## Politechnika Świętokrzyska

## Wydział Elektroniki, Automatyki i Informatyki

### Kierunek: Informatyka

### Rok: II, semestr: IV

### Studia: stacjonarne

### Skład zespołu: Januszek Hubert, Jagodziński Maksymilian

### Grupa dziekańska: 2ID12A

Kielce, 2017

Temat: Zaawansowana aplikacja użytkowa do obsługi i zarządzania bazą danych systemu bibliotecznego

Przedmiot: Programowanie w języku Java (projekt)

**Spis treści**

1. Tematyka projektu ……………………………………………………………………. 3-4

2. Użyty język programowania oraz biblioteki ………………………………………………. 4

3. Instrukcja kompilacji/uruchomienia oraz obsługi programu ……………………………… 4

4. Opis działania programu ………………….……………………………………………...... 5

5. Opis użytych algorytmów .................................................................................................. 5-6

6. Zrzuty ekranu z przykładowym działaniem ……………………………..………………. 6-7

7. Wkład członków ….............................................................................................................. 8

# 1. Tematyka projektu

Projekt dotyczy aplikacji do obsługi wypożyczalni książek. W ramach działania programu możliwe jest dodawanie, usuwanie i edycja klientów, książek, pracowników oraz wprowadzanie pewnych zmian w strukturze samej bazy danych. Program został wzorowany na sposobie działania bibliotek uczelnianych i jest przeznaczony dla użytkowników, którzy są pracownikami wypożyczalni i mają kontakt z klientem, który dokonuje wypożyczeń jednej, lub więcej książek, lub dla administratorów bazy danych.

W ramach prawidłowego funkcjonowania wypożyczalni książek należy zadbać o poprawność wszelkich niezbędnych danych, poczynając od ich dodania, przechowywania i edycji, aż po usunięcie. Podstawowymi informacjami są informacje o klientach i o posiadanych zbiorach bibliotecznych. Należy też pamiętać o samym mechanizmie wypożyczeń książek – trzeba zapamiętać kto wypożyczył jaką książkę, w jakiej ilości i w jakim terminie. Terminy wypożyczeń są niezbędne do prawidłowego działania wypożyczalni. Dzięki temu istnieje możliwość naliczania opłat za przetrzymywanie książek.

Fundamentalne informacje o klientach, które muszą być przechowywane w takich miejscach to: numer karty (ID) użytkownika, imię i nazwisko. Karta użytkownika musi posiadać unikalny numer, aby poprawnie rozróżnić dwóch różnych klientów, którzy posiadają, np. takie same nazwisko i imię.

Informacjami o książkach, które wypożyczalnia musi posiadać są: unikalny numer (ID), tytuł, autor (lub autorzy), gatunek, rok wydania, wydawnictwo, fizyczna lokalizacja w bibliotece, liczba posiadanych i wypożyczonych egzemplarzy. Dane te mogą służyć do grupowania książek po dowolnie wybranym polu, aby otrzymać tylko te książki, którymi jest zainteresowany użytkownik. Najważniejszymi informacjami podanymi powyżej jest ilość posiadanych i wypożyczonych książek. W przypadku, gdy liczba wypożyczonych książek wynosi 0, to nie ma już możliwości dalszego jej wypożyczania i należy zaczekać do momentu, aż któryś z klientów zwróci ją do biblioteki, bądź wypożyczalnia zakupi dodatkowe egzemplarze.

Tematyka projektu to bardzo szeroki temat, w którym wiele informacji jest ze sobą ściśle powiązanych. Nie można dopuścić do utraty danych, ponieważ działanie wypożyczalni byłoby zaburzone. Należy dbać o poprawność wpisywanych danych, aby uniknąć problemów w przyszłości.

# 2. Użyty język programowania oraz biblioteki

Biblioteki, które zostały wykorzystane w projekcie to:

* jdbc6.jar – biblioteka służąca do nawiązywania połączenia z bazą danych
* JTattoo.jar – biblioteka służąca do nadania alternatywnego wyglądu okien

W programie projektowym użyto języka Java 8 wraz z biblioteką graficzną Swing, oraz środowiska programistycznego NetBeans IDE 8.2. Do utworzenia bazy danych wykorzystany został program SQL Developer 4.1.5.21.

# 3. Instrukcja kompilacji/uruchomienia i obsługi programu

W celu poprawnego uruchomienia programu niezbędne jest uprzednie utworzenie bazy danych. W programie SQL Developer (bądź innym programem tego typu) należy otworzyć plik *database\_create.sql* znajdujący się w folderze *database* i uruchomić go. Zostaną utworzone tabele i triggery oraz wrzucone dane do tych tabel.

Następnym krokiem jest uruchomienie projektu w środowisku programistycznym i edycja pliku *DatabaseConnection.java* w linijce 22, która wygląda następująco:

Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE", "username", "password");

Należy zedytować pola *username* i *password*, w miejsce których należy podać nazwę użytkownika i hasło do bazy danych. W zależności od ustawień na poszczególnych komputerach, może być też wymagana zmiana numeru portu, którym domyślnie jest 1521.

Po wykonaniu powyższych czynności można przejść do kompilacji programu i uruchomienia go. Pierwszym oknem, które ukazuje się użytkownikowi jest okno logowania, w którym należy podać nazwę użytkownika i hasło (zgodnie z danymi zawartymi w tabeli *Użytkownicy bazy*). Po wpisaniu nieprawidłowych danych program zwróci błąd z tym związany. W przypadku podania prawidłowych danych użytkownik uzyskuje dostęp do bazy danych. W zależności od posiadanych uprawnień widok jest inny dla bibliotekarza i dla administratora bazy danych.

W obu widokach widoczne są panele, na których umieszczone są ikony wraz z nazwami, które przedstawiają jaka czynność zostanie wykonana po ich uruchomieniu. W operacjach związanych z dodawaniem, bądź edycją danych dostępne są pola, do których użytkownik może wpisywać dane z klawiatury. Pola obsługują walidację danych, więc niemożliwe jest wpisanie danych o niepoprawnym formacie (np. wymagane są dane liczbowe, a użytkownik wpisze ciąg znaków).

Więcej informacji na temat poruszania się w strukturze programu jest przedstawione w rozdziałach: **4. Opis działania programu** oraz **6. Zrzuty ekranu z przykładowym działaniem**.

# 4. Opis działania programu

Program łączy się z bazą danych na komputerze lokalnym za pomocą połączenia typu Connection zaimportowanej z klasy java.sql.\*. Na podstawie wpisanego loginu i hasła program decyduje, które okno zostanie pokazane użytkownikowi. Jeśli jego stanowisko pracy to *bibliotekarz*, to uruchomione zostanie okno *UserFrame,* natomiast jeśli stanowisko to *administrator*, uruchomione zostanie okno *AdminFrame*.

Po załadowaniu się jednego z wcześniej wymienionych okien, użytkownik programu może od razu z niego korzystać i wprowadzać modyfikacje w zawartości bazy danych. Operacje te są różne dla bibliotekarza i administratora. Bibliotekarz zajmuje się czynnościami związanymi z obsługą klientów i zarządzaniem książkami, administrator natomiast zajmuje się tworzeniem, bądź usuwaniem klientów oraz pracowników, a także nadawaniem uprawnień do logowania się do bazy danych.

Wszystkie opcje, z których może korzystać dany użytkownik są jednoznacznie podpisane, więc nie występują tu żadne niejednoznaczności, na skutek których użytkownik nie wiedziałby jak wykonać daną procedurę.

# 5. Opis użytych algorytmów

**Funkcja regexChecker z klasy RegexChecker**

Funkcja przyjmuje dwa parametry wywołania. Pierwszy z nich jest to ciąg znaków odpowiadający tak zwanemu Patternowi regexa czyli ciągu służącemu do walidacji danych. Drugim parametrem jest natomiast badany ciąg znaków.

W funkcji kompilowany jest Pattern za pomocą metody compile(), następnie wywoływana jest metoda matcher() związana z klasą Matcher.

Wartością zwracaną jest typ boolean czyli prawda lub fałsz w zależności od tego czy badany ciąg nzaków spełnia wymagania Patternu.

**Funkcja loginToBase z klasy Login**

Funkcja ta nie przyjmuje żadnych parametrów i nie zwraca żadnej wartości.

Na początku sprawdzane jest poprzez polecenie SQL czy podany login i hasło jest zgodne z którymkolwiek wierszem w tabeli

UZYTKOWNICY\_BAZY, jeśli nie to wypisywany jest odpowiedni komunikat o tym, że login kub hasło nie jest prawidłowe.

W przypadku gdy login i hasło się zgadzają, pobierany jest z zapytania identyfikator pracownika w celu stwierdzenia czy

jest on bibliotekarzem czy administratorem. Na podstawie tej informacji uruchamiany jest odpowiedni interfejs(ramka).

**Funkcja updateTable**

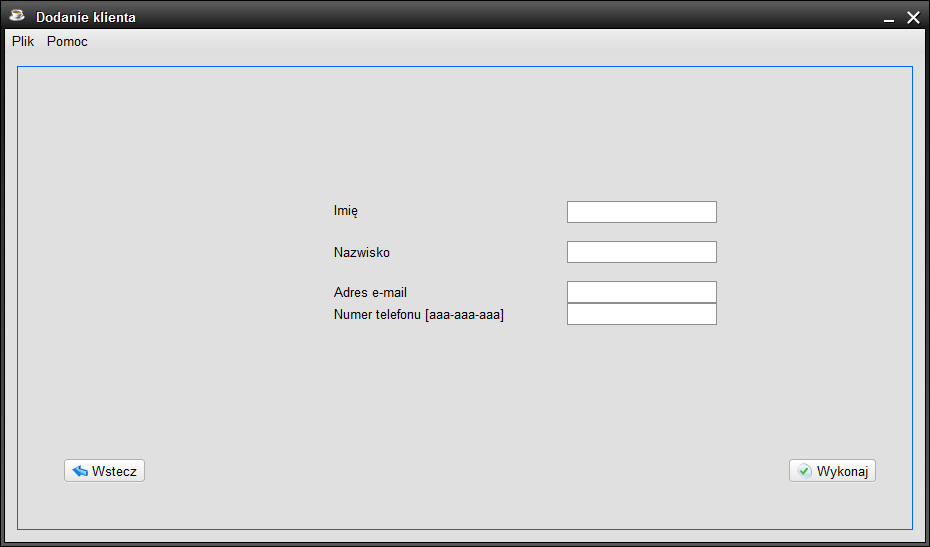
Funkcja ta w zależności gdzie jest implementowana i do jakiego celu jest używana przyjmuje różną liczbę parametrów.

Wykorzystywana jest ona do wyświetlenia odpowiedniej tabeli(jej nazwa przekazana jest jako parametr).

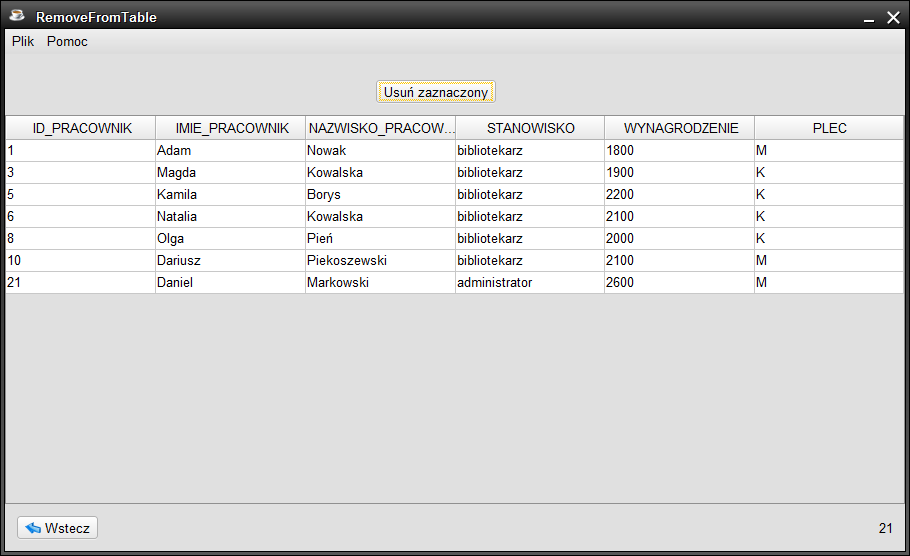
Na początku składane jest zapytanie SQL z odpowienią nazwą tabeli. Następnie tworzony jest widok tabeli wraz z jej zawartością

poprzez wywołanie funkcji setModel dla zmiennej będącą nazwą tabeli.

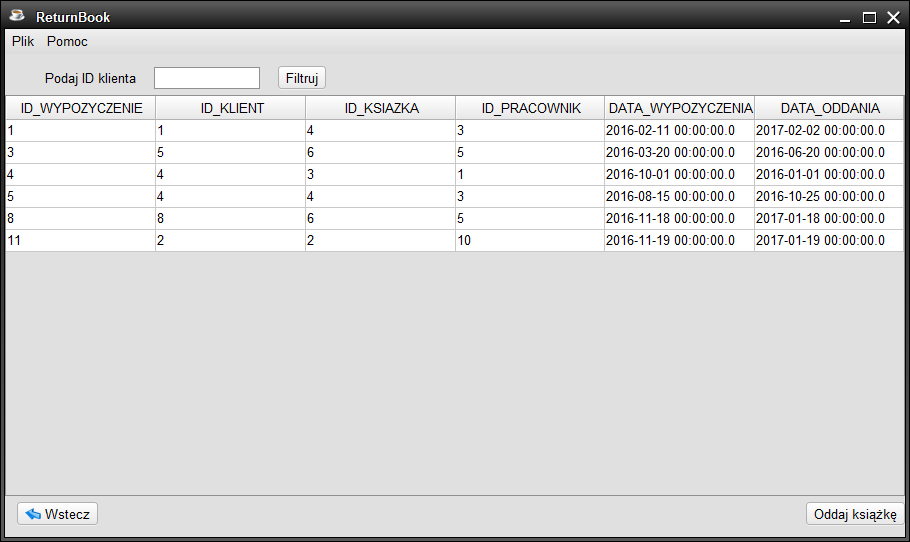
# 6. Zrzuty ekranu z przykładowym działaniem



Okienko wyświetlane dla akcji dodania klienta. Po wpisaniu danych do formularza i naciśnięciu buttona "*Wykonaj*" sprawdzane jest przed wstawieniem rekordu do bazy czy dane, które użytkownik próbuje wprowadzić są prawidłowe. Jeśli tak, to procedura kończy się powodzeniem.



Okienko wyświetlane dla akcji usuwania. W tym konkretnym przypadku rozpatrujemy usuwanie z tabeli *Pracownicy*. Wystarczy zaznaczyć na wyświetlanej tabeli rekord, który chcemy usunąć po czym nacisnąć button "*Usuń zaznaczony*".



Ramka wyświetlana po wybraniu opcji "*Zwrot książki*". Domyślnie wyświetlana jest tabela z wszystkimi wypożyczeniami. Gdy w pole powyżej tabeli wprowadzimy ID interesującego nas klienta i naciśniemy button "*Filtruj*", wówczas otrzymamy wszystkie wypożyczenia tego klienta. Wystarczy zaznaczyć odpowiedni wiersz w wyświetlanej tabeli oraz nacisnąć button "*Oddaj książkę*" by wypożyczona książka została oddana i ten wpis nie widniał już w bazie.

# 7. Wkład członków

Wkład włożony przez obu członków zespołu rozkłada się po równo.