Politechnika Warszawska
Wydział Mechatroniki
Instytut Automatyki i Robotyki

Programowanie w języku Java
Projekt ćwiczeniowy

Odtwarzacz i edytor muzyki

Autorzy:

Konrad Traczyk
Jakub Szlendak

Prowadzący:
prof. dr hab. Barbara Siemiątkowska
mgr. inż Iryna Gorbenko

20 grudnia 2015

1 Wstęp

Przedmiotem projektu było zaprojektowanie aplikacji odtwarzającej pliki muzyczne z funkcją edycji w języku programowania Java. Ustalono następujące wymagania co do funkcjonalności:

- odtwarzanie plików w formacie MP3 i WAVE
- edycja plików w obydwu formatach
- realizacja podstawowych operacji edycyjnych - skracanie ścieżki muzycznej, wycinanie fragmentu ścieżki, modyfikacja poszczególnych próbek w ścieżce
- obsługa listy odtwarzania
- obsługa kilku trybów odtwarzania (pojedyncza piosenka, powtarzanie piosenki itp.)

2 Implementacja aplikacji

Podstawowym elementem odtwarzacza muzyki jest silnik dekodujący pliki MP3. Biblioteki standardowe języka Java nie dostarczają takiej funkcjonalności. Ponieważ zaimplementowanie dekodowania ramek pliku MP3 jest zadaniem złożonym i wykraczającym poza zakres merytoryczny tego projektu, wykorzystano do tego bibliotekę JLayer dostarczaną na licencji GPL przez firmę JavaZoom. Z wspomnianej biblioteki wykorzystano dekoder ramek MP3 oraz konwerter plików MP3 na format WAVE. Klasa odtwarzacza dostarczana przez wymienioną bibliotekę została zmodyfikowana do potrzeb projektu.

Program był rozwijany jednocześnie na dwóch systemach operacyjnych: Windows 7 oraz Linux Ubuntu 14.04. Jego implementacja nie jest zależna od systemu operacyjnego, jest to zatem aplikacja wieloplatformowa. W implementacji projektu zastosowano środowisko IntelliJ IDEA firmy JetBrains oraz system kontroli wersji Git.

3 Algorytm odtwarzania

Za odtwarzanie muzyki odpowiadają klasy *AdvancedPlayer* i *Player*. Jak wspomniano wcześniej, klasa odtwarzacza (AdvancedPlayer), należąca do biblioteki JLayer, została zmodyfikowana na potrzeby projektu. Modyfikacja polegała na zmianach w funkcji odtwarzającej tak, aby możliwe było odtwarzanie od dowolnej ramki. Działanie funkcji odtwarzającej polega na dekodowaniu i odtwarzaniu kolejnych ramek pliku w pętli, aż odtworzona zostanie ostatnia ramka bądź proces zostanie przerwany ręcznie.

Klasa *Player* opakowuje odwarzanie pliku w dodatkowe funkcjonalności - odtwarzanie od zadanego momentu, pauza, stop itp. Dodatkowo, obsługuje listę odtwarzania. Lista odtwarzania została zamodelowana jako kolekcja typu *LinkedList*. Pozwala to na wydajne operacje modyfikacji kolejności bądź wstawiania elementów.

Na liście agregowane są obiekty klasy *PlaylistItem*, zawierające obiekt klasy *File* przechowujący referencję do pliku muzycznego oraz napisy zawierające metadane piosenki, odczytane z pliku. Metadane są odczytywane podczas tworzenia obiektu klasy *PlaylistItem*. Jeżeli nie uda się odczytać metadanych, przyjmuje się że tytułem

piosenki jest nazwa pliku, a pozostałe pola są puste. Podejmowana jest także próba odczytania pliku graficznego w folderze z piosenką - jeżeli znajduje się w nim jakiś plik graficzny, jest on traktowany jako okładka albumu.

4 Algorytm edycji

Zaimplementowano następujące możliwości edycji pliku muzycznego:

- konwersja pliku MP3 do formatu WAVE
- wyciszanie fragmentu pliku o zadanej długości i początku
- wycinane zadanego fragmentu pliku ("przycinanie" piosenki)
- zmiana głośności piosenki

Ponieważ edycję przeprowadza się na poszczególnych próbkach w pliku muzycznym, konieczne jest operowanie na plikach nieskompresowanych. Dlatego operacje edycyjne w programie przeprowadzane są na plikach WAVE. Możliwe jest załadowanie do edytora pliku w formacie MP3, ale zostanie on przekonwertowany do tymczasowego pliku WAVE umieszczonego w folderze programu. Po zakończeniu edycji plik tymczasowy jest usuwany. Nalezy także nadmienić, że plik źródłowy nie jest modyfikowany podczas edycji.

Operacje edycyjne rozpoczynają się od pobrania nagłówka pliku WAVE. Zawarte są w nim informacje takie jak:

- format pliku
- liczba kanałów
- bitrate i sample rate (liczba bitów i próbek na sekundę)

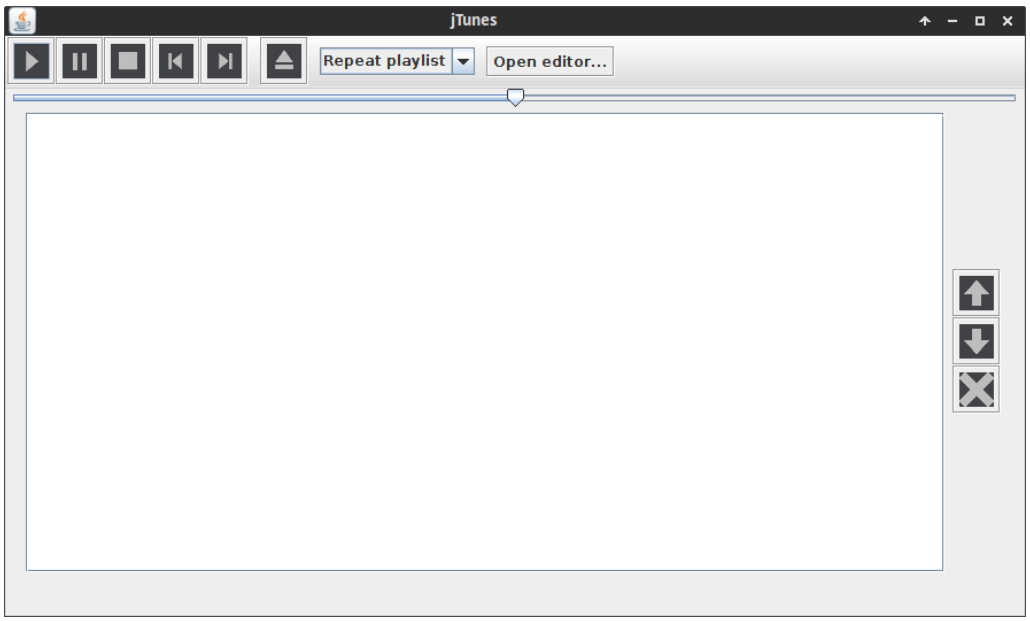
W nagłówku zawarta jest także informacja o pozycji bajtowej pierwszej próbki. Ta pozycja jest pobierana i traktowana jako początek danych o dźwięku. Mając tę informację, można poruszać się po pliku jak po tablicy typu **byte**. Należy przy tym pamiętać, że pojedyncza próbka zapisana jest na dwóch bajtach. Modyfikacja zawartości komórek tej tablicy wpływa na dźwięk.

W konsekwencji, wyciszanie fragmentu piosenki polega na wyzerowaniu pewnej ilości próbek, zaś zmiana głośności na przemnożeniu ich przez współczynnik.

Z kolei wycinanie fragmentu piosenki polega na wycięciu części bajtów i wstawieniu ich do strumienia wyjściowego, odpowiednio modyfikując nagłówek.

5 Interfejs użytkownika

Interfejs użytkownika został zaimplentowany przy użyciu biblioteki Swing.



Rysunek 1: Okno główne programu

Rysunek 1 przedstawia okno programu tuż po jego załadowaniu. Program jest w trybie odtwarzania. W górnej części okna widoczny jest pasek narzędzi odtwarzania. Funkcje kolejnych przycisków:

- rozpoczęcie odtwarzania
- pauza
- zatrzymanie odtwarzania
- następna piosenka z listy
- poprzednia piosenka z listy
- przycisk otwierania plików
- lista rozwijana trybów odtwarzania
- przycisk przełączenia na tryb edycji

Pod paskiem narzędzi umiejscowiono wskaźnik postępu odtwarzania, zaś poniżej znajduje się panel listy odtwarzania (por. rys. 2).

Poszczególne pozycje na liście zawierają okładkę albumu (jeśli została znaleziona, w przeciwnym razie wyświetlany jest obrazek domyślny), metadane piosenki (kolejno: wykonawcę, album i tytuł, bądź nazwę pliku jeśli metadane nie są dostępne) oraz długość piosenki w minutach i sekundach. Za wyświetlanie tych informacji odpowiada klasa *PlaylistItemRenderer*.

Po prawej stronie panelu listy odtwarzania umieszczono przyciski manipulowania listą. Pozwalają one na przesuwanie zaznaczonego elementu playlisty do góry, w dół bądź usuwanie go.

Aby rozpocząć odtwarzanie należy klikąć przycisk „Otwórz”, a następnie wybrać co najmniej jeden plik muzyczny do otwarcia (możliwe jest wybranie wielu plików, patrz rys. 3). Wybrane pliki zostają dodane do listy odtwarzania. Następnie należy wybrać jedną z 4 opcji porządku odtwarzania za pomocą listy rozwijanej:

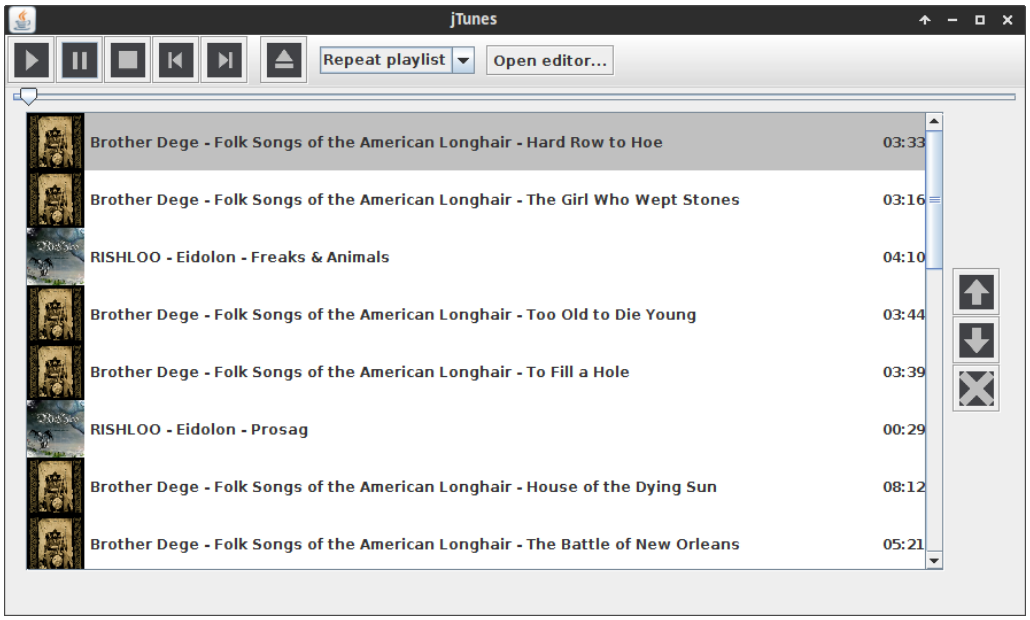
Repeat playlist odtwarzanie w porządku listy - piosenki z listy są odtwarzana kolejno, gdy osiągnięty zostanie koniec listy, odtwarzanie jest przenoszone na początek

Shuffle playlist odtwarzanie losowe z listy - piosenki z listy są odtwarzane w losowej kolejności

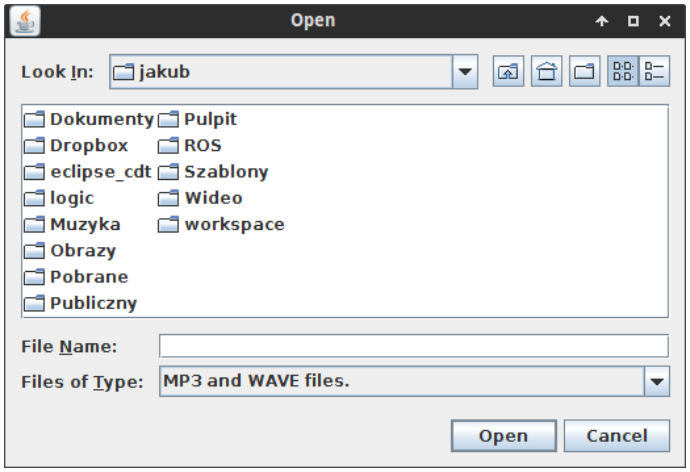
Play single odwarzanie zaznaczonej piosenki, po osiągnięciu końca piosenki otwarzanie zostaje zakończone

Repeat single zaznaczona piosenka jest odwarzana w pętli

Po wybraniu trybu odtwarzania można rozpocząć otwarzanie przyciskiem „Odtwarzaj" lub przez dwukrotne kliknięcie piosenki na liście.



Rysunek 2: Okno główne programu podczas odtwarzania

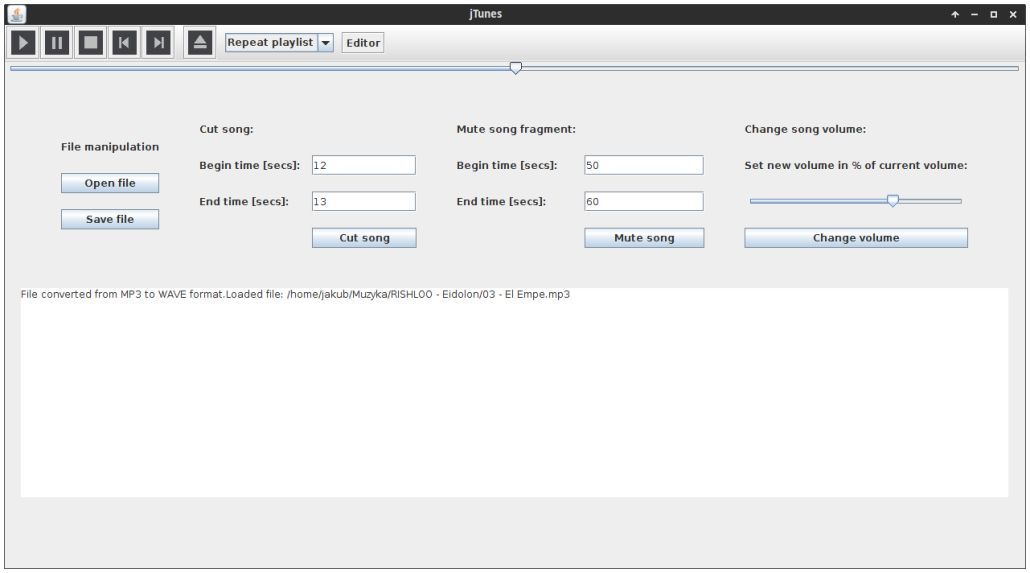


Rysunek 3: Okno otwierania pliku

Po wybraniu przycisku edycji otwarty zostaje widok edycji (por. rys. 4). Przyciski odtwarzania są nadal dostępne i odtwarzanie może być kontynuowane. W panelu edycji widoczne są cztery sekcje:

- przyciski otwierania i zapisu pliku
- kontrolki do przycinania pliku
- kontrolki do wyciszania fragmentu pliku
- kontrolki do zmiany głośności pliku

Przyciski edycyjne są nieaktywne do momentu załadowania pliku za pomocą przycisku „Open file". Po zakończeniu edycji plik można zapisać za pomocą przycisku „Save file". W celu wykonania przycięcia bądź wyciszenia piosenki, należy wpisać w odpowiednie pola edycyjne moment początku i końca modyfikacji (w sekundach). Modyfikacja głośności sprowadza się do wybrania nowej głośności (w procentach) i kliknięcia przycisku „Change volume". Możliwe jest wykonanie kliku operacji edycyjnych na jedno załadowanie pliku.



Rysunek 4: Ekran edycji

6 Testy aplikacji

Testy aplikacji przeprowadzono za pomocą frameworku JUnit z wykorzystaniem narzędzi opakowujących funkcjonalność JUnit wbudowanych w środowisko IntelliJ IDEA. Ze względu na specyfikę projektu nie było możliwe przetestowanie sensu stricte wszystkich funkcjonalności. Przeprowadzono testy następujących klas:

- Editor
- MP3Player
- PlaylistItemRenderer
- PlaylistItem
- Playlist

6.1 Testy jednostkowe klasy Editor

6.1.1 Metoda convertMP3ToWav()

W ramach testu konwertowano dany plik MP3 na plik WAVE o podanej nazwie a następnie za pomocą metod assertTrue sprawdzano czy plik istnieje, czy jest plikiem oraz czy jego rozszerzenie jest „wav“

6.1.2 Metoda convertWavToMP3()

Przebieg testu jest identyczny jak metody convertMP3ToWav.

6.1.3 Metoda loadSong()

W ramach testu wywołano metodę podając za argument plik testowy. Nie korzystano z metod asercji JUnit - powodzenie testu oznacza brak wyrzuconych wyjątków.

6.1.4 Metody edycyjne

Metody edycyjne przetestowano wywołując je na poprawnie załadowanym pliku. Powodzenie testu zależy od braku wyjątków.

6.2 Testy jednostkowe klasy MP3Player

Do celów testowych w metodzie setUp utworzono instancję klasy MP3Player, dodano jej instancję Playlist oraz dodano do niej dwie przykładowe piosenki.

6.2.1 Metoda addListener()

W ramach testu utworzono i dodano anonimową implementację interfejsu PlayerEventListener a następnie za pomocą funkcji assertNotNull sprawdzono, czy instancja klasy MP3Player rzeczywiście posiada dodany PlayerEventListener.

6.2.2 Metody sterowania odtwarzaniem

Testy metod pauseSong(), stopSong(), resumeSong() opierały się na podobnym schemacie. Uruchomiono odtwarzanie, programowo odczekano 100ms a następnie wywoływano testową metodę i za pomocą metody assertEquals sprawdzano, czy stan odtwarzania jest odpowiedni.

6.2.3 Metody set i get

Testy metod typu set i get polegały na ustawieniu danego pola za pomocą metody set na znany obiekt a następnie asercji, że metoda get zwraca ten obiekt.

6.3 Testy jednostkowe klasy PlaylistItemRenderer

6.3.1 Metoda getListCellRendererComponent()

Przeprowadzono dwa testy tej metody, dla piosenki posiadającej metadane oraz dla piosenki bez metadanych. Nie korzystano z asercji.

6.3.2 Metoda getScaledImage()

Za pomocą asercji assertNotNull sprawdzono, czy skalowanie testowego obrazu nie zwraca wskaźnika pustego.

6.4 Testy jednostkowe klasy Playlist

W ramach metody setUp() przygotowano testową listę odtwarzania z trzema przykładowymi pozycjami.

6.4.1 Metody modyfikacji zawartości

Metody modyfikacji zawartości przetestowano wywowując je na przykładowych instancjach PlaylistItem, sprawdzając za pomocą metody assertEquals czy rezultaty metod są zgodne z oczekiwaniami

6.4.2 Metoda getSize()

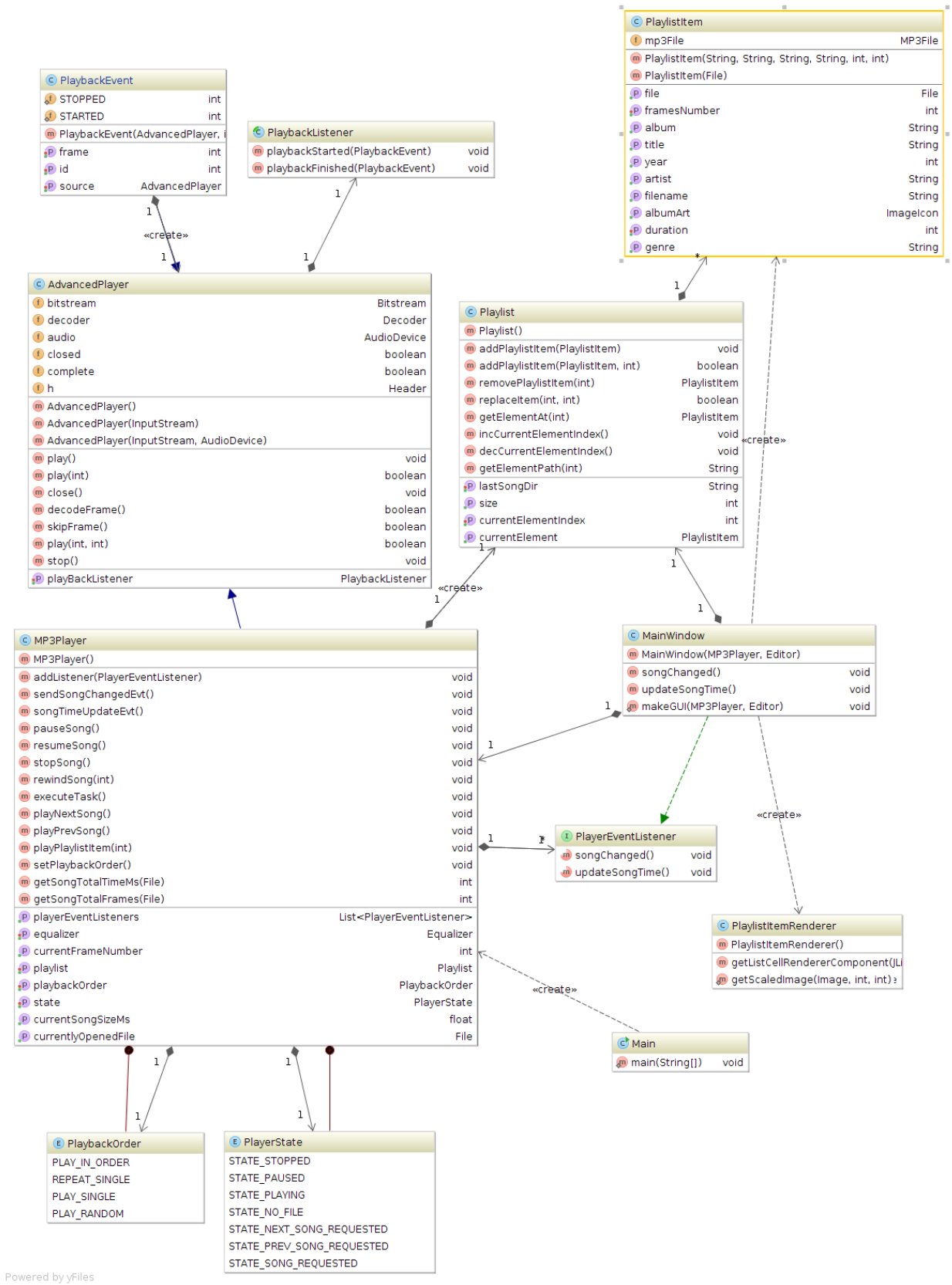
Sprawdzono, czy metoda zwraca odpowiedni rozmiar dla niepustej listy oraz czy zwraca 0 dla nowoutworzonej instancji Playlist.

6.4.3 Metody manipulacji indeksem aktualnej piosenki

Metody pobrania, ustawienia, inkrementacji i dekrementacji indeksu aktualnej piosenki

7 Schemat klas

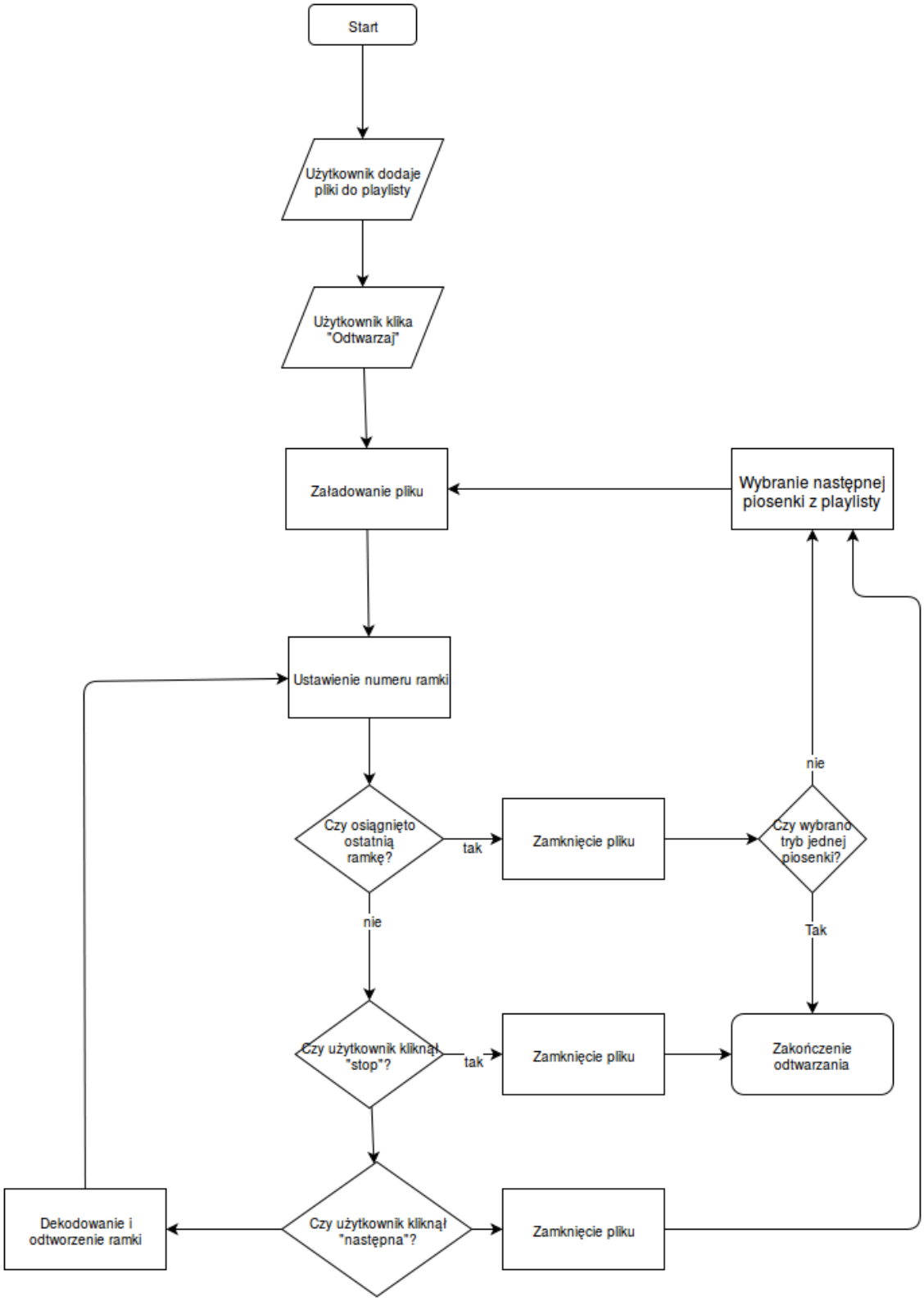
Na rysunku 5 zawarto schemat klas wg standardu UML, wygenerowany za pomocą narzędzi wbudowanych w środowisko IntelliJ IDEA.



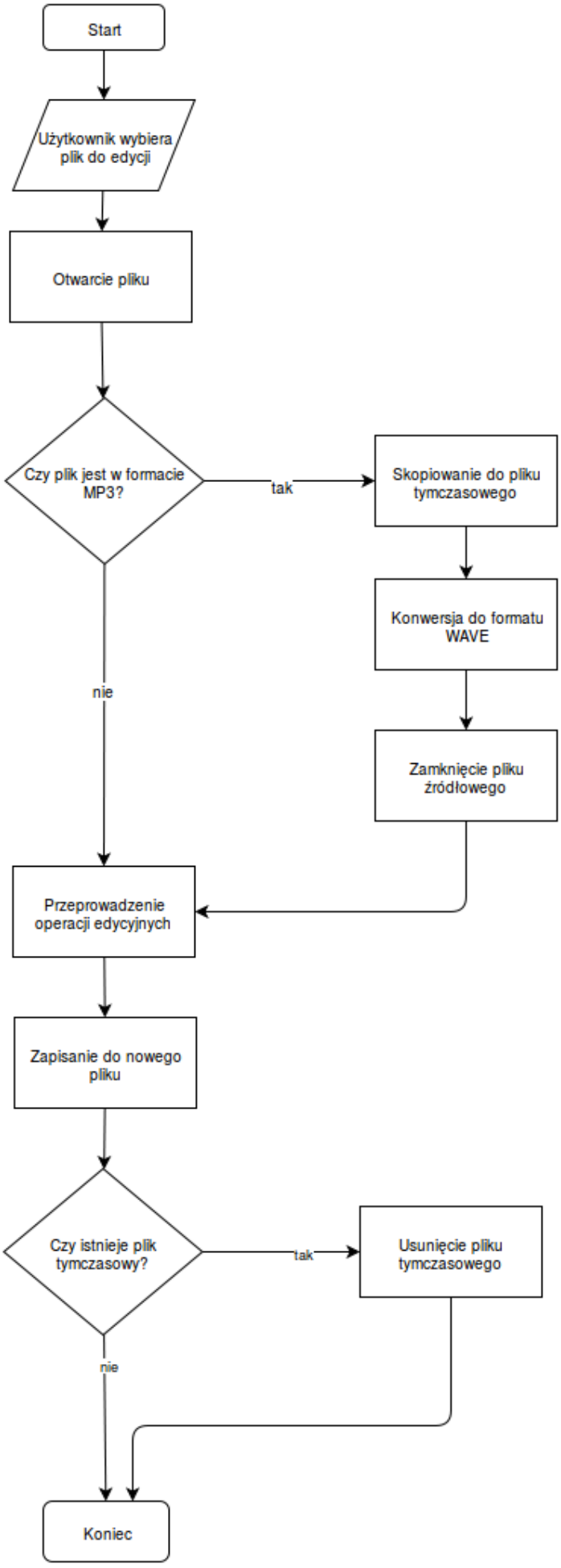
Rysunek 5: Schemat klas

8 Schemat blokowy algorytmu

Poniżej zawarto schemat blokowy algorytmu odtwarzania i edycji pliku.



Rysunek 6: Algorytm odtwarzania



Rysunek 7: Algorytm edycji