

**Dokumentacja projektu**

Przedmiot:**Programowanie obiektowe**

**Tytuł projektu:Prosta aplikacja bankowa**

Prowadzący: Wykonawca:

mgr inż. Ewa Żesławska Erwin Kulig

w64132

3IIZ/2020-GP02

Rzeszów 2022

**Spis treści**

[**1.** **Opis założeń projektu** 3](#_Toc73798610)

[**2.** **Specyfikacja wymagań** 3](#_Toc73798611)

[**3.** **Diagram przypadków użycia** 3](#_Toc73798612)

[**4.** **Harmonogram realizacji projektu (diagram Gantta)** 4](#_Toc73798613)

[**5.** **Opis techniczny projektu** 5](#_Toc73798614)

[**6.** **Prezentacja warstwy użytkowej projektu** 5](#_Toc73798615)

[**7.** **Raporty z testów jednostkowych** 6](#_Toc73798616)

[**8.** **System kontroli wersji** 7](#_Toc73798617)

[**9.** **Literatura** 7](#_Toc73798618)

[**10.** **Dokumentacja projektu** 7](#_Toc73798619)

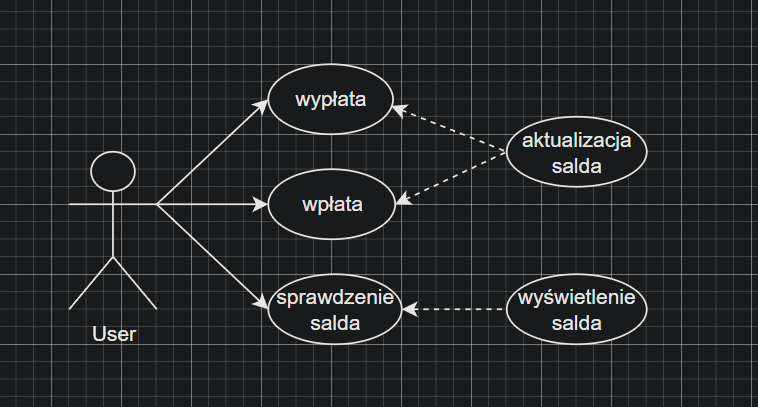
1. **Opis założeń projektu**

W obecnych czasach niemal każdy posiada swoje własne konto bankowe, gdzie gromadzi swoje fundusze. Wraz ze stałym postępem cyfryzacji odchodzimy od standardowych bankomatów, i korzystamy z aplikacji bankowych w wersji mobilnej. Są jednak pewne czynności, których nie jesteśmy wykonać przez internet np. wypłata gotówki z konta bankowego. Założeniem projektu jest stworzenie prostej i przejrzystej aplikacji bankowej, zwłaszcza dla osób starszych, które nie są tak obeznane z nowoczesnymi metodami bankowości, zawierające jej podstawowe funkcje takie jak: sprawdzenie salda, depozyt, wypłata środków z konta bankowego.

1. **Specyfikacja wymagań**
   1. **Wymagania funkcjonalne**

* Użytkownik ma możliwość wpłacenia dowolnej kwoty, jak i wypłacenia
* Użytkownik ma możliwość sprawdzenia salda na koncie w dowolnym momencie
* W momencie wybrania złej opcji, lub nie zaznaczenia żadnej wyświetlany jest komunikat o błędzie
* Użytkownik może zobaczyć jaka transakcja została wykonana ostatnio wybierając opcje „Poprzednia transakcja”
  1. **Wymagania niefunkcjonalne**
* Aplikacja jest bardzo prosta w użyciu i przejrzysta
* Aplikacja tworzona jest w języku Java w środowisku IntelliJ
* Aplikacja nie potrzebuje połączenia z internetem

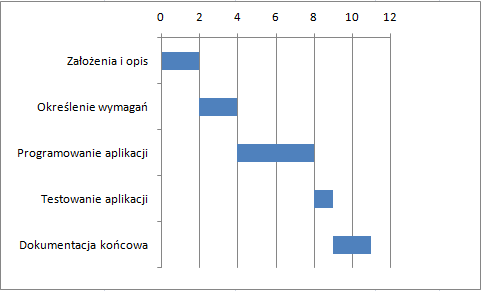
1. **Diagram przypadków użycia**

****

Rysunek 1. Diagram przypadków użycia

1. **Harmonogram realizacji projektu (diagram Gantta)**

Poniżej zamieszono harmonogram realizacji projektu.



Rysunek 2. Diagram Gantta

1. **Opis techniczny projektu**

Należy umieścić informacje odnośnie wykorzystywanego języka, narzędzi oraz minimalnych wymagań sprzętowych.

Np.:

* Środowisko programistyczne Javy: Java JDK 17
* Środowisko programistyczne: InteliJ 2021.2.2
* Urządzenie z system operacyjnym Windows 10

**6.Prezentacja warstwy użytkowej projektu**

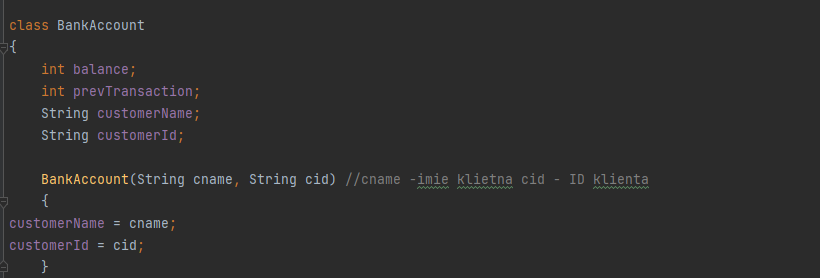
Stworzenie klasy **BankAccount** (Rysunek 1) która posiada 4 zmienne:

balance – czyli pieniądze które znajdują się na koncie

prevTransaction – wskazuje czy ostatnia kwota była wypłacona czy wypłacona

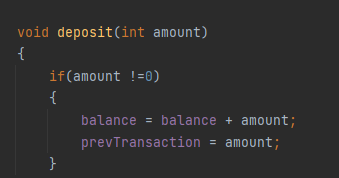
customerName, customerId – Imię oraz identyfikator klienta

Konstruktor BankAccount – który zawiera dwie wartości cname i cid które będą potrzebne do wyświetlania wiadomości powitalnej

****

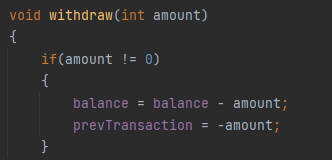
Rysunek 1. Klasa BankAccout wraz z zmiennymi

Pierwsza metoda która została stworzona, jest metoda deposit (Rysunek 2) która przyjmuje jedną zmienną całkowitą jako parametr amount, ponieważ gdy zamierzamy wypłacić jakąś kwotę program przyjmie tę kwotę jako parametr. Następnie zapiasana jest instrukcja warunkowa if !=0 aby nie wykonywać niepotrzebnie operacji jeśli wpisana wartość jest równa 0. Następnie zwiększana zmiennca balance o kwotę zawartą w zmiennej amount.



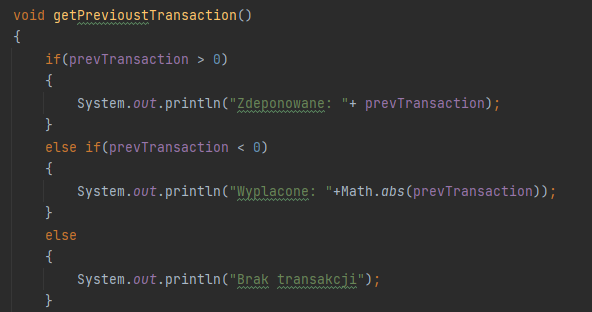
Rysunek 2. Metoda deposit

Kolejna jest metoda withdraw (Rysunek 3) która odpowiada za wypłatę która również przyjmuje zmienną amount jako parametr, ponieważ musimy wiedzieć jaka kwota ma zostać wypłacona. Następnie aktualizowana jest zmienna balance o wypłaconą kwotę. Znak ‘-‘ przy zmiennej amount oznacza, że jest to wypłata.



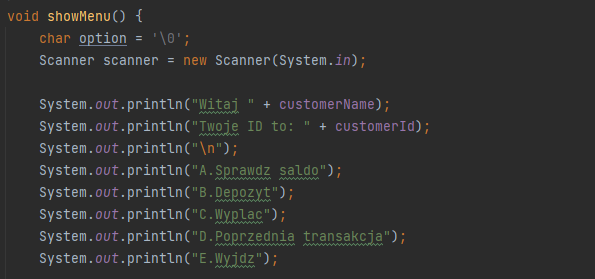
Rysunek 3. Metoda withdraw

Metoda getPreviousTransaction (Rysunek 4) ma za zadanie zwrócić informacje o wpłacie pewnej kwoty albo wypłacie, aby to zrobić sprawdzany jest warunek > 0 co oznacza kwota jest wpłacona prawidłowo. W przypadku wypłacania użyta została metoda Math.abs która z dowolnej wartości daje nam wartość dodatnią np. gdy wypłacamy kwotę 1000 zwrócona zostanie informacja o wypłacie kwoty 1000, a nie -1000. Else w przypadku gdy żaden z warunków nie pasuje wyświetla informację o błędzie.



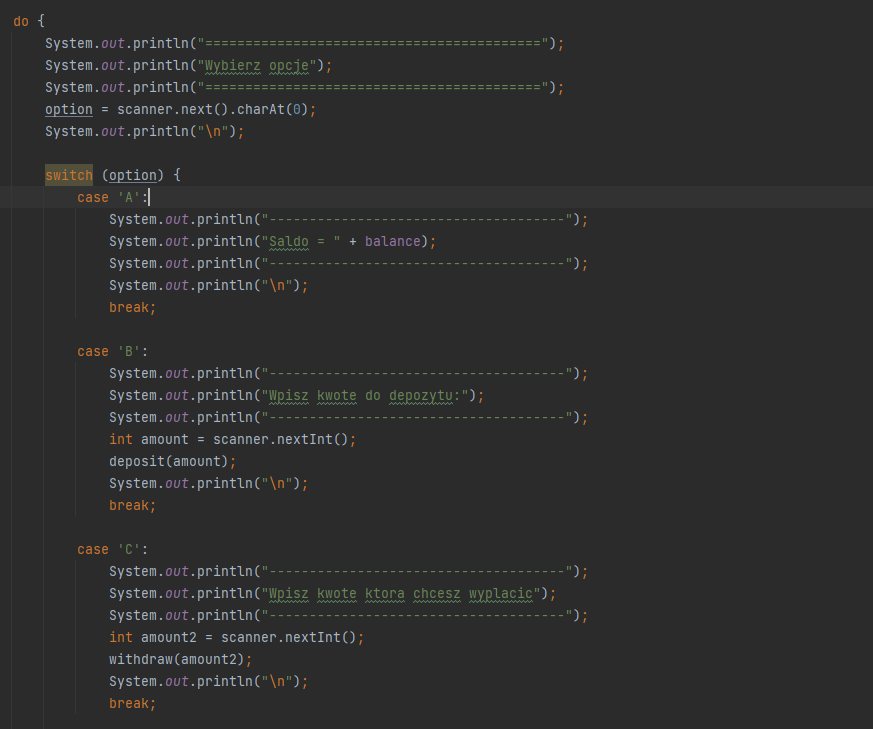
Rysunek 4. Metoda getPreviousTransaction

W metodzie showMenu wyświetlane jest menu (Rysunek 5) wraz z wiadomością powitalną oraz z listą opcji które użytkownik ma do wyboru podczas korzystania z aplikacji.

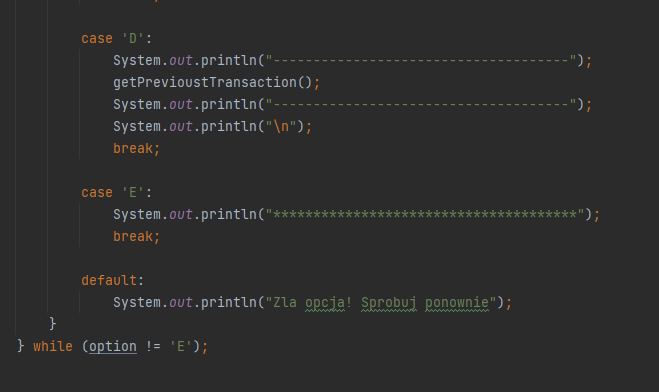


Rysunek 5. Metoda showMenu

Na koniec każda z dostępnych operacji została zaprogramowana za pomocą instrukcji switch case i klasy Scanner (Rysunek 6.1 i 6.2). Odbywa się to przez pobranie wartości przez konsolę i przekazanie jej konkretnej metodzie która odpowiada za konkretną transakcję.



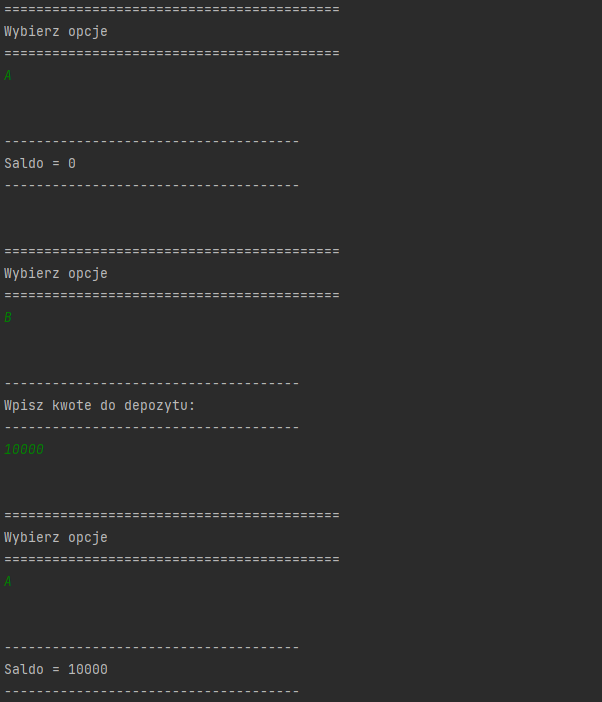
Rysunek 6.1 Switch case



Rysunek 6.2 Switch case

1. **Raporty z testów jednostkowych**

Został przeprowadzony test w którym najpierw wpłacona została pewna kwota, a następnie sprawdzony został rzeczywisty stan depozytu:



1. **System kontroli wersji**

Plik z kodem znajduje się pod adresem https://github.com/w64132/projekt.git

1. **Literatura**
2. Java - programowanie obiektowe w 3h https://youtu.be/OvY0f-IWlos
3. Podstawowe wejście - Klasa Scanner https://javastart.pl/baza-wiedzy/java-podstawy-jezyka/podstawowe-wejscie-wyjscie
4. Git tutorial for Beginners - https://youtu.be/DVRQoVRzMIY
5. **Dokumentacja projektu**

Do projektu stworzono dołączono komenatrze wraz z odpowiadającymi im rysunkami.