom letzten Pfadtrenner. Andernfalls erhält `filename` den selben Wert wie `pathname`.

Die nachfolgende Tabelle enthält Beispiele, die die obigen Anforderungen verdeutlichen und ergänzen.

Argument für parsePathname()	Resultat getPathname()	Resultat getFilename()
Leerer String bzw. nur Leerzeichen oder Tabs	Leerer String	Leerer String
file.mp3	file.mp3	file.mp3
␣␣/my-tmp/file.mp3	Linux: ␣␣/my-tmp/file.mp3 Windows: ␣␣\my-tmp\file.mp3	file.mp3
//my-tmp////part1//file.mp3/	Linux: /my-tmp/part1/file.mp3/ Windows: \my-tmp\part1\file.mp3\	Leerer String
d:\\\\part1///file.mp3	Linux: **/d/**part1/file.mp3 Windows: **d:**\part1\file.mp3	file.mp3
-	-	-
␣-␣	-	-

Zur Entwicklung der Methode `parsePathname()` können Sie das bereitgestellte **Testbett in der Datei TestParsePathname.java** nutzen. Die Tests überprüfen die in der Tabelle aufgeführten Beispiele und noch einige weitere.

Testen der Klasse unter Windows und Linux

Um sicherstellen zu können, dass ihre Anwendung auf Windows und auch Linux korrekt ausgeführt wird, nutzen wir die System-Eigenschaften (engl. *system properties*) "os.name" (Betriebssystem-Name) und "file.separator" (Pfadtrenner) sowohl in der Klasse AudioFile, also auch in den JUnit-Tests. Vor der Testausführung werden die Properties durch Java gesetzt. Die benötigten Codezeilen sind bereits vorhanden, bitte machen Sie sich mit den Tests vertraut und spielen Sie diese für beide Betriebssysteme durch.

Weiterhin ist es an manchen Stellen wichtig zu wissen, ob das Programm unter Windows ausgeführt wird. Hierzu können Sie die folgende Hilfsmethode nutzen:

```
private boolean isWindows() {
    return System.getProperty("os.name").toLowerCase()
        .indexOf("win") >= 0;
}
```

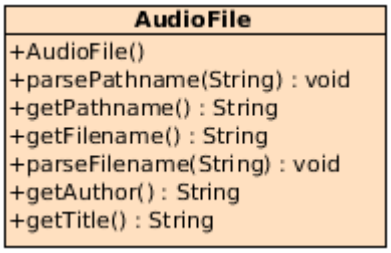
Teilaufgabe c) Zerlegen von Dateinamen

In dieser Teilaufgabe soll die Methode `parseFilename(String filename)` zur Analyse eines übergebenen Dateinamens entwickelt werden. Häufig finden sich bei Audiodateien Dateinamen, die gemäß dem folgenden Schema aufgebaut sind:

```
Autor␣-␣Titel.Endung
```

Autor (engl. *author*) gibt an, von wem das Lied mit dem Titel (engl. *title*) stammt, Endung (engl. *extension*) nimmt typischerweise die Werte "mp3", "mp4", "wav" etc. an.

`parseFilename(String filename)` soll den übergebenen Dateinamen in die beiden oben genannten Bestandteile zerlegen und das Ergebnis der Zerlegung in den neuen Attributen `author` und `title` speichern. Getter für Autor und Titel sollen den Zugriff ermöglichen. Das rechts dargestellte UML-Klassendiagramm zeigt die erweiterte Klasse `AudioFile`.



Die nachfolgende Tabelle zeigt anhand von Beispielen, wie die neue `parse`-Methode funktionieren soll.

Argument für parseFilename()	Resultat getAuthor()	Resultat getTitle()
◡Falco◡◡-◡◡Rock◡me◡◡ ◡◡Amadeus◡.mp3◡◡	Falco	Rock◡me◡◡◡◡Amadeus
Frankie◡Goes◡To◡Hollywood ◡-◡The◡Power◡Of◡Love.ogg	Frankie◡Goes◡To◡Hollywood	The◡Power◡Of◡Love
audiofile.aux	Leerer String	audiofile
◡◡◡A.U.T.O.R◡◡◡- ◡◡T.I.T.E.L◡◡.EXTENSION	A.U.T.O.R	T.I.T.E.L
Hans-Georg◡Sonstwas◡-◡ Blue-eyed◡boy-friend.mp3	Hans-Georg◡Sonstwas	Blue-eyed◡boy-friend
.mp3	Leerer String	Leerer String
Falco◡- ◡Rock◡me◡Amadeus.	Falco	Rock◡me◡Amadeus
-	Leerer String	-

Zur Entwicklung der neuen Methoden können Sie die Tests in der **bereitgestellten Datei TestParseFilename.java** nutzen.

Teilaufgabe d) Hauptkonstruktor für AudioFile

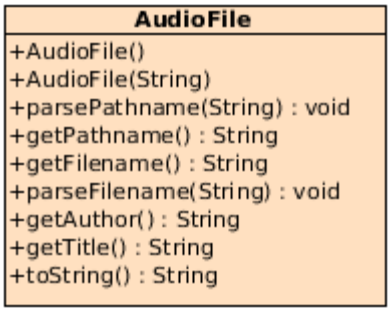
Bisher wurden die Parse-Methoden an einem mit dem Default-Konstruktor erzeugten Objekt aufgerufen. Ziel ist jedoch, bereits bei der Erzeugung eines AudioFile-Objekts den Pfad mitzugeben und die Analysen durchzuführen. Im Ergebnis sind alle vier Attribute (`pathname`, `filename`, `author`, `title`) nach der Erzeugung belegt.

Implementieren Sie einen parametrisierten Konstruktor `AudioFile(String path)`, der als Argument den Pfadnamen einer Audiodatei erwartet, dies wird der Hauptkonstruktor. Der Default-Konstruktor soll weiterhin verfügbar sein.

Teilaufgabe e) Stringrepräsentation für AudioFile-Objekte

Für die Ausgabe von Objekten der Klasse `AudioFile`, überschreiben wir in der Klasse `AudioFile` die Methode `toString()`. Die Methode soll einen String nach den folgenden Vorgaben liefern:

- falls die Methode `getAuthor()` einen leeren String liefert, soll nur der Titel zurückgegeben werden.
- Ansonsten soll die Methode Interpret und Titel getrennt durch "‿-‿" zurückgegeben.



Zur Entwicklung der `toString`-Methode können Sie die Tests in der **bereitgestellten Datei TestToString.java** nutzen. Bitte führen Sie auch diese Tests für beide Betriebssysteme durch (siehe Methode `setUp()`).

Nach der Implementierung der `toString`-Methode ergibt sich das angezeigte Klassendiagramm.

Hinweise zur Abnahme Ihrer Implementierung der Vorführaufgabe 06

Laden Sie alle zum Aufgabenblatt gehörigen Abnahme-Tests herunter und speichern Sie diese im Source-Folder cert Ihres Projekts (siehe Abschnitt „Vorbereitung“ am Anfang dieses Aufgabenblatts). Dann führen Sie die Abnahme-Tests in Eclipse aus.

Achtung: testen Sie die Lösung mit beiden Betriebssystemen, indem Sie in der `setUp`-Methode der AudioFileTest.java die entsprechende Zeile als Kommentarzeile setzen:

```
// sep = Utils.emulateWindows();
```

… und die gewünschte Emulation aktivieren:

```
sep = Utils.emulateLinux();
```

Wenn alle Tests ohne Fehler für beide Betriebssysteme durchlaufen, mailen Sie bitte ihre `AudioFile.java`-Datei als Dateianhang an den APA-Server.