Uniwersytet Rzeszowski

Wydział Matematyczno-Przyrodniczy

Kierunek: Informatyka

„ClinMD” - wirtualna rejestracja

Wykonał: Szymon Kulikowski,

Informatyka 2-gi rok, lab. 2

Opiekun: dr inż. Wojciech Kozioł

Rzeszów 2017

Spis treści

[1. Opis aplikacji: 3](#_Toc473657376)

[2. Instrukcja obsługi: 4](#_Toc473657377)

[2.1. Panel logowania: 4](#_Toc473657378)

[2.2. Panel pacjenta: 5](#_Toc473657379)

[2.3. Panel lekarza: 6](#_Toc473657380)

[3. Baza danych: 8](#_Toc473657381)

[3.1. Opis bazy danych: 8](#_Toc473657382)

[4. Omówienie kodu programu (wybrane aspekty) 10](#_Toc473657383)

[4.1. ConectionManager.java 10](#_Toc473657384)

[4.2. Login.java 10](#_Toc473657385)

[4.3. Pacjent.java 12](#_Toc473657386)

[4.4. Lekarz.java 13](#_Toc473657387)

[4.5. Register.java 13](#_Toc473657388)

[4.6. Forget.java 14](#_Toc473657389)

[4.7. Rejestracja.java 14](#_Toc473657390)

[4.8. Badanie.java 15](#_Toc473657391)

[4.9. EdycjaD.java 16](#_Toc473657392)

[4.10. EdycjaG.java 17](#_Toc473657393)

[4.11. Historia.java 17](#_Toc473657394)

[4.12. Table.java 18](#_Toc473657395)

[4.13. Wyjatki.java 19](#_Toc473657396)

# Opis aplikacji:

Aplikacja „ClinMD” służy do elektronicznego zarządzania rejestracją w przychodni. Pacjent może stworzyć własne konto, podając najważniejsze dane takie jak imię, nazwisko, PESEL, numer telefonu oraz informacje o ubezpieczeniu.

Po stworzeniu konta Pacjent może umówić się na wizytę do lekarza specjalisty czy też lekarza rodzinnego, lub na badanie takie jak USG, EKG oraz badania krwi. Jeżeli pacjent posiada już umówione wizyty może też je sprawdzić, tj. jego datę, godzinę, imię i nazwisko lekarza oraz numer pokoju, w którym wizyta się odbędzie.

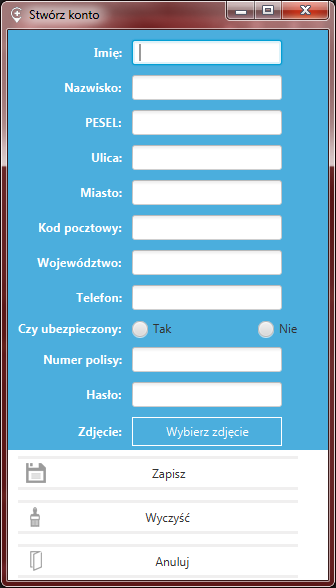
Dostęp do aplikacji posiadają również lekarze, którzy mają predefiniowane konta, zakładane podczas przyjęcia do pracy. Lekarz w swoim panelu może wybrać dni, w których będzie przyjmować, dokładne godziny przyjęć oraz salę. Kolejną opcją jest sprawdzenie wszystkich umówionych wizyt lub badań na dany dzień. Ostatnią opcją jest edycja historii pacjenta, w której lekarz może zapisać diagnozę oraz dodatkowe informacje o pacjencie, tworząc dzięki temu cyfrową wersję kartoteki.

Aplikacja została napisana w języku Java, w technologii JavaFX. Środowiskiem pracy był NetBeans IDE 8.1 z nakładką „darcula” oraz własnymi modyfikacjami. Baza danych stworzona w programie SQLiteStudio(3.1.1).

# Instrukcja obsługi:

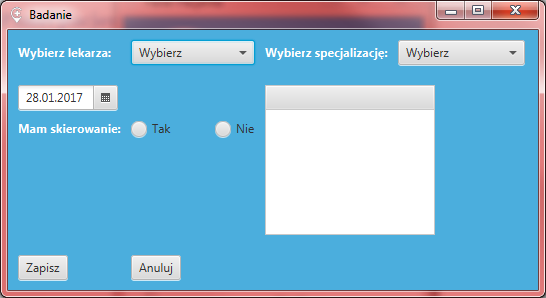
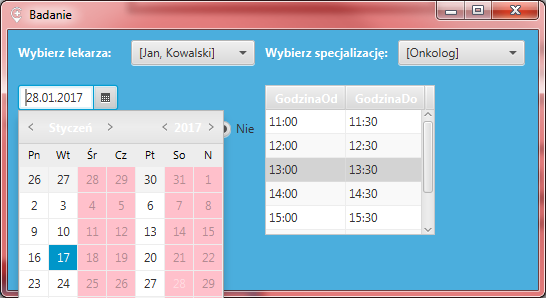
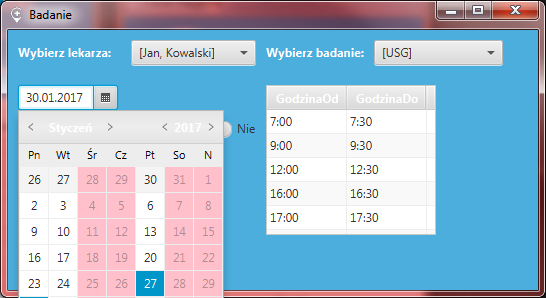
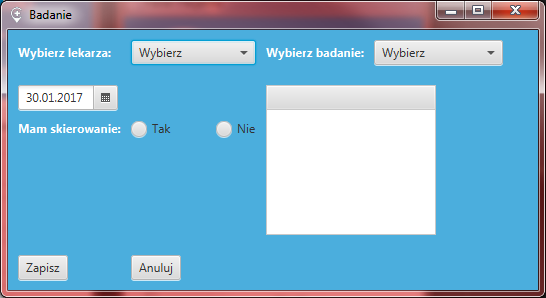
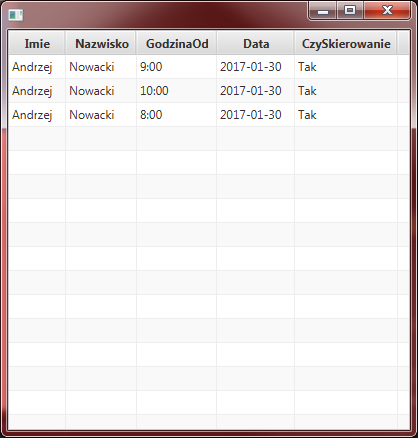
## Panel logowania:

* Po naciśnięciu przycisku „Zaloguj” wyświetlą nam się pola, w których będziemy mogli wpisać PESEL oraz Hasło w celu zalogowania się do aplikacji.
* „Stwórz konto” przeniesie nas do kolejnego okna, w którym będziemy mogli stworzyć swoje konto, po właściwym wpisaniu danych do formularza.
* „Przypomnij hasło” ukaże nam okno, dzięki któremu będziemy mogli odzyskać zapomniane hasło po wpisaniu wymaganych danych.



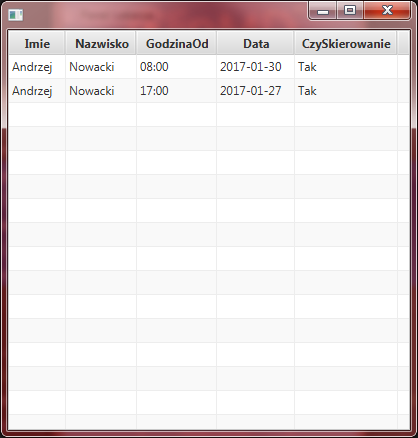
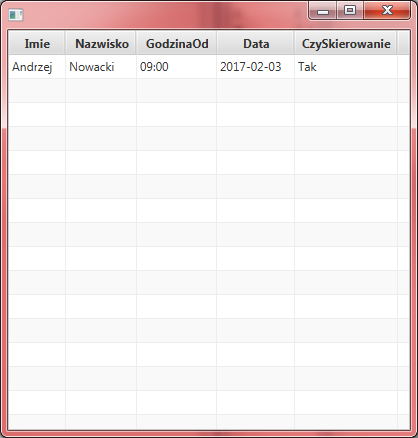
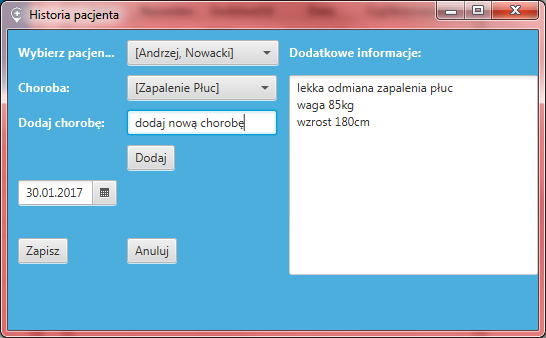
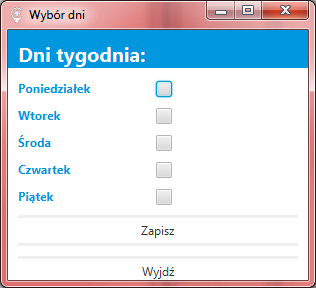
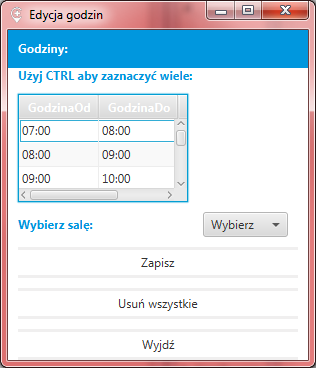
## Panel pacjenta:

Po zalogowaniu się jako pacjent oczom użytkownika ukażą się jego dane takie jak imię, nazwisko, pesel, numer polisy i wiadomość o tym czy jest ubezpieczony oraz avatar ustawiony podczas rejestracji. Poniżej znajdują się cztery przyciski przenoszące nas do odpowiednich okien:

* „Zarejestruj” przeniesie nas do panelu rejestracji, w którym pacjent będzie mógł wybrać lekarza lub specjalizacje lekarza, do którego chce się umówić. Po wybraniu lekarza pacjent będzie mógł wybrać dzień, na który chce się zarejestrować (dni oznaczone na czerwono to dni, w których lekarz nie przyjmuje), oraz godzinę wizyty wyświetloną w tabeli.
* „Umów badanie” przeniesie nas do panelu rejestracji, w którym pacjent będzie mógł wybrać lekarza lub badanie, na które chce się umówić. Po wybraniu lekarza pacjent będzie mógł wybrać dzień, na który chce się zarejestrować (dni oznaczone na czerwono to dni, w których lekarz nie przyjmuje), oraz godzinę wizyty wyświetloną w tabeli.
* „Sprawdź rejestrację” pokazuje wszystkie umówione wizyty oraz badania danego pacjenta. Godzinę, dzień, lekarza lub badanie oraz pokój, w jakim się ono odbędzie.

## Panel lekarza:

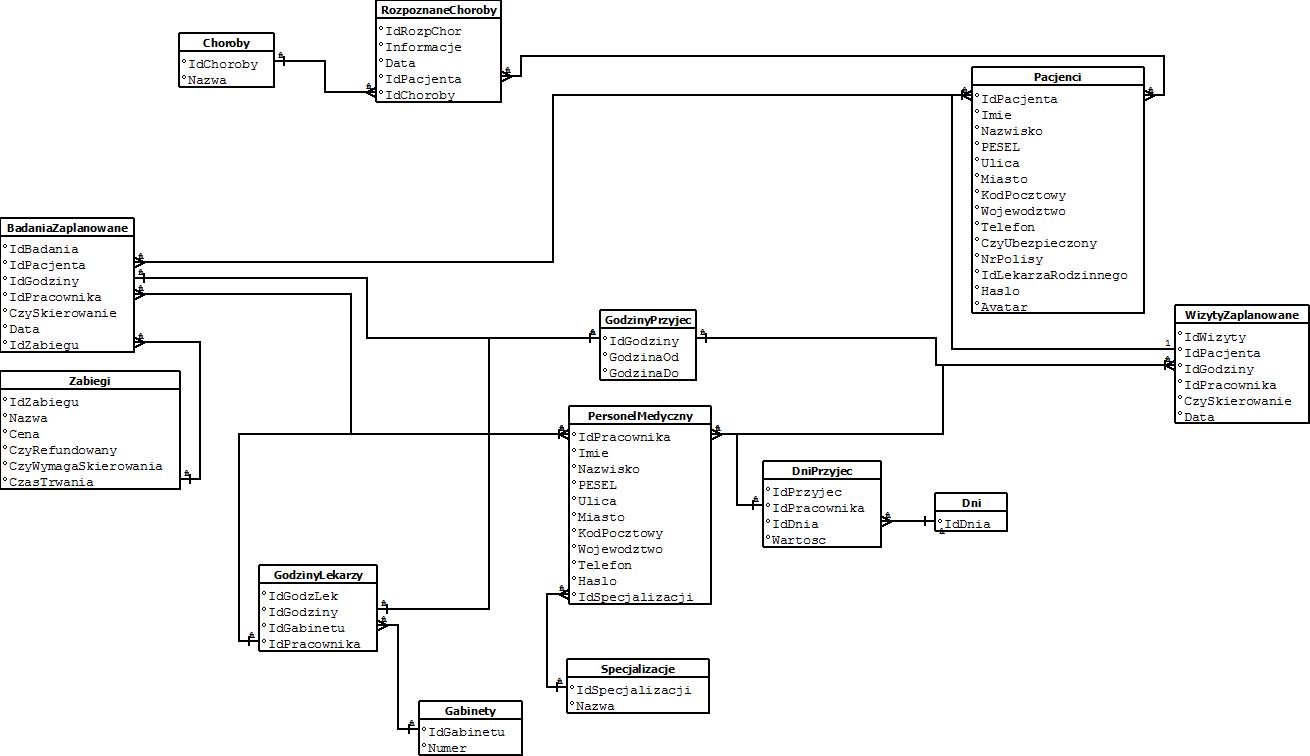
Po zalogowaniu się jako lekarz oczom użytkownika ukażą się jego dane takie jak imię, nazwisko, nr telefonu, specjalizacja oraz avatar. Poniżej znajduje się pięć przycisków przenoszących nas do odpowiednich okien:

* „Pokaż listę pacjentów” pokazuje listę pacjentów, którzy są umówieni do zalogowanego lekarza:
* „Pokaż listę badań” wyświetla wszystkie umówione badania, które ma wykonać lekarz:
* „Historia pacjenta” to okno, w którym lekarz po wybraniu pacjenta, może wybrać z listy zdiagnozowaną chorobę, lub dodać nową do listy. Może dodać datę, w której pacjent został przyjęty, oraz wpisać dodatkowe informacje do wirtualnej kartoteki pacjenta:
* „Zmień dni przyjęć ” pozwala wybrać dni, w których lekarz będzie przyjmował w przychodni:
* „Zmień godziny przyjęć” w tym oknie lekarz dodaje lub usuwa godziny, w których przyjmuje i numer gabinetu. Należy pamiętać, że godziny można dodawać jedynie pojedynczo:

# Baza danych:

## Opis bazy danych:

* Baza danych odpowiedzialna jest za poprawne działanie przychodni lekarskiej. Przechowuje informacje o personelu, pacjentach, chorobach, historii choroby, godzinach przyjęć, umówionych wizytach i badaniach oraz zdiagnozowanych chorobach.
* **BadaniaZaplanowane:** informacje badaniach, na które umówili się pacjenci
* **Choroby:** spis zdiagnozowanych chorób
* **Dni:** tabela pomocnicza do zapisu dni, w których przyjmują lekarze
* **DniPrzyjec:** przechowuje informacje o dniach w jakich przyjmuje dany lekarz
* **Gabinety:** spis gabinetów w placówce
* **GodzinyLekarzy:** spis godzin w jakich przyjmuje dany lekarz
* **GodzinyPrzyjec:** wszystkie możliwe godziny przyjęć w placówce
* **Pacjenci:** zbiór wszystkich zarejestrowanych pacjentów
* **PersonelMedyczny:** wszyscy lekarze pracujący w przychodni
* **RozpoznaneChoroby:** wirtualna kartoteka, w której przechowywana jest historia pacjenta
* **Specjalizacje:** tabela przechowująca wszystkie specjalizacje lekarzy
* **WizytyZaplanowane:** spis wszystkich zaplanowanych wizyt pacjentów
* **Zabiegi:** lista zabiegów, możliwych do wykonania w placówce



# Omówienie kodu programu (wybrane aspekty)

## ConectionManager.java

Utworzenie metody Connection, która odpowiada za połączenie z bazą danych. getConnection sporządza połączenie z bazą danych pod zadanym adresem url (DB\_URL) przez odpowiedni sterownik (DRIVER)

Zarejestrowanie sterownika i wskazanie ścieżki z bazą danych.

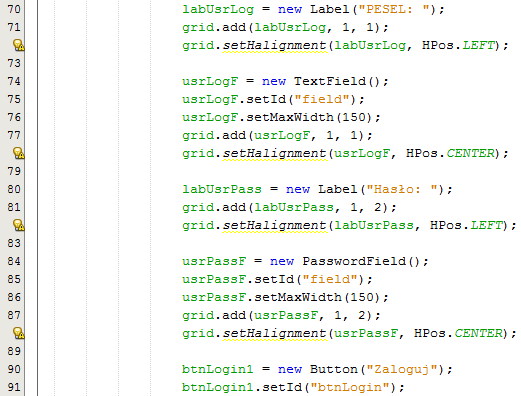
## Login.java

Połączenie z plikiem screen.css w celu ostylowania okien aplikacji.

Inicjalizacja połączenia z bazą danych.

Ustawienie ikony programu.

Podstawowe ustawienia siatki, w której będą umieszczane elementy programu.



Ustawienie elementów w odpowiednich miejscach na siatce: nazwa elementu, pozycja x, pozycja y.

Inicjalizacja labeli i pól tekstowych.

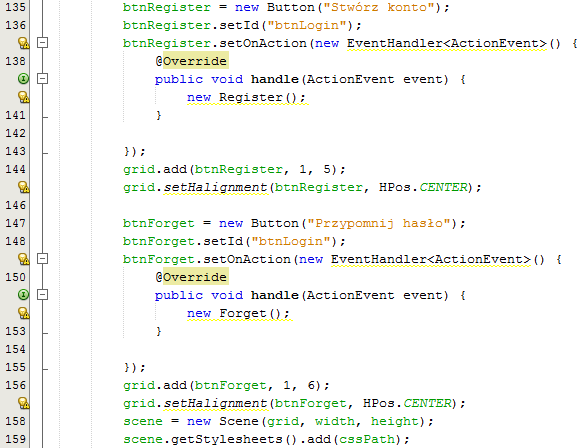
W przypadku braku wyników wyświetlany jest komunikat o błędnym PESELu lub haśle.

Jeżeli znaleziono, to do zmiennej idPacjenta jest przypisywane id znalezionej osoby (id z bazy danych).

Uruchomienie zapytania i sprawdzenie czy w którejś z tabel znajduje się osoba o podanym PESELu i haśle.

Pobranie wszystkich danych za pomocą zapytania w SQL pacjenta lub lekarza, pesel i hasło zostaną wpisane do odpowiednich pól.

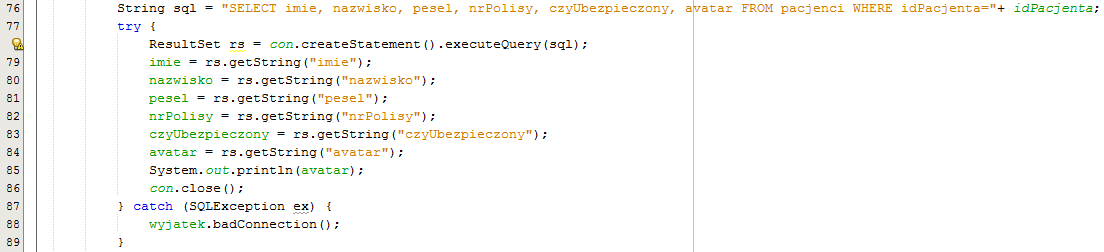
System logowania do aplikacji.



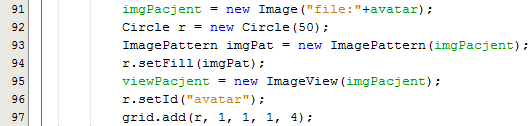
Ten fragment kodu odpowiedzialny jest za pobranie danych, które będą wyświetlane w górnym pasku po zalogowaniu do panelu pacjenta.

Przyciski btnRegister i btnForget przenoszą nas odpowiednio do okna z formularzem rejestracyjnym i okna, w którym można przypomnieć zapomniane hasło. Poprzez wywołanie funkcji znajdującej się w odpowiedniej klasie.

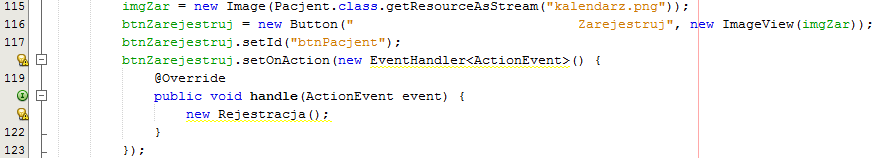
## Pacjent.java



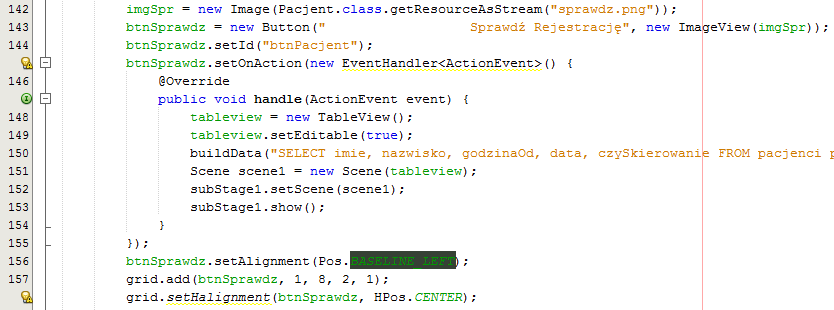
Zapytanie SQL, oraz przekazanie go do ResultSet, który umożliwia przypisanie danych z BD do odpowiednich pól w programie.



W tym momencie pobierany jest avatar użytkownika i wyświetlany jest jako koło



Tutaj jest umieszczanie obrazka do przycisku.

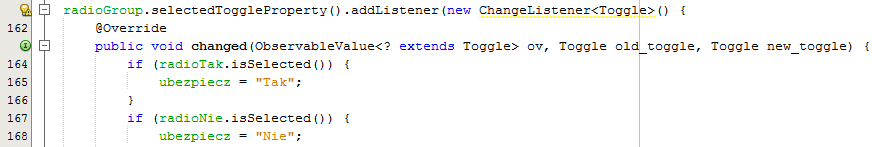


Bezpośrednio po naciśnięciu przycisku „Sprawdź rejestrację” pobierane są dane z BD i wyświetlane są w nowym oknie w formie tabeli.

## Lekarz.java

W tej klasie wszystkie operacje jakie się odbywają są analogiczne do klasy Pacjent.java. Różnią się jedynie nazwami i zapytaniami w SQL.

## Register.java



Ten fragment kodu odpowiada za radioButtony, a mianowicie pilnuje, żeby nie były wciśnięte obydwa na raz.



Jeśli użytkownik nie wybierze zdjęcia to ładowany jest domyślny avatar.

Zastrzeżenie, że tylko pliki graficzne mogą zostać wybrane.

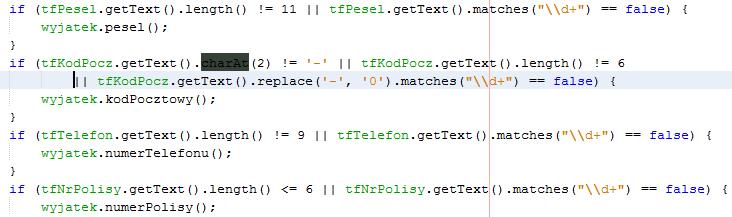
Inicjalizacja filechooser, dzięki któremu użytkownik może wybrać własny avatar.

1.

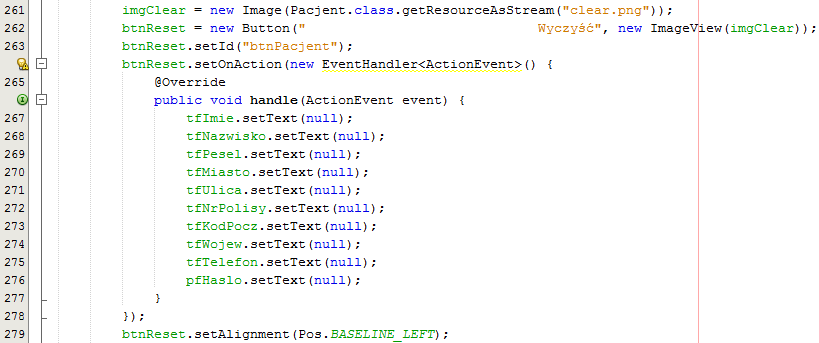
2.

3.

4.

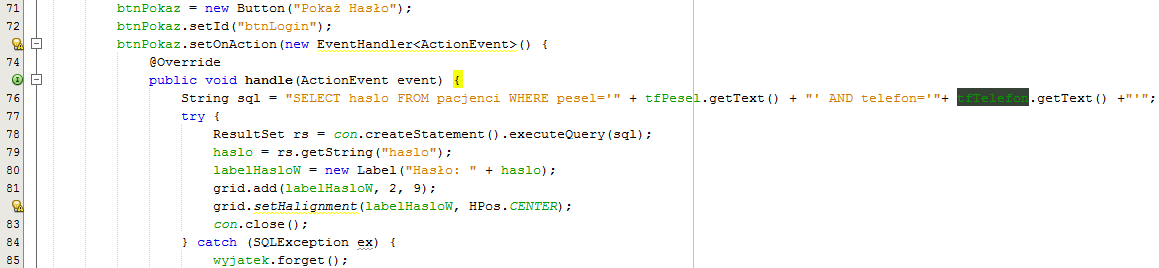


1. Sprawdzam numer PESEL czy ma długość 11 cyfr i czy składa się z samych cyfr. Jeżeli nie wyświetlam alert.
2. Sprawdzam czy kod pocztowy jako trzeci znak posiada ‘ - ‘, następnie czy ma długość sześciu znaków a na koniec czy składa się z samych cyfr, przez co muszę zastąpić myślnik dowolną liczbą, która nie ma wpływu na zapis do BD.
3. Sprawdzam nr telefonu, tak aby miał 9 cyfr i same cyfry.
4. Sprawdzam czy numer polisy ma więcej niż 6 cyfr i ma same cyfry.



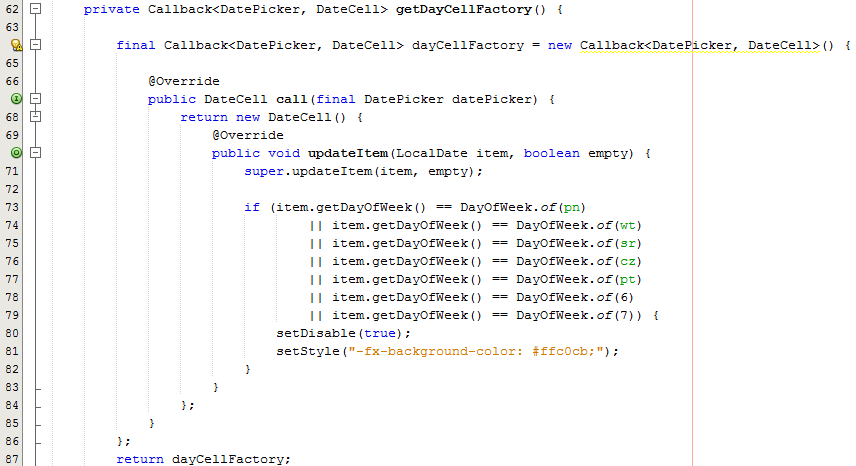
Metoda czyszcząca pola w formularzu rejestracyjnym, ustawiająca tekst w polach na null.

## Forget.java

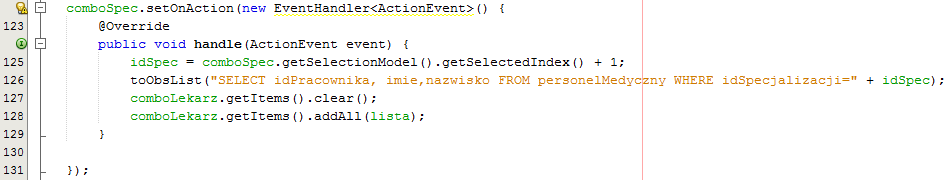


Wypisywanie hasła po sprawdzeniu numeru pesel i nr telefonu użytkownika.

## Rejestracja.java



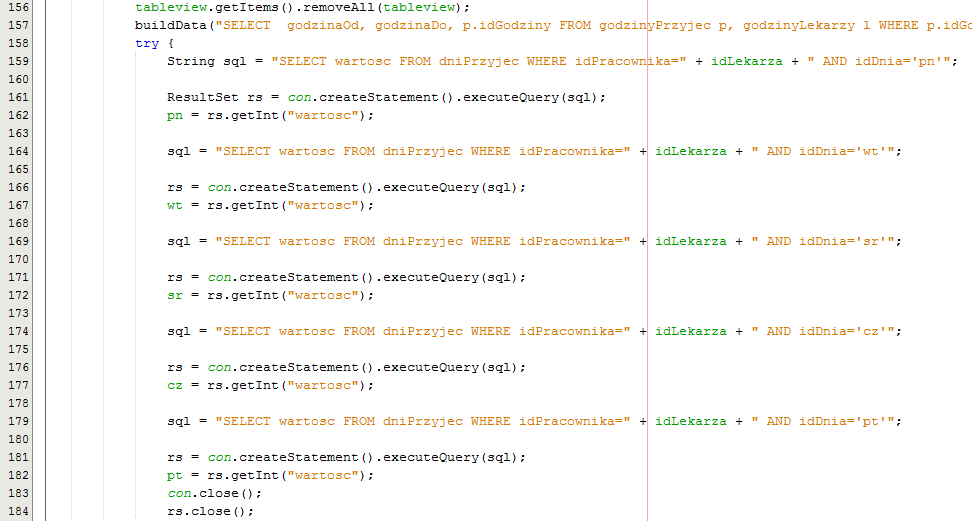
Funkcja odpowiedzialna za wygląd kalendarza, a dokładnie za wyłączenie dni oraz zaznaczenie ich na czerwono.



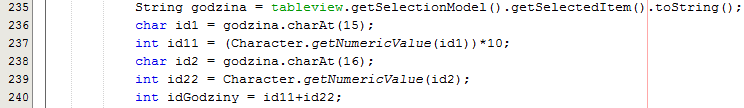
Uzależnienie comboLekarz od tego co będzie wyświetlane w comboSpec, poprzez pobranie od wybranego elementu z comboSpec.



Fragment kodu odpowiedzialny za pobranie odpowiedniego idLekarza. Następuje to poprzez zapisanie do string całej zawartości wybranej w comboLekarz, następnie wyodrębnienie pierwszego znaku, który jest numerem id a następnie przekonwertowanie z char na int.



Ten fragment pobiera z BD odpowiednie wartości dla dni, w których lekarz o zadanym id przyjmuje w przychodni. Dzięki temu możliwe jest odpowiednie zaznaczenie na czerwono dni w kalendarzu.

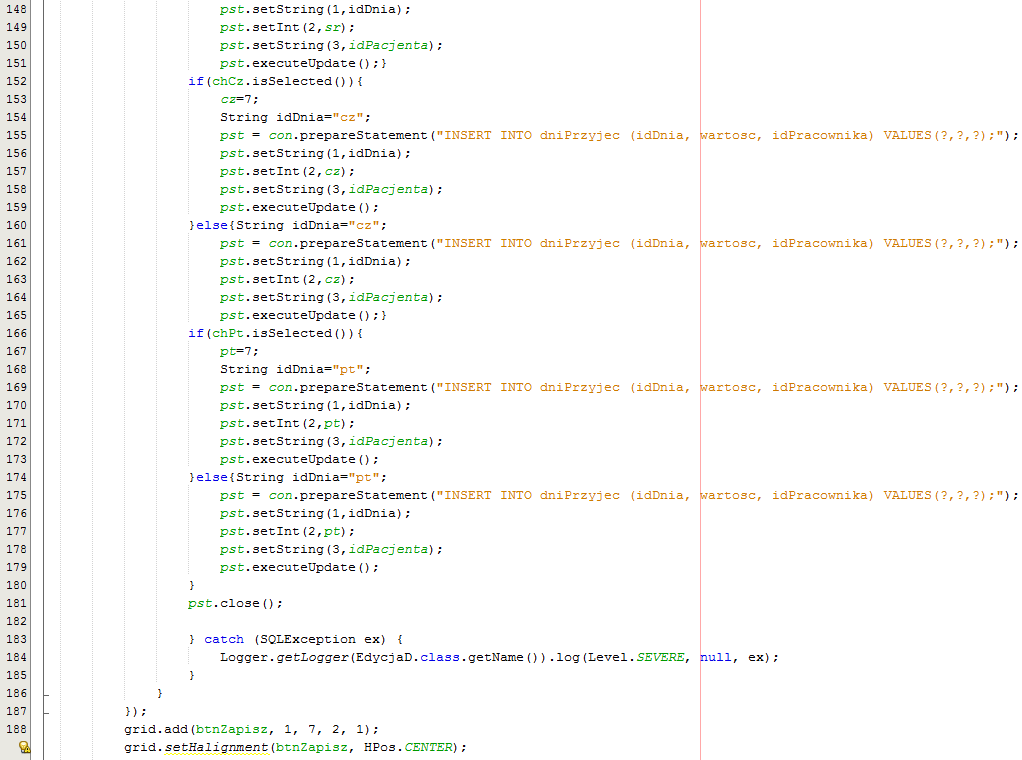


Pobierany do string jest zaznaczony wiersz z tableview, po czym znak na 15 i 16 miejscu przekształcany jest na cyfrę, następnie znak z 15 miejsca mnożony jest razy dziesięć, dzięki czemu po z sumowaniu otrzymujemy odpowiednią liczbę, która odzwierciedla id godziny, którą zaznaczyliśmy.

## Badanie.java

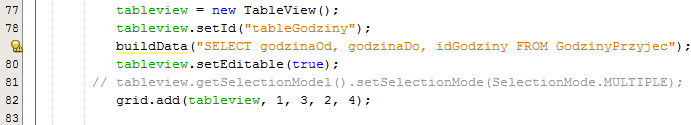
Wszystko w tej klasie odbywa się analogicznie do klasy Rejestracja.java

## EdycjaD.java

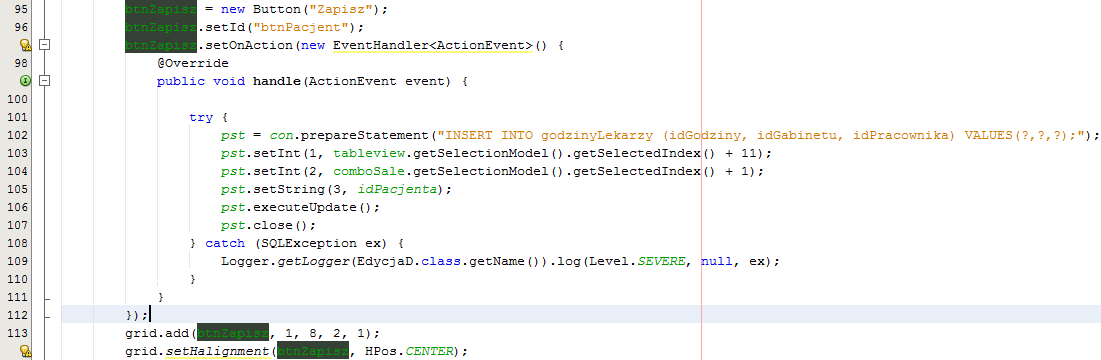


Ten przycisk ma za zadanie odczytanie z checkBoxów zaznaczonych dni tygodnia (tych, które wybrał lekarz). Następnie podczas sprawdzania w zależności czy warunek jest spełniony czy nie, uruchamiany jest odpowiedni fragment kodu zapisujący do BD wartość z dniem tygodnia.

## EdycjaG.java

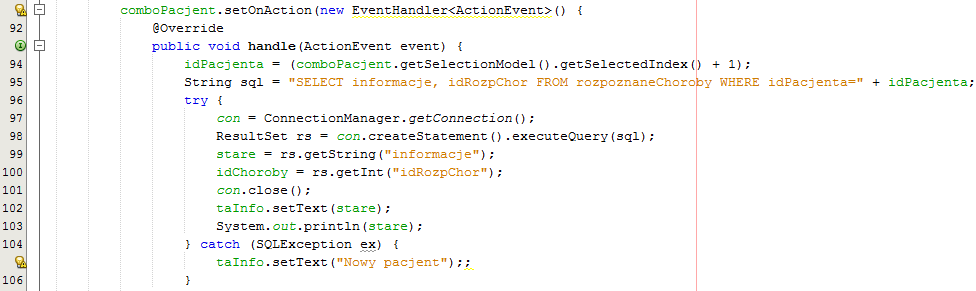


Wpisywanie do tabeli wszystkich możliwych godzin, w jakich otwarta jest przychodnia, aby lekarz mógł wybrać odpowiednie i zaakceptować, jako swoje godziny pracy.



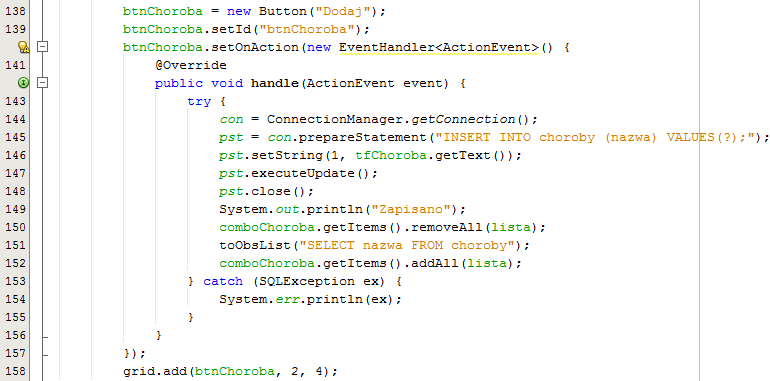
Przycisk, w którym po naciśnięciu wysyłane są dane do BD, z odpowiednim indeksem godziny, aby móc je następnie poprawnie wyświetlić w tabeli w klasie Rejestracja i Badanie.

## Historia.java



ComboBox, w którym po wybraniu odpowiedniego pacjenta, lekarzowi wyświetlone zostają informacje na jego temat, takie jak dodatkowe informacje historii choroby.

W przypadku gdy pacjent jest nowy, odpowiednio jest to zaznaczone.



Opcja dodania nowej choroby, do listy chorób, jeśli wcześniej ona nie istniała, a następnie odświeżenie comboBox, w celu pokazania wszystkich chorób.

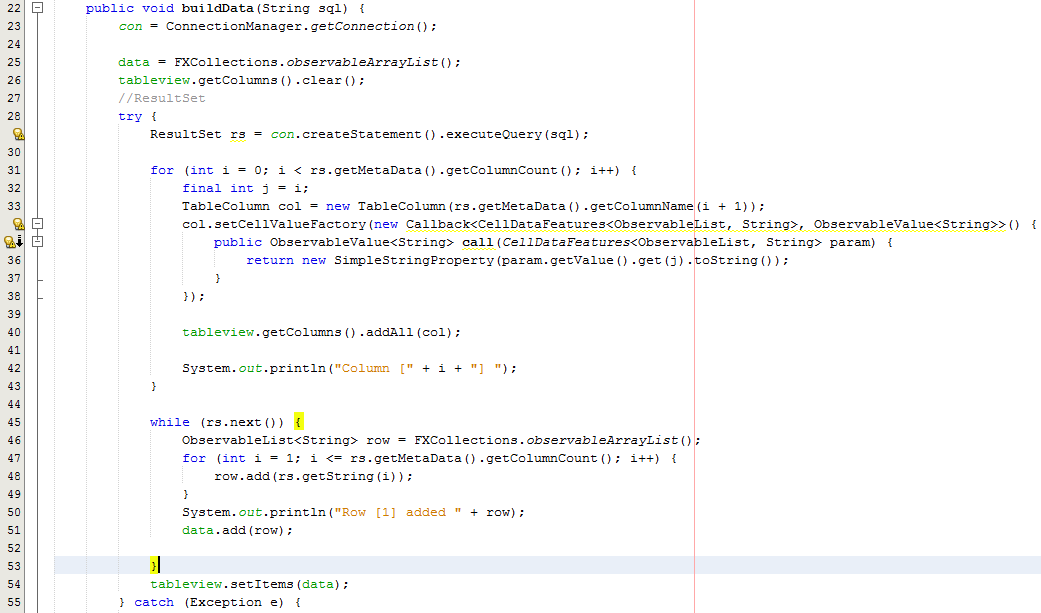
## 

A tutaj nadpisanie danych o dodatkowych informacjach pacjenta.

Tu następuje zapisanie do BD i dacie wizyty i chorobie, do odpowiedniego pacjenta.

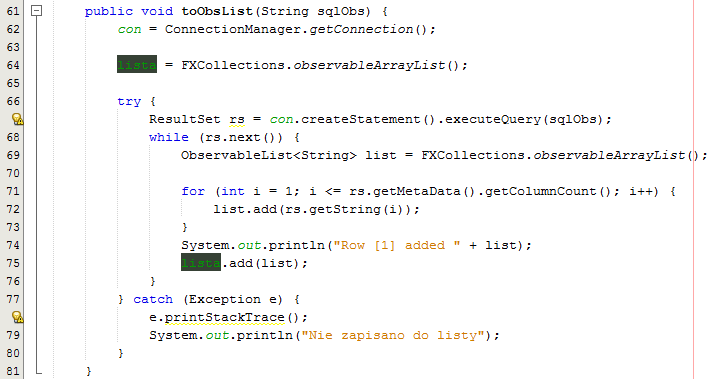
## Table.java

Metoda do dynamicznego tworzenia tabel, której wielkość uzależnia się od przekazanego w konstruktorze zapytania SQL.



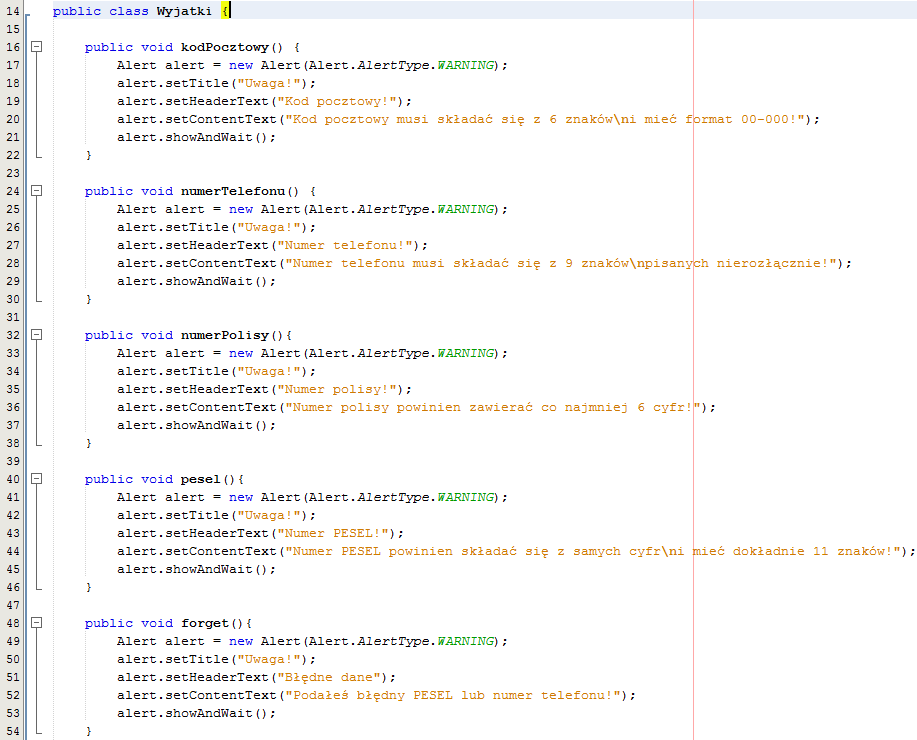
Pętlą while sprawdzam, czy w wierszach znajdują się dane, jeśli tak to są one przekazywane do observableArrayList. Po czym dodawane są do tabeli.

Pętla for przebiega po wszystkich nazwach kolumn zliczając ich ilość i jednocześnie zapisując ich nazwy, po czym wyświetlane są one w tableview.



Zmodyfikowana metoda buildData, w której przechwytuję jedynie dane z wierszy w bazie danych i opakowuje je w ObservableList, aby móc je potem wyświetlać np w combobox.

## Wyjatki.java



Klasa wyjątek służy do obsługi wyjątków w programie. Takich jak puste pola w formularzach, czy niepoprawnie wypełnione dane.

Efektem wystąpienia wyjątku jest wyskoczenie okienka popUp z alertem i odpowiednimi informacjami. Dla każdego wyjątku stworzona jest osobna metoda go obsługująca.