****

**Programowanie obiektowe**

*System biblioteczny 2*

Prowadzący: Autor:

mgr inż. Ewa Żesławska *Kacper Krowicki*

nr albumu: 117861

Kierunek: IiE/Ist/S/2020/21, grupa lab 1

Rzeszów 2022

Spis treści

[**1.** **Streszczenie** 3](#_Toc93070903)

[**2.** **Wstęp** 3](#_Toc93070904)

[**3.** **Wymagania funkcjonalne i niefunkcjonalne** 3](#_Toc93070905)

[**4.** **Projekt aplikacji** 4](#_Toc93070906)

[**5.** **Diagramy UML** 7](#_Toc93070907)

[**6.** **Opis strony użytkowej projektu** 9](#_Toc93070908)

[**7.** **Podsumowanie i wnioski** 13](#_Toc93070909)

[**8.** **Literatura** 14](#_Toc93070910)

1. **Streszczenie**

Sprawozdanie przedstawia kolejne etapy budowy aplikacji wspomagającej zarządzanie biblioteką: od postawienia wymagań, które ma spełniać aplikacja, poprzez projekt aplikacji i sekwencje czynności po działającą aplikację.

1. **Wstęp**

W życiu codziennym coraz częściej spotykamy się z zastosowaniem komputerów do wspomagania wykonywania różnorodnych czynności: zarządzanie, obsługa, sprzedaż, produkcja itd. W związku z tym wybór tematu projektu nie stwarzał problemu. Wystarczyło wejść do sklepu, urzędu, biblioteki żeby zobaczyć działanie różnych użytecznych aplikacji. Postanowiłem stworzyć aplikację, której zadaniem będzie wspomaganie zarządzania biblioteką. Celem projektu jest utworzenie aplikacji umożliwiającej użytkownikowi prowadzenie bazy danych posiadanych woluminów oraz zarejestrowanych czytelników, prowadzenie spisu woluminów wypożyczonych, a także wyświetlanie informacji o czytelnikach takich jak dane personalne oraz obecna ilość wypożyczonych woluminów.

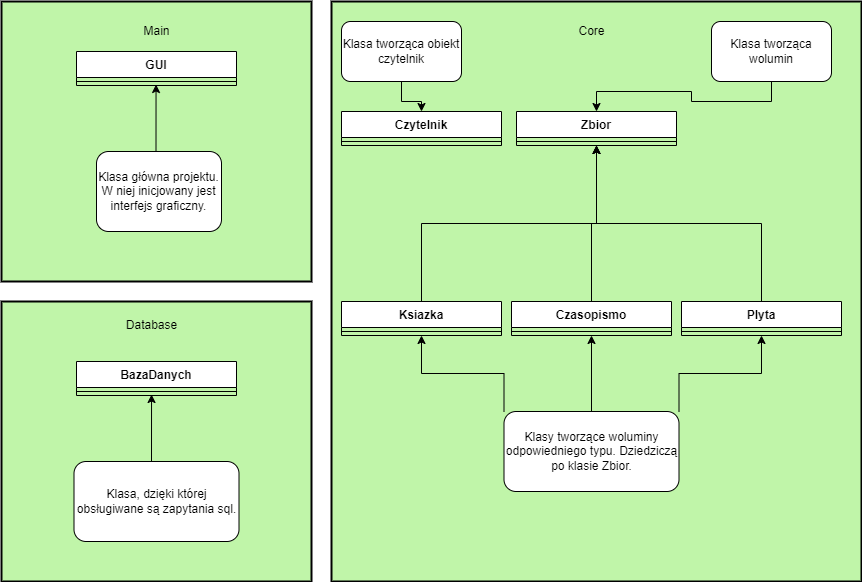
1. **Wymagania funkcjonalne i niefunkcjonalne**

Projektowany system powinien umożliwiać realizację następujących działań i posiadać poniżej wymienione cechy:

* Dodawanie/usuwanie czytelników
* Dodawanie/usuwanie woluminów
* Wyszukiwanie czytelników spośród wszystkich dostępnych na liście
* Wyszukiwanie woluminu spośród wszystkich dostępnych na liście
* Zatwierdzenie zwrotu woluminu, który ma status wypożyczenia u danego czytelnika
* Zatwierdzenie wypożyczenia woluminu dla danego czytelnika
* Wyświetlanie informacji o stanie konta czytelnika (ilość wypożyczonych woluminów)

1. **Projekt aplikacji**

Aplikacja to są dane i działania wykonywane na danych. W tym rozdziale zostaną opisane klasy, które będą reprezentować dane i działania wykonywane na nich.



Rysunek Diagram klas

**Zbiór**

Dane o woluminach będą przechowywane w Klasie Zbior. Ale okazuje się że operacje na woluminach różnych typów są podobne. Korzystając z możliwości jakie daje programowanie obiektowe (dziedziczenie) wszystkie wspólne działania wykonywane woluminach umieścimy w klasie nadrzędnej Zbior. Klasa bazowa również narzuci strukturę danych, w której będą przechowywane dane. Będzie to lista. Klasa Zbior posiada następujące atrybuty:

* tytul – tytuł woluminu
* nrEw – numer ewidencyjny woluminu
* keyWord – słowo kluczowe woluminu
* wypozyczona – informacja o statusie wypożyczenia danego woluminu
* przedmioty – statyczna lista wszystkich woluminów.

Klasa Zbior posiada metody wspólne wraz z klasami które po niej dziedziczą:

* getTytul() – zwraca tytuł danego woluminu,
* setTytul() – ustawia tytuł dla danego woluminu,
* getNrEw() – zwraca numer ewidencyjny danego woluminu,
* setNrEw() – ustawia numer ewidencyjny danego woluminu,
* getKeyWord() – zwraca słowo kluczowe danego woluminu,
* setKeyWord() – ustawia słowo kluczowe dla danego woluminu,
* getPrzedmioty() – zwraca liste wszystkich woluminów w postaci tablicy dwuwymiarowej.
* toString() – zwraca opis obiektu.

Metody wykorzystywane jedynie w klasie Zbior:

* isWypozyczona() – zwraca wartość true jeżeli wolumin jest wypożyczony lub wartość false w przeciwnym wypadku.
* setWypozyczona() – ustawia status wypożyczenia dla woluminu,
* getPrzedmiotyList() – zwraca liste wszystkich woluminów,
* delWolumin() – usuwa wolumin z listy wszystkich dostępnych woluminów.

Klasy Ksiazka, Czasopismo, Plyta są klasami pochodnymi w stosunku do klasy Zbior. Klasy pochodne będą zawierać poniższe atrybuty wraz z metodami get oraz set.

**Klasa Ksiazka:**

* ksiazki – statyczna lista zawierająca wszystkie książki,
* autor – autor książki
* rok – rok wydania książki

**Klasa Czasopismo:**

* Czasopisma – statyczna lista zawierająca wszystkie czasopisma,
* numer – numer czasopisma
* rok – rok wydania czasopisma
* wydawnictwo – wydawnictwo czasopisma

**Klasa Plyta:**

* Plyty – statyczna lista zawierająca wszystkie płyty
* wykonawca – wykonawca płyty

**Klasa Czytelnik**

Klasa Czytelnik przechowuje informacje o ilości zarejestrowanych czytelników, oraz informacje o statusie ich kont. W tym celu posiada poniższe atrybuty wraz z metodami get oraz set:

* imie – imię czytelnika, który zostaje zarejestrowany w bibliotece
* nazwisko – nazwisko czytelnika, który zostaje zarejestrowany w bibliotece
* czytelnicy – statyczna lista, do której zostaje przypisany każdy nowo utworzony czytelnik
* wypozyczone – lista wypożyczonych woluminów, indywidualna dla każdego obiektu czytelnik

**Klasa BazaDanych**

Klasa tworząca połączenie z bazą danych postgresql. Zapewnia metody, dzięki którym realizowane są zapytania sql.

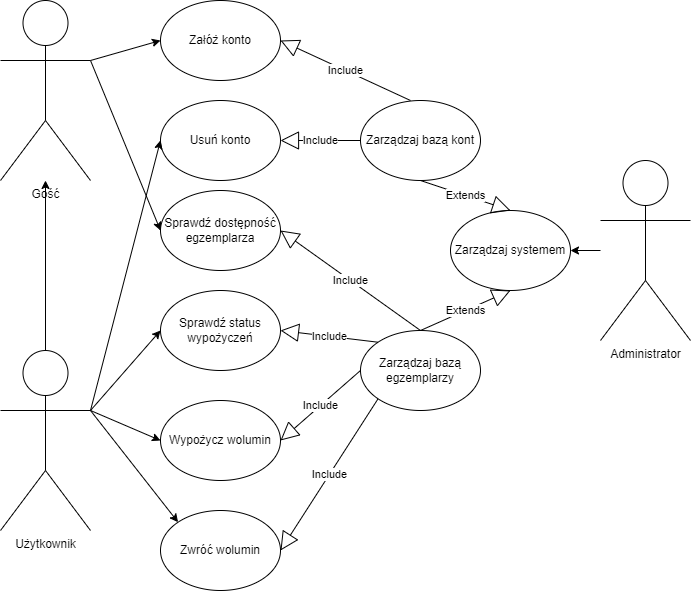
**Klasa GUI**

Klasa GUI jest klasą główną programu. Tworzy interfejs graficzny, oraz zapewnia funkcjonalności związane z interakcją użytkownik – program. Posiada metody umożliwiające prawidłowe działanie programu tj. :

* actionPerformed() – zawiera obsługę zdarzeń związaną z elementami interfejsu użytkownika,
* keyReleased() – pozwala na wywołanie metody wyszukującej rekord w tabeli w momencie naciśnięcia dowolnego przycisku przez użytkownika,
* dodajCzytelnik() – tworzy obiekt czytelnik na podstawie danych wprowadzonych przez użytkownika, dodaje go do listy
* pokazCzytelnik() – wyświetla we wskazanej we wskazanej jako parametr liście wszystkich zarejestrowanych czytelników
* usunCzytelnik() – usuwa czytelnika wybranego przez użytkownika z listy zarejestrowanych
* dodajWolumin() – dodaje wolumin do listy wszystkich woluminów
* pokazWolumin() – wyświetla we wskazanej liście wszystkie dostępne woluminy
* usunWolumin() – usuwa wskazany przez użytkownika wolumin z listy wszystkich dostępnych
* ustawComboWypo() – ustawia listę wyboru czytelników oraz dostępnych dla nich książek do wypożyczenia
* dodajWypo() – sprawdza czy wybrany przez użytkownika czytelnik ma możliwość wypożyczenia danego woluminu oraz w przypadku pozytywnym przypisuję wolumin do czytelnika
* ustawComboZwrot() – ustawia listę wyboru wszystkich czytelników,
* ustawComboZwrotDane() – ustawia listę wyboru woluminów do zwrotu dla wskazanego przez użytkownika czytelnika
* dodajZwrot() – zatwierdza zwrot wybranego woluminu dla wybranego czytelnika
* szukajCzyt() – przeszukuje listę wszystkich czytelników pod kontem informacji podanych przez użytkownika, następnie aktualizuje tabele z czytelnikami wyświetlając tylko pasujące rekordy
* szukajWolum() - przeszukuje listę wszystkich woluminów pod kontem informacji podanych przez użytkownika, następnie aktualizuje tabele z woluminami wyświetlając tylko pasujące rekordy
* rysujTabelaWolum() – uzupełnia tabele danymi o woluminach
* rysujTabelaCzyt() – uzupełnia tabele danymi o czytelnikach

1. **Diagramy UML**
   1. **Diagram przypadków użycia**

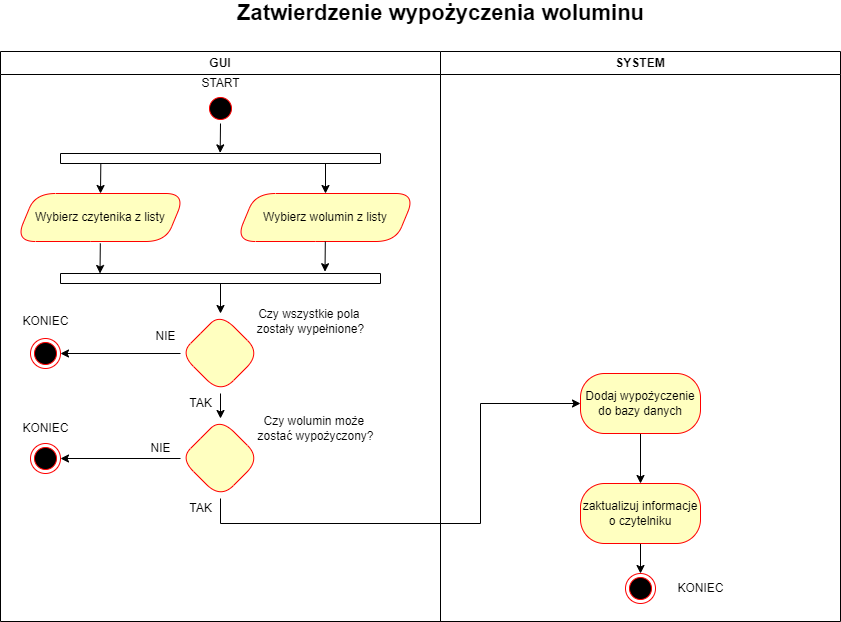
Na rysunku 2 przedstawiono diagram przypadków użycia projektowanej aplikacji.

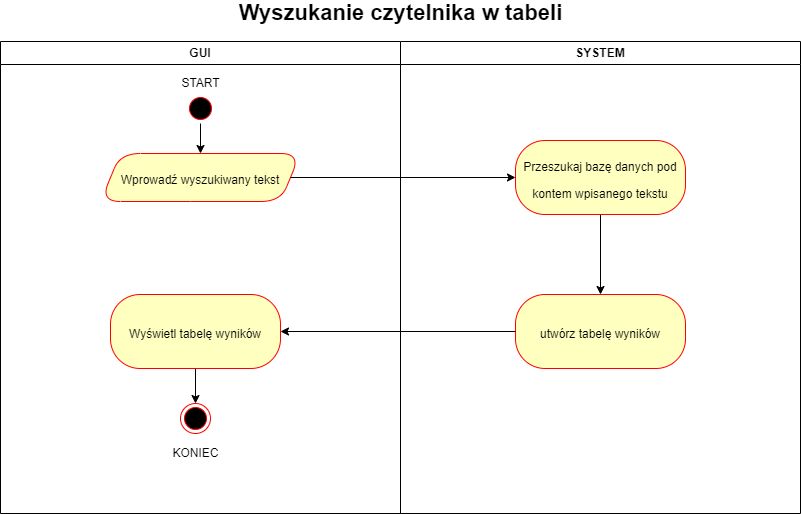
Rysunek Diagram przypadków użycia

* 1. **Diagram sekwencji czynności**

Zadaniem tego rozdziału jest pokazanie sekwencji kroków jakie są wykonywane przez modelowany fragment systemu. Oto przykładowe działania, które trzeba wykonać aby zatwierdzić wypożyczenie (Rysunek 3) lub wyszukać użytkownika w tabeli (Rysunek 4).



Rysunek Zatwierdzenie wypożyczenia



Rysunek Wyszukanie czytelnika w tabeli

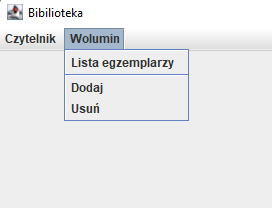
1. **Opis strony użytkowej projektu**

Po uruchomieniu aplikacji pokazuje się puste okno z menu wyboru. Elementy menu są pogrupowane w zakładkach – nazwanych podobnie jak nazwa przechowywanych obiektów. W każdej zakładce są umieszczone elementy menu, dzięki którym możemy nawigować pomiędzy obszarami pracy, np. wchodząc w zakładkę Czytelnik → lista czytelników (Rysunek 7), pojawi się panel wyświetlający listę czytelników. Zakładka Czytelnik posiada również opcje służące do dodawania i usuwania czytelników, a także opcje zwrotu oraz wypożyczenia woluminu czytelnikowi. Aby zatwierdzić wypożyczenie lub zwrot musi być dodany przynajmniej jeden czytelnik oraz jeden wolumin. Wypożyczyć można tylko wolumin, który aktualnie nie jest wypożyczony. Wchodząc w zakładkę Wolumin → lista egzemplarzy (Rysunek 8), wyświetlona zostanie tabela wszystkich woluminów wcześniej utworzonych w menu Wolumin → dodaj (Rysunek 9). Użytkownik ma do dyspozycji listę rozwijalną, z której może wybrać typ wyświetlanych woluminów. Jest to odpowiednio wszystkie, książka, czasopismo oraz płyta. Wolumin, jeżeli nie jest wypożyczony, może zostać usunięty w menu Wolumin → usuń (Rysunek 10).

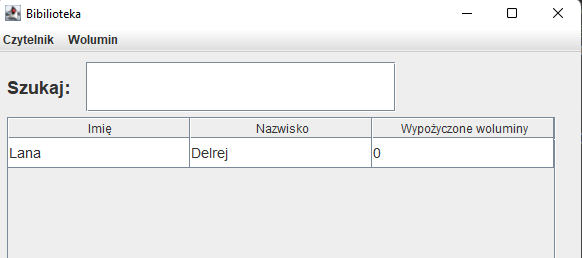
Obraz zawierający stół

Opis wygenerowany automatycznie

Rysunek Lista menu Czytelnik



Rysunek Lista menu Wolumin

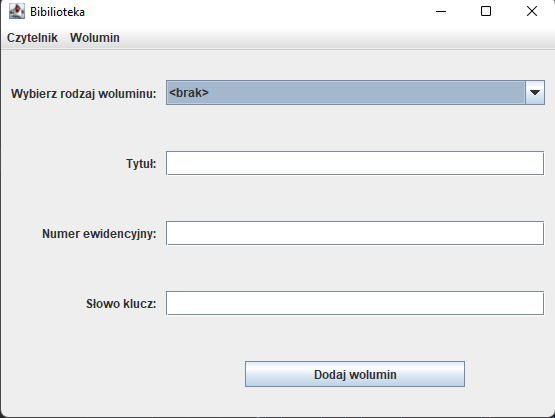


Rysunek Menu lista czytelników

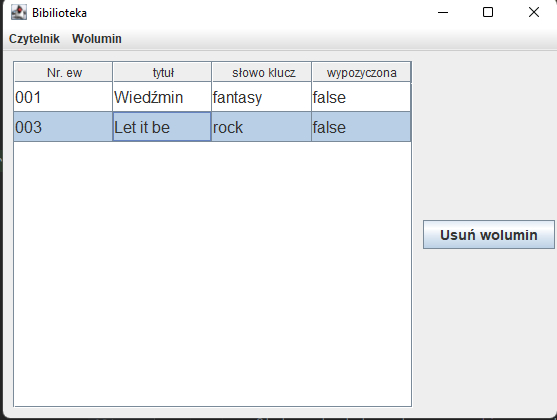
Obraz zawierający stół

Opis wygenerowany automatycznie

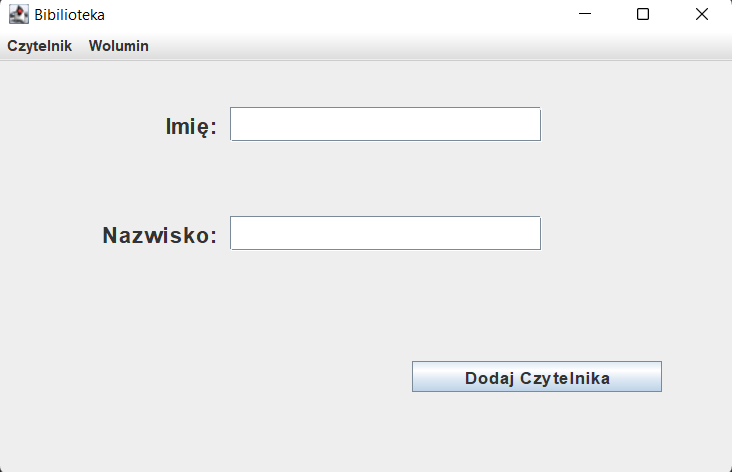
Rysunek Menu lista egzemplarzy



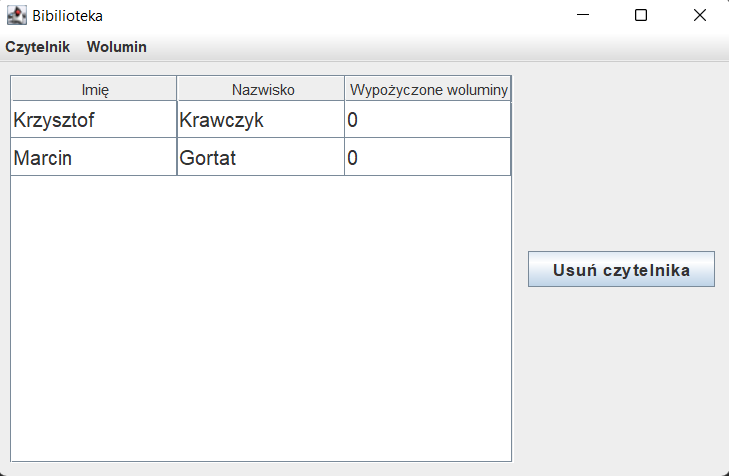
Rysunek Menu dodaj wolumin



Rysunek Menu usuń wolumin



Rysunek 11 Menu dodaj czytelnika



Rysunek 12 Menu usuń czytelnika

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 13 Menu zatwierdź zwrot

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 14 menu zatwierdź wypożyczenie

1. **Podsumowanie i wnioski**

Projekt udało się zrealizować zgodnie z postawionymi wymaganiami. Od strony użytkowej można poprawić następujące elementy:

* brak możliwość wyszukiwania osób lub woluminów do usunięcia,
* brak możliwości sortowania tabeli,

W przyszłości istnieje możliwość implementacji systemu logowania, dzięki któremu czytelnicy będą mieli dostęp do informacji o swoim koncie oraz o egzemplarzach dostępnych w bibliotece np. z poziomu własnego komputera.

Pliki źródłowe programu znajdują się w repozytorium pod adresem: [repozytorium](https://github.com/kmkcc2/ProgramowanieObiektowe/tree/main/ProjektZaliczenie). Będą tam dostępne do dnia 14.03.2022r. Baza danych tymczasowo została utworzona na serwerze lokalnym „localhost”.

1. **Literatura**
2. G. Cornell, C. S. Horstmann, Java. Podstawy, Helion, 2008.
3. B. Eckel, Thinking in Java. Edycja polska (wydanie IV), Helion, 2006.
4. M. Lis, Java. Ćwiczenia praktyczne, Helion, 2006.
5. H. Schildt, Java. Kompendium programisty, Helion, 2005.
6. Dokumentacja JavaSE7, <https://docs.oracle.com/javase/7/docs/api/> (data dostępu 03.01.2022)
7. Tutoriale języka Java dostępne na stronie <https://www.w3schools.com/java/> (data dostępu 03.01.2022)