EasyPlay

Easyplay

Dokumentation easyplay

Oliver köchli, samuel kurmann

[Jahr]

Inhalt

[2 Zeitplanung 2](#_Toc420660557)

[3 Klassen erstellen 2](#_Toc420660558)

[3.1 Klasse Lied 2](#_Toc420660559)

[3.2 Klasse Bibliothek 3](#_Toc420660560)

[3.3 Klasse Warteliste 3](#_Toc420660561)

[3.4 Klasse Playlist 4](#_Toc420660562)

[4 GUI 5](#_Toc420660563)

[4.1 GUI erstellen 5](#_Toc420660564)

[5 Funktionen 7](#_Toc420660565)

[5.1 Bibliothek erstellen 7](#_Toc420660566)

[5.2 Playlist erstellen 7](#_Toc420660567)

[5.3 Warteliste erstellen 7](#_Toc420660568)

[5.4 Daten Speichern 7](#_Toc420660569)

[5.5 Daten Laden 7](#_Toc420660570)

[5.6 Titel abspielen 8](#_Toc420660571)

[5.7 Titel wiederholen 9](#_Toc420660572)

[5.8 Playlist wiederholen 9](#_Toc420660573)

[5.9 Titel automatisch wiedergeben 9](#_Toc420660574)

[5.10 Funktion in GUI implementieren 10](#_Toc420660575)

[6 Test 11](#_Toc420660576)

[6.1 Testcase 001: Erstmaliger Start 11](#_Toc420660577)

[6.2 Testcase 002: Normaler Start 11](#_Toc420660578)

[6.3 Testcase 003: Playliste erstellen 12](#_Toc420660579)

[6.4 Testcase 004: Lied abspielen 12](#_Toc420660580)

[6.5 Testcase 005: Warteliste 13](#_Toc420660581)

[6.6 Testcase 006: Lautstärke 13](#_Toc420660582)

[6.7 Testcase 007: Hinzufügen 14](#_Toc420660583)

[6.8 Testcase 008: Beenden 14](#_Toc420660584)

[6.9 Testcase 009: Warteliste leeren 15](#_Toc420660585)

# Zeitplanung

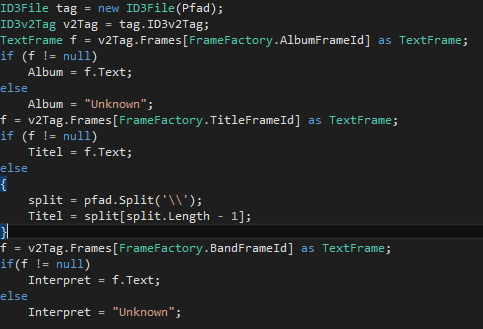
Bevor mit dem Programmieren des MP3-Players beginnen konnten, mussten wir eine Zeitplanung erstellen, nach welcher wir uns orientieren konnten.

# Klassen erstellen

Wir haben begonnen mit dem erstellen aller Klassen, damit wir diese gleich von Anfang an haben und nicht nach und nach erstellen müssen.

## Klasse Lied

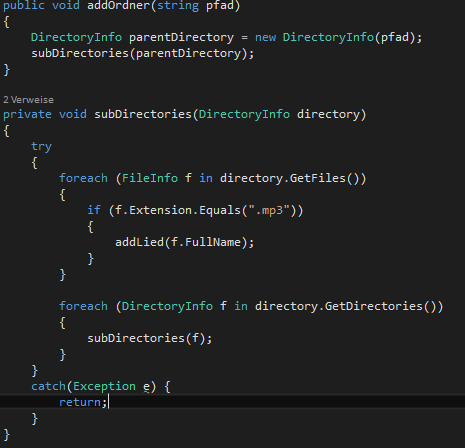
Als erstes haben wir die Klasse Lied implementiert, da diese Klasse das Herzstück des MP3-Players ist. Dabei hatten wir das Problem, dass wir die ID3-Tag Daten aus der MP3-Datei lesen mussten. Damit wir dies tun konnten, mussten wir eine zusätzliche Bibliothek einbinden, diese Bibliothek stellt die benötigten Methoden zum Auslesen zur Verfügung.



Als wir dieses Problem gelöst hatten, gab es in der Klasse Lied keine weiteren Probleme.

## Klasse Bibliothek

Als wir die Klasse Lied fertig implementiert hatten, konnten wir beginnen die Klasse Bibliothek zu implementieren. In der Klasse Bibliothek geht es in erster Linie darum, die Lieder des MP3-Players in einer Liste zu speichern und diese zur Verfügung zu stellen. In dieser Klasse hatten wir das Problem, dass wir einen kompletten Ordner und dessen Unterordner nach MP3-Dateien zu durchsuchen und diese in das Programm ein zu lesen.



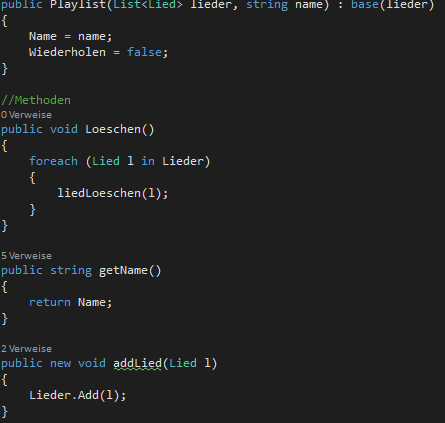
Als wir dieses Problem beseitigen konnten, gab es keine weiteren Probleme mit der Klasse Bibliothek.

## Klasse Warteliste

Nachdem wir die Klasse Bibliothek implementiert hatten, konnten wir beginnen die Klasse Warteliste zu implementieren. In dieser Klasse hatten wir keine Probleme, da wir keine neuen Funktionen implementieren mussten.

## Klasse Playlist

Als letzte Klasse mussten wir nur noch die Klasse Playlist erstellen. In der Klasse Playlist hatten wir keine grossen Probleme, wir mussten nur daran denken, dass wir Lieder, welche wir in die Playlist einfügen nicht als neue Objekte einfügen, sondern die Objekte nehmen, welche wir bereits in der Bibliothek haben.

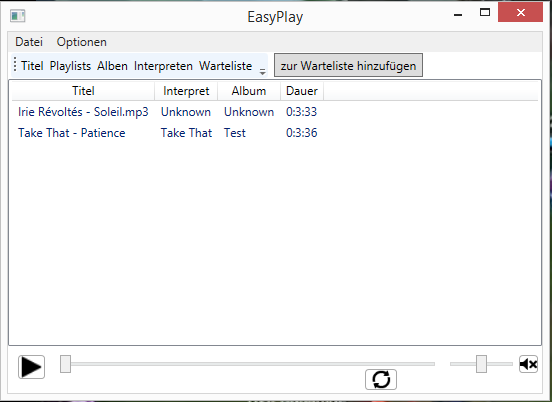


# GUI

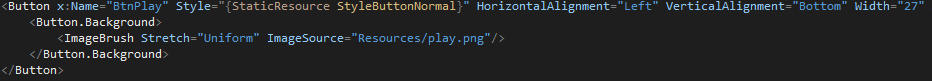
Parallel zum Klassen erstellen haben wir begonnen unser GUI zu erstellen.

## GUI erstellen

Als erstes haben wir unser GUI erstellt. Zum Wechseln zwischen Playlist und der Bibliothek, usw. haben wir uns entschieden eine Toolbar zu brauchen. Für die Funktionen um Lieder, bzw. Ordner hinzuzufügen, haben wir uns entschieden eine MenuBar zu benützen. Die Darstellung der Titel haben wir mit ListViews gemacht, da dies unserer Meinung nach die einfachste Lösung war. Wir haben uns entschieden den Fortschritt eines Titels mit Hilfe eines Sliders darzustellen, auch die Laustärkeregelung haben wir mit einem Slider geregelt.



Für spezielle Funktionen, welche nicht immer zur Verfügung stehen sollen, haben wir uns entschieden Schaltfläche hinter der Toolbar darzustellen. Dabei hatten wir jedoch ein Problem mit den Schaltflächen, welchen wir ein Bild als Inhalt gegeben haben. Bei diesen Schaltflächen wurde das Bild durch den MouseOverEffekt verdeckt, wenn der Benutzer mit dem Mauszeiger über die Schaltfläche fährt. Dieses Problem konnten wir nur lösen, indem wir den Standard Style der Schaltflächen überschreiben. Damit wir dies tun konnten mussten wir eine Fenster Ressourcen definieren, welche wir dann den entsprechenden Schaltflächen zuweisen konnten.



# Funktionen

Nachdem wir das Layout erstellt hatten, mussten wir nun die Funktionen implementieren.

## Bibliothek erstellen

Als erstes haben wir uns entschieden die Funktion zu implementieren, mit welcher wir die Bibliothek erstellen und einen Ordner, welcher vom Benutzer definiert wurde, einlesen und die darin enthaltenen MP3-Dateien einzulesen. Diese Funktion soll jedoch nur ausgeführt werden, falls noch kein Save-File vorhanden ist. Dabei hatten wir keine Probleme, da wir dem Benutzer nur die Möglichkeit geben mussten ein Ordner auszuwählen und den Pfad des gewählten Ordners dem Konstruktor der Klasse Bibliothek mitgeben. Um dem Benutzer die Auswahl Möglichkeit zu geben, haben wir uns für den FolderBrowserDialog entschieden.

## Playlist erstellen

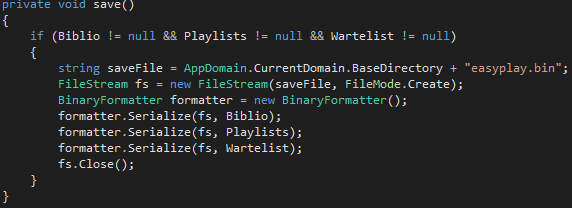
Da wir nun die Möglichkeit hatten eine Bibliothek zu erstellen, konnten wir mit der Funktion beginnen. Beim Erstellen der Playlist mussten wir besonders darauf achten, dass der Benutzer nicht die Möglichkeit hat zwei Playlists mit dem gleichen Namen zu erstellen, da der Name eindeutig sein muss. Beim Erstellen der Playlist hatten wir keine Probleme, da alles komplizierte in den Klassen abgehandelt wird.

## Warteliste erstellen

Nachdem wir die Funktion zum Erstellen von Playlists implementiert hatten, war das implementieren der Funktion um die Warteliste zu erstellen keine grosse Sache mehr, da es sich bei der Warteliste im Grunde um eine Art Playlist handelt.

## Daten Speichern

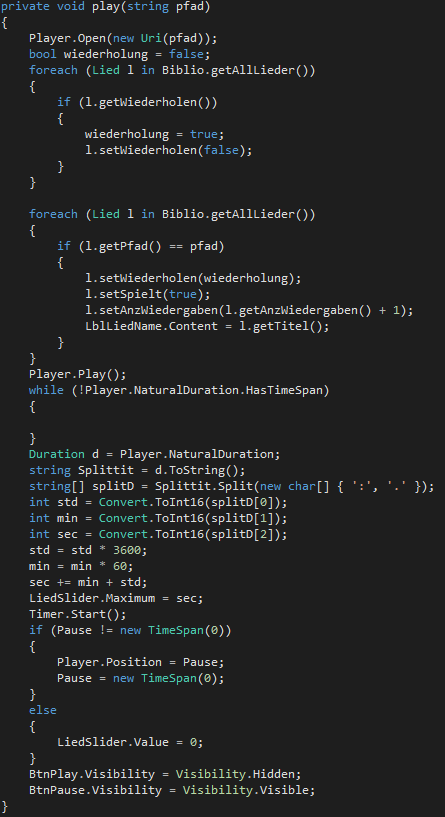
Da wir nun alle Funktionen zum erstellen implementiert hatten, mussten wir noch die erstellen Objekte speichern können. Wir haben uns im Voraus entschieden das Speichern der Objekte in einem File zu realisieren. Dabei haben wir uns entschieden, die Daten in ein .bin File zu schreiben. Dazu mussten wir die Klassen, welche wir speichern Serialisierbar zumachen. Um die Daten dann zu serialisieren haben wir den BinaryFormater verwendet. Beim Speichern der Daten hatten wir keine Probleme, wir mussten nur darauf achten, dass die Variablen, welche wir speichern ein Objekt beinhalten.



## Daten Laden

Nachdem wir nun die Möglichkeit haben die Daten zu speichern, mussten wir natürlich noch eine Funktion implementieren, welche die Daten wieder aus der Datei lesen kann. Dazu mussten wir natürlich wieder den BinaryFormater benützen, um die Objekte wieder in die Variabeln zu speichern. Beim Laden der Daten hatten wir keine Probleme.

## Titel abspielen

Da wir nun schon ziemlich alles erstellt haben und alle benötigten Funktionen hatten, konnten wir uns der Funktion zum Abspielen eines Titels widmen. Beim Abspielen eines Titels, mussten wir darauf achten, ob der Titel bereits am Spielen war, sprich ob er pausiert wurde. Falls der Titel Pausiert war, soll dieser an der aktuellen Stelle weiter gehen und nicht wieder von vorne beginnen. Zusätzlich mussten wir diverse andere Sachen beachten, beispielsweise mussten wir überprüfen ob ein Titel Wiederholt wird. Falls dies der Fall ist, soll der nächste Titel auch wieder holt werden. Desweitern muss der Slider auf die Länge des Titels eingestellt werden, damit dieser mit dem Fortschritt des Titels übereinstimmt. Im gleichen Schritt haben wir gleich noch das Pausieren eines Titels implementiert.

## Titel wiederholen

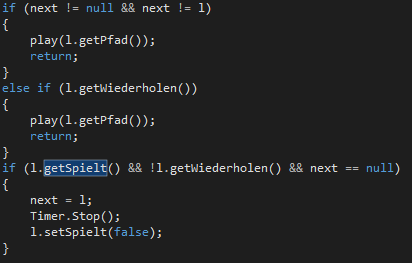
Nachdem wir nun Titel abspielen konnten, haben wir begonnen die Funktion zu implementieren, mit welcher es möglich ist einen einzelnen Titel zu wiederholen. Bei dieser Funktion gab es keine grossen Probleme.

## Playlist wiederholen

Da wir nun die einzelnen Titel wiederholen konnten, mussten wir noch die Funktion zum wiederholen von Playlists implementieren. Bei dieser Funktion hatten wir keine Probleme, da wir nur ein Flag im Objekt der aktuellen Playlist setzten mussten.

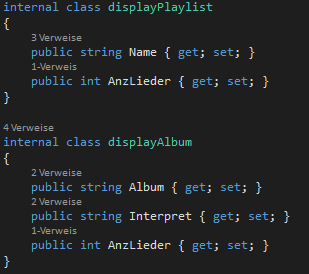
## Titel automatisch wiedergeben

Nun mussten wir noch implementieren, dass nach Ende eines Titels der nächste Titel automatisch abgespielt wird. Dies war ziemlich kompliziert, da wir darauf achten mussten ob nun die Titel aus einer Playlist, der Bibliothek oder aus der Warteliste wiedergegeben wurden. Nachdem wir dieses kleine Problem beseitigt hatten, mussten wir nun noch Prüfen ob der aktuell spielende Titel wiederholt werden soll und ob die Playlist wiederholt werden soll. Nachdem wir dies lösen konnten mussten wir noch prüfen ob der Spielende Titel von einem Wiedergegebenen Album oder Interpret kommt, falls ja, dann sollen nur Titel aus dem gleichen Album oder Interpret abgespielt werden. Doch schliesslich konnten wir diese kleinen jedoch ziemlich komplizierten Probleme beseitigen und die AutoPlayfunktion funktioniert einwandfrei.



## Funktion in GUI implementieren

Zu guter Schluss mussten wir noch alle Funktionen korrekt mit dem GUI verknüpfen, damit die Funktion, welche wir zuvor implementiert hatten ausgeführt werden können. Dabei gab es einzig bei der Darstellung der Daten ein kleines Problem und zwar mussten wir zur Darstellung der Daten interne Klassen definieren, da es ansonsten nicht korrekt mit dem Binding funktioniert hätte.



# Test

Zu guter Letzt mussten wir nur noch den Test durchführen, zum Glück sind dabei keine Probleme aufgetaucht.

## Testcase 001: Erstmaliger Start

|  |  |
| --- | --- |
| **Name** | Erstmaliger Start |
| **Auslöser / Akteur** | User am PC |
| **Vorausbedingungen** | * Client PC ist eingeschaltet * Programm ist auf dem PC * Lieder sind auf dem PC |
| **Funktionsablauf** | * User startet EasyPlay * Fenster mit Ordnerstruktur erscheint * User wählt Ordner bzw. Lieder aus * Lieder werden hinzugefügt |
| **Erwartetes Ergebnis** | * Lieder sind nun in der Bibliothek vorhanden |
| **Testergebnis** | * Die Lieder werden erfolgreich geladen und sind nur in der Bibliothek vorhanden |

## Testcase 002: Normaler Start

|  |  |
| --- | --- |
| **Name** | Normaler Start |
| **Auslöser / Akteur** | User am PC |
| **Vorausbedingungen** | * Client PC ist eingeschaltet * Programm ist auf dem PC * Programm wurde bereits einmal gestartet * Lieder sind auf dem PC |
| **Funktionsablauf** | * User startet EasyPlay * Lieder sind direkt, ohne anzugeben von wo wieder in der Bibliothek |
| **Erwartetes Ergebnis** | * Lieder sind direkt in der Bibliothek vorhanden |
| **Testergebnis** | * Lieder werden korrekt geladen |

## Testcase 003: Playliste erstellen

|  |  |
| --- | --- |
| **Name** | Playliste erstellen |
| **Auslöser / Akteur** | User am PC |
| **Vorausbedingungen** | * Client PC ist eingeschaltet * Programm ist auf dem PC * Lieder sind auf dem PC * EasyPlay ist gestartet |
| **Funktionsablauf** | * User erstellt neue Playliste per Buttonclick * User gibt Namen der Playliste aus * User wählt Lieder aus der Bibliothek aus |
| **Erwartetes Ergebnis** | * Die ausgewählten Lieder sind in der Playliste vorhanden |
| **Testergebnis** | * Die ausgewählten Lieder sind in der Playlist vorhanden |

## Testcase 004: Lied abspielen

|  |  |
| --- | --- |
| **Name** | Lied abspielen |
| **Auslöser / Akteur** | User am PC |
| **Vorausbedingungen** | * Client PC ist eingeschaltet * Programm ist auf dem PC * Lieder sind auf dem PC * Programm ist gestartet |
| **Funktionsablauf** | * User klickt auf ein Lied * Lied wird abgespielt |
| **Erwartetes Ergebnis** | * Lied wird abgespielt |
| **Testergebnis** | * Das Lied wird korrekt abgespielt |

## Testcase 005: Warteliste

|  |  |
| --- | --- |
| **Name** | Warteliste |
| **Auslöser / Akteur** | User am PC |
| **Vorausbedingungen** | * Client PC ist eingeschaltet * Programm ist auf dem PC * Lieder sind auf dem PC * Programm ist gestartet |
| **Funktionsablauf** | * User klickt mit rechtem Mausbutton auf Lied und dann auf Warteliste * Lied wird zur Warteliste hinzugefügt * Sobald Lied an der Reihe ist wird es abgespielt |
| **Erwartetes Ergebnis** | * Lied zur richtigen Zeit abspielen |
| **Testergebnis** | * Das Lied wird zum richtigen Zeitpunkt abgespielt |

## Testcase 006: Lautstärke

|  |  |
| --- | --- |
| **Name** | Lautstärke |
| **Auslöser / Akteur** | User am PC |
| **Vorausbedingungen** | * Client PC ist eingeschaltet * Programm ist auf dem PC * Lieder sind auf dem PC * Programm ist gestartet * Ein Lied wird abgespielt |
| **Funktionsablauf** | * User verändert die Lautstärke per Regler * Lautstärke verändert sich |
| **Erwartetes Ergebnis** | * Lautstärke wird je nachdem lauter oder leiser |
| **Testergebnis** | * Die Lautstärke ist korrekt veränderbar |

## Testcase 007: Hinzufügen

|  |  |
| --- | --- |
| **Name** | Hinzufügen |
| **Auslöser / Akteur** | User am PC |
| **Vorausbedingungen** | * Client PC ist eingeschaltet * Programm ist auf dem PC * Lieder sind auf dem PC * Programm ist gestartet |
| **Funktionsablauf** | * Unter Datei Lied / Ordner hinzufügen anklicken * Lieder / Ordner auswählen und auf hinzufügen klicken * Lieder werden zur Bibliothek hinzugefügt |
| **Erwartetes Ergebnis** | * Lieder sind in der Bibliothek vorhanden |
| **Testergebnis** | * Alle im Ordner vorhanden Lieder werden in der Bibliothek gespeichert |

## Testcase 008: Beenden

|  |  |
| --- | --- |
| **Name** | Beenden |
| **Auslöser / Akteur** | User am PC |
| **Vorausbedingungen** | * Client PC ist eingeschaltet * Programm ist auf dem PC * Lieder sind auf dem PC * Programm ist gestartet |
| **Funktionsablauf** | * User klickt auf den Beenden Button * Programm beendet sich |
| **Erwartetes Ergebnis** | * Programm ist beendet |
| **Testergebnis** | * Das Programm wird beendet und der aktuelle Stand des Programms wird gespeichert |

## Testcase 009: Warteliste leeren

|  |  |
| --- | --- |
| **Name** | Warteliste leeren |
| **Auslöser / Akteur** | User am PC |
| **Vorausbedingungen** | * Client PC ist eingeschaltet * Programm ist auf dem PC * Lieder sind auf dem PC * Programm ist gestartet * Lieder sind in der Warteliste |
| **Funktionsablauf** | * User klickt unter Optionen auf „Warteliste leeren“ * alle Lieder werden aus der Warteliste gelöscht |
| **Erwartetes Ergebnis** | * Warteliste ist leer |
| **Testergebnis** | * Die Warteliste wird geleert. |