Inżynieria Oprogramowania

Opis pakietów, klas, metod

aplikacji

PC Banking

Autorzy projektu
- Adrian Dajakaj
- Maksym Ovsiienko
- Oleksii Sytnik

Pakiet

AuthentificationAndRegistration

Odpowiada za proces logowania oraz rejestracji użytkownika

1. Klasa AuthentificationScreen
    a. Wraz z AuthentificationScreen.form dostarcza interfejs GUI odpowiedzialny za logowanie oraz przejście do interfejsu rejestracji.
    b. Interfejs dostarczony przez klasę zawiera
        i. pola tekstowe odpowiednio na „username” (nazwę użytkownika) i „password” (hasło)
        ii. przyciski „Sign in” (zaloguj), „Sign up” (zarejestruj) oraz „Exit” (zamknij program)
    b. Przeprowadza weryfikację istnienia wprowadzonego przez użytkownika loginu oraz hasła w bazie danych przy pomocy metody Database.verifyUser()
2. Klasa RegistrationScreen1
    a. Wraz z RegistrationScreen1.form dostarcza interfejs GUI odpowiedzialny za pierwszy panel rejestracji- dane logowania przy pomocy metody CreateScreen()
    b. Interfejs GUI zawiera
        i. pole tekstowe na „username” (nazwa użytkownika)
        ii. pole tekstowe na „Email” (adres e-mail)
        iii. pole tekstowe na „password” (hasło)
        iv. pole tekstowe na „Enter again password” (potwierdzenie hasła wprowadzonego powyżej)
        v. informacje o wymaganiach dotyczących hasła
            1. Passwords match
            2. At least 12 characters
            3. At least one letter
            4. At least one digit
            5. At least one special character
        vi. Przyciski „submit” (zatwierdź dane), „return (cofnij), „exit” (wyjdź)
        vii. Panel „Automatic log out” wraz z zegarem odliczającym czas- odpowiadają za automatyczne wylogowanie, mechanizm automatycznego wylogowania korzysta z klas AppTimer oraz MouseAction
    c. W klasie sprawdzana jest poprawność wprowadzanych danych, przy wykorzystaniu funkcji z klasy Database weryfikowana jest poprawność nazwy użytkownika oraz unikalność nazwy użytkownika

    d. Jeżeli wprowadzone dane są niepoprawne, to wyświetlane są stosowne komunikaty
3. Klasa RegistrationScreen2
    a. Wraz z RegistrationScreen2.form dostarcza interfejs GUI odpowiedzialny za drugi panel rejestracji- dane osobowe użytkownika przy pomocy metody CreateScreen()
    b. Interfejs GUI zawiera
        i. Pole tekstowe na „First Name” (imię użytk.)
        ii. Pole tekstowe na „Last Name”) (nazwisko)
        iii. Pole wyboru „Sex” (płeć użytkownika)
        iv. Pole tekstowe na „City” (miejscowość)
        v. Pole tekstowe na Post code (kod pocztowy)
        vi. Pole tekstowe na „Street” (ulica)
        vii. Pole tekstowe na „Street nr” (nr. Domu)
        viii. Pole tekstowe na „PESEL” (nr. PESEL)
        ix. Pole tekstowe na „Phone number” (nr. Tel.)
        x. Przyciski „submit” (zatwierdź dane), „return (cofnij), „exit” (wyjdź)
        xi. Panel „Automatic log out” wraz z zegarem odliczającym czas- odpowiadają za automatyczne wylogowanie, mechanizm automatycznego wylogowania korzysta z klas AppTimer oraz MouseAction
    c. W klasie sprawdzana jest poprawność wprowadzonych danych i w razie potrzeby wyświetlane komunikaty o niepoprawności
4. Klasa EmailVerificationScreen
    a. Wraz z EmailVerificationScreen.form dostarcza interfejs GUI odpowiedzialny za weryfikację adresu e-mail przy pomocy metody CreateScreen()
    b. Interfejs GUI zawiera
        i. Pole tekstowe na 6-cyfrowy kod weryfikacyjny przesłany na adres e-mail
        ii. Przyciski „submit” (zatwierdź dane), „return (cofnij), „exit” (wyjdź)
        iii. Panel „Automatic log out” wraz z zegarem odliczającym czas- odpowiadają za automatyczne wylogowanie, mechanizm automatycznego wylogowania korzysta z klas AppTimer oraz MouseAction
    c. Przy pomocy metody generateCode(6) generowany jest 6-cyfrowy kod weryfikacyjny
    d. Kod jest wysyłany na adres e-mail użytkownika przy pomocy funkcji JavaMail.SendMail(user.email, code)
5. Klasa CreatedAccountAndCardNumberScreen
    a. Wraz z CreatedAccountAndCardNumberScreen.form dostarcza interfejs GUI przy pomocy metody CreateScreen()
    b. Interfejs GUI zawiera
        i. Informacje o kontach, jakie zostaną przypisane do użytkownika

ii. Wygenerowane indywidualne numery kont: głównego i oszczędnościowego
iii. Wygenerowany numer karty
iv. Wygenerowany numer PIN
v. Wygenerowany numer AppCode (kod aplikacji)
vi. Przyciski „submit" (zatwierdź dane), „return (cofnij), „exit" (wyjdź)
vii. Panel „Automatic log out" wraz z zegarem odliczającym czas- odpowiadają za automatyczne wylogowanie, mechanizm automatycznego wylogowania korzysta z klas AppTimer oraz MouseAction

c. Metoda generateAccountNumber() generuje 26-cyfrowe unikalne numery kont: głównego i oszczędnościowego, z kodem IBAN domyślnie ustawionym na „PL"
d. Metoda generateCardNumber() generuje 16-cyfrowy numer karty bankowej użytkownika
e. Metoda generatePIN() generuje 4- cyfrowy PIN do karty oraz kod aplikacji (również 4-cyfrowy)

Pakiet

CreditCard

Odpowiada za płatności kartą debetową użytkownika

1. Klasa PayWithCardScreen1
   a. Wraz z PayWithCardScreen1.form dostarcza interfejs GUI pierwszego ekranu płatności kartą przy pomocy metody CreateScreen()
   b. Interfejs GUI zawiera
      i. Pole tekstowe „Card Number"- odpowiada za podanie numeru karty
      ii. Pole tekstowe „PIN"- odpowiada za wprowadzenie numeru PIN karty
      iii. Przyciski „submit" (zatwierdź dane), „exit" (wyjdź)
      iv. Panel „Automatic log out" wraz z zegarem odliczającym czas- odpowiadają za automatyczne wylogowanie, mechanizm automatycznego wylogowania korzysta z klas AppTimer oraz MouseAction
   c. Sprawdzana jest poprawność podanych przez użytkownika danych i wyświetlane stosowne komunikaty
2. Klasa PayWithCardScreen2

a. Wraz z PayWithCardScreen2.form dostarcza interfejs GUI drugiego ekranu płatności kartą przy pomocy metody CreateScreen()
b. Interfejs GUI zawiera
    i. Pole tekstowe „Payment Amount”- odpowiada za podanie kwoty, jaką użytkownik chce wypłacić
    ii. Przyciski „submit” (zatwierdź dane), „exit” (wyjdź)
    iii. Panel „Automatic log out” wraz z zegarem odliczającym czas- odpowiadają za automatyczne wylogowanie, mechanizm automatycznego wylogowania korzysta z klas AppTimer oraz MouseAction
c. Sprawdzana jest dostępność wpisanej przez użytkownika kwoty na koncie użytkownika, jeśli kwota jest za duża (przekracza stan konta), wyświetlany jest stosowny komunikat i płatność kartą jest uniemożliwiona

---

Pakiet

Credits

Odpowiada za obsługę zaciągania pożyczki gotówkowej

1. Klasa Credit
    a. Wraz z Credit.form dostarcza interfejs GUI ekranu zaciągania pożyczki gotówkowej przy pomocy konstruktora i metody CreateScreen()
    b. Interfejs GUI zawiera
        i. Wyświetlony stan konta użytkownika
        ii. Wyświetlony numer konta użytkownika
        iii. Pole tekstowe na wpisanie kwoty, jaką użytkownik chce pożyczyć
        iv. Pole tekstowe na liczbę lat, na jaką pożyczka ma zostać zaciągnięta
        v. Informację o 5% oprocentowaniu kredytu i pole wyboru, w którym użytkownik określa czy zgadza się z warunkami kredytu
        vi. Przyciski „Take new credit” do potwierdzenia chęci wzięcia kredytu
        vii. Poniżej informacje o kwocie już spłaconej przez użytkownika oraz kwocie pozostałej do spłaty oraz możliwość zapłaty raty kredytu przy pomocy przycisku „Pay debt”
        viii. Panel „Automatic log out” wraz z zegarem odliczającym czas- odpowiadają za automatyczne wylogowanie,

mechanizm automatycznego wylogowania korzysta z klas AppTimer oraz MouseAction

c. Metoda hasEnoughMoney() sprawdza czy użytkownik ma na koncie wystarczającą kwotę, aby opłacić ratę kredytu
d. Metoda isProvidedDataIsValid() sprawdza poprawność wprowadzonych przez użytkownika danych- czy wpisane wartości są liczbami
e. Metoda convertStringToDate() konwertuje datę typu String podaną na wejściu na typ Date
f. Metoda convertDateToString() konwertuje podaną na wejściu datę typu Date na typ String
g. Metoda diffOfYear() zwraca różnicę między datą dzisiejszą a datą wzięcia kredytu
h. Metoda checkDebt() zwraca informację o kwocie pozostałej do spłaty

---

Pakiet

Settings

Odpowiada za ustawienia konta użytkownika

2. Klasa ChangeDataScreen
    a. Wraz z ChangeDataScreen.form dostarcza interfejs GUI ekranu zmiany danych osobowych
    b. W przypadku błędnych danych wyświetla stosowne komunikaty
    c. automatyczne wylogowanie, podobnie jak poprzednio
3. Klasa ChangeEmailScreen
    a. Wraz z ChangeDataScreen.form dostarcza interfejs GUI ekranu zmiany adresu e-mail
    b. automatyczne wylogowanie, podobnie jak poprzednio
4. Klasy ChangePasswordScreen1, ChangePasswordScreen2
    a. wraz z ChangePasswordScreen1.form, ChangePasswordScreen2.form dostarczają interfejs GUI ekranu zmiany hasła
    b. Sprawdzają poprawność hasła
    c. automatyczne wylogowanie, podobnie jak poprzednio
5. Klasa ChangeUsernameScreen
    a. wraz z ChangeUsernameScreen.form dostarczają interfejs GUI ekranu zmiany nazwy użytkownika
    b. Sprawdzają poprawność nazwy użytkownika
    c. automatyczne wylogowanie, podobnie jak poprzednio

6. Klasa SettingsMainScreen
    a. wraz z SettingsMainScreen.form dostarczają interfejs GUI ekranu z przyciskami odsyłającymi do odpowiednio: zmiany loginu, zmiany hasła, wyświetlenia danych użytkownika, wylogowania, powrotu do poprzedniego ekranu oraz wyjścia
    b. automatyczne wylogowanie, podobnie jak poprzednio
7. Klasa ShowUserDataScreen
    a. wraz z ShowUserDataScreen.form dostarczają interfejs GUI ekranu prezentacji danych użytkownika
    b. automatyczne wylogowanie, podobnie jak poprzednio

Pakiet

src

Zawiera klasy, których nie trzeba było umieszczać w dodatkowych pakietach

1. Klasa Database
    a. Metoda connectToDatabase()
        i. Odpowiada za połączenie z bazą danych SQL
    b. Metoda addUser()
        i. Odpowiada za dodanie rekordu: loginu, hasła, adresu e-mail oraz kodu pin aplikacji do tabeli Users w bazie danych
    c. Metoda addUserData()
        i. Odpowiada za dodanie rekordu: imienia, nazwiska, płci, nr. Telefonu, miejscowości, kodu pocztowego, ulicy, numeru domu oraz numeru pesel do tabeli UsersData w bazie danych, gdzie kluczem jest nazwa użytkownika(dane przypisywane do unikalnej nazwy użytkownika)
    d. Metoda setUserData()
        i. Pozwala na ustawienie nowych danych przedstawionych w metodzie addUserData()
    e. Metoda addOrdinaryAccountNumber()
        i. Odpowiada za dodanie rekordu numeru konta głównego do tabeli OrdinaryAccounts w bazie danych, gdzie kluczem jest nazwa użytkownika(dane przypisywane do unikalnej nazwy użytkownika)
    f. Metoda addSavingsAccountNumber()
        i. Odpowiada za dodanie rekordu numeru konta oszczędnościowego do tabeli OrdinaryAccounts w bazie

danych, gdzie kluczem jest nazwa użytkownika(dane przypisywane do unikalnej nazwy użytkownika)

g. Metoda addCard()
    i. Odpowiada za dodanie rekordu: numeru karty oraz PIN do karty do tabeli Cards w bazie danych, gdzie kluczem jest nazwa użytkownika (dane przypisywane do unikalnej nazwy użytkownika)
h. Metoda addToHistory()
    i. String database to jedna ze zbioru nazw tabeli w bazie danych: {HistoryOrdinary, HistorySavings}
    ii. Kolejne argumenty funkcji to kolejne niezbędne dane wykonywanego przelewu, których z powodu ich liczby pozwolę sobie nie wypisywać ☺
    iii. Odpowiada za dodanie rekordu z danymi przelewu do jednej z powyższych tabel (w zależności od argumentu database)
i. Metoda getHistoryFrom()
    i. String database to jedna ze zbioru nazw tabeli w bazie danych: {HistoryOrdinary, HistorySavings}
    ii. Metoda zwraca tablicę 2D z wszystkimi rekordami-przelewami wychodzącymi użytkownika o podanej jako drugi argument nazwie użytkownika
j. Metoda getHistoryTo()
    i. String database to jedna ze zbioru nazw tabeli w bazie danych: {HistoryOrdinary, HistorySavings}
    ii. Metoda zwraca tablicę 2D z wszystkimi rekordami-przelewami przychodzącymi do użytkownika o podanej jako drugi argument nazwie użytkownika
k. Metoda addCredit()
    i. Odpowiada za dodanie rekordu: kwoty pożyczki, kwoty już zapłaconej, daty rozpoczęcia pożyczki i czasu trwania pożyczki, do tabeli Credits w bazie danych, gdzie kluczem jest nazwa użytkownika (dane przypisywane do unikalnej nazwy użytkownika)
l. Metoda getCredit()
    i. zwraca tablicę z danymi kredytu użytkownika pobranymi z tabeli Credits bazy danych dla nazwy użytkownika podanej na wejściu
m. Metoda getEmail()
    i. zwraca e-mail użytkownika dla podanej na wejściu nazwy użytkownika z tabeli Users bazy danych
n. Metoda getAppCode()
    i. Zwraca kod do aplikacji na podstawie podanej nazwy użytkownika z tabeli Users bazy danych
o. Metoda getOrdinaryAccountNumber()
    i. Zwraca numer konta głównego użytkownika na podstawie podanej nazwy użytkownika z tabeli OrdinaryAccounts bazy danych

p. Metoda getSavingsAccountNumber()
    i. Zwraca numer konta oszczędnościowego użytkownika na podstawie podanej nazwy użytkownika z tabeli SavingsAccounts bazy danych
q. Metoda getOrdinaryAccountBalance()
    i. Zwraca stan konta głównego użytkownika na podstawie podanej nazwy użytkownika z tabeli OrdinaryAccounts bazy danych
r. Metoda getSavingsAccountBalance()
    i. Zwraca stan konta oszczędnościowego użytkownika na podstawie podanej nazwy użytkownika z tabeli SavingsAccounts bazy danych
s. Metoda getUsernameByCard()
    i. Zwraca nazwę użytkownika na podstawie podanego numeru karty użytkownika z tabeli Cards bazy danych
t. Metoda getCard()
    i. Zwraca tablicę z numerem karty oraz kodem pin dla podanej nazwy użytkownika z tabeli Cards bazy danych
u. Metoda getUserData()
    i. Zwraca tablicę z danymi osobowymi użytkownika na podstawie podanej nazwy użytkownika z tabeli UsersData
v. Metoda setUsername()
    i. Pozwala ustawić nową nazwę użytkownika podaną jako drugi argument metody, na podstawie dotychczasowej nazwy użytkowinka (1. Argument) w tabeli Users bazy danych
w. Metoda setPassword()
    i. Pozwala ustawić nowe hasło podane jako drugi argument metody, na podstawie nazwy użytkowinka (1. Argument) w tabeli Users bazy danych
x. Metoda setEmail()
    i. Pozwala ustawić nowy e-mail podany jako drugi argument metody, na podstawie nazwy użytkowinka (1. Argument) w tabeli Users bazy danych
y. Metoda setOrdinaryAccountBalance()
    i. Pozwala ustawić nowy stan konta podany jako drugi argument metody, na podstawie nazwy użytkowinka (1. Argument) w tabeli OrdinaryAccounts bazy danych
z. Metoda setSavingsAccountBalance()
    i. Pozwala ustawić nowy stan konta podany jako drugi argument metody, na podstawie nazwy użytkowinka (1. Argument) w tabeli SavingsAccounts bazy danych
aa. Metoda setCreditAmountPayed()
    i. Pozwala ustawić zapłaconą już kwotę za pożyczkę (podane jako drugi argument metody), na podstawie nazwy użytkowinka (1. Argument) w tabeli Credits bazy danych

bb. Metoda isUsernameTaken()
    i. W tabeli Users bazy danych sprawdza, czy podana na wejściu nazwa użytkownika nie została już wykorzystana
    ii. Jeśli nazwa użytkownika wykorzystana, zwracana jest prawda, w przeciwnym wypadku zwracany fałsz
cc. Metoda verifyUser()
    i. W tabeli Users bazy danych sprawdza, czy użytkownik o podanym loginie i haśle istnieje
    ii. Jeśli użytkownik istnieje, zwracana jest prawda, w przeciwnym wypadku zwracany fałsz
dd. Metoda hasCredit()
    i. W tabeli Credits bazy danych sprawdza, czy użytkownik o podanej nazwie użytkownika posiada kredyt
    ii. Jeśli posiada, zwracana jest prawda, w.p.w. zwracany fałsz
ee. Metoda hasCard()
    i. W tabeli Cards bazy danych sprawdza, czy użytkownik o podanej nazwie użytkownika posiada kartę
    ii. Jeśli posiada, zwracana jest prawda, w.p.w. zwracany fałsz
ff. Metoda verifyCard()
    i. W tabeli Cards bazy danych sprawdza, czy karta o podanym numerze oraz kodzie pin istnieje już w bazie danych
    ii. Jeśli istnieje, zwracana jest prawda, w.p.w. zwracany fałsz
gg. Metoda verifyOrdinaryAccountNumber()
    i. W tabeli OrdinaryAccounts bazy danych sprawdza, czy numer konta głównego o podanym numerze istnieje już w bazie danych
    ii. Jeśli istnieje, zwracana jest prawda, w.p.w. zwracany fałsz
hh. Metoda verifySavingsAccountNumber()
    i. W tabeli SavingsAccounts bazy danych sprawdza, czy numer konta oszczędnościowego o podanym numerze istnieje już w bazie danych
    ii. Jeśli istnieje, zwracana jest prawda, w.p.w. zwracany fałsz
ii. Metoda verifyPhoneNumber()
    i. W tabeli UsersData bazy danych sprawdza, czy numer telefonu o podanym numerze istnieje już w bazie danych
    ii. Jeśli istnieje, zwracana jest prawda, w.p.w. zwracany fałsz
jj. Metoda getUserByPhone()
    i. Szuka podanego, jako argument numeru telefonu w tabeli UsersData bazy danych i, jeśli istnieje, zwraca nazwę użytkownika przypisaną do tego nr. Telefonu
kk. Metoda isAccountNumberOrdinary()

i. Zwraca prawdę, jeśli podany na wejściu numer konta jest kontem głównym/ zwykłym, w.p.w zwraca fałsz
ii. W celu zwrócenia powyższych przeszukuje tabelę OrdinaryAccounts bazy danych

ll. Metoda isAccountNumberSavings()
i. Zwraca prawdę, jeśli podany na wejściu numer konta jest kontem oszczędnościowym, w.p.w zwraca fałsz
ii. W celu zwrócenia powyższych przeszukuje tabelę SavingsAccounts bazy danych

mm. Metoda getUserNameByAccount()
i. Dla podanej jako drugi argument nazwy tablicy bazy danych ze zbioru: {Ordinary, Savings}, zwraca nazwę użytkownika dla podanego jako pierwszy argument numeru konta

nn. Metoda deleteUser()
i. Usuwa z tabeli Users użytkownika o podanej na wejciu nazwy użytkownika

oo. Metoda deleteCredit()
i. Usuwa z tabeli Credits bazy danych pożyczkę przypisaną do podanej na wejściu nazwy użytkownika

2. Klasa DataValidation
   a. Odpowiada za walidację danych
   b. Wszystkie metody w klasie zwracają prawdę w przypadku poprawnego literału podanego na wejściu, w.p.w. zwracają fałsz
3. Klasa JavaMail
   a. Odpowiada za przesył wiadomości e-mail
   b. Przyjmuje w konstruktorze dwa argumenty: recepient to adres e-mail, natomiast code to 6-cyfrowy kod weryfikacyjny
   c. Wykorzystuje klasy: java.io.File, java.util.Properties oraz javax.mail.*
4. Klasa PreScreen
   a. Przechowuje obiekt typu JFrame
   b. Metoda getjFrame() zwraca powyższy obiekt
5. Klasa Screen
   a. Odpowiada za wyświetlanie paneli przygotowanych przez wybrane klasy
   b. Dziedziczy po klasie PreScreen
   c. Jako argumenty konstruktora przyjmuje obiekt klasy User, obiekt Screen prev_screen, oraz Screen next_screen
   d. Tworzy nowy obiekt typu JFrame
   e. Zawiera niezaimplementowaną metodę CreateScreen(), implementowaną w innych klasach
6. Klasa User
   a. Przechowuje informacje o zalogowanym użytkowniku: dane logowania, dane osobowe, numery kont, numer karty, pin do karty, kod do aplikacji

    b. Wykorzystywana w wielu innych klasach (m.in. odpowiadających za przelewy) do pobierania informacji o aktualnie korzystającym z aplikacji użytkowniku
7. Klasa FAQScreen
    a. Wraz z FAQScreen.form tworzą interfejs GUI ekranu zawierającego FAQ- najczęściej zadawane pytania i odpowiedzi na te pytania
    b. Podobnie, jak w innych interfejsach GUI, zawiera panelik odpowiedzialny za automatyczne wylogowanie, z licznikiem czasu
8. Klasa MainScreen
    a. Wraz z MainScreen.form tworzy interfejs ekranu głównego- ekranu startowego po pomyślnym zalogowaniu się użytkownika
    b. Zawiera przyciski:
        i. Credit
            1. Pozwala uruchomić interfejs GUI kredytów gotówkowych
        ii. Profile Settings
            1. Pozwala uruchomić interfejs GUI ustawień konta
        iii. FAQ
            1. Pozwala uruchomić interfejs GUI FAQ
        iv. Log Out
            1. Pozwala ręcznie wylogować się z konta
        v. Profile Settings
            1. Pozwala uruchomić interfejs GUI ustawień konta
        vi. Domestic Transfer
            1. Pozwala uruchomić interfejs GUI przelewu krajowego
        vii. Foreign Transfer
            1. Pozwala uruchomić interfejs GUI przelewu zagranicznego
        viii. Own Transfer
            1. Pozwala uruchomić interfejs GUI przelewu własnego
        ix. Standing Order Transfer
            1. Pozwala uruchomić interfejs GUI zlecenia stałego
        x. Blik Phone Transfer
            1. Pozwala uruchomić interfejs GUI przelewu BLIK na telefon
        xi. Ordinary History
            1. Pozwala uruchomić interfejs GUI historii przelewów konta głównego
        xii. Savings History
            1. Pozwala uruchomić interfejs GUI historii przelewów konta oszczędnościowego
    c. Ponadto wyświetla podgląd numerów kont- zwykłego i oszczędnościowego oraz stan powyższych kont

9. Klasa configure
    a. Mini-aplikacja konsolowa, w której należy podać dane konfiguracyjne aplikacji bankowej:
        i. Adres e-mail
        ii. Hasło adresu e-mail
        iii. Nazwę bazy danych na komputerze użytkownika
        iv. Nazwę użytkownika MySQL (najczęściej „root”)
        v. Hasło MySQL
    b. Informacje w podpunktach i.-ii. są potrzebne do poprawnego działania przesyłu informacji na adres e-mail
    c. Informacje w podpunktach iii.-v. są potrzebne do poprawnego połączenia się z bazą danych

---

Pakiet

timer

Zawiera klasy pomocnicze dla mini-panelu automatycznego wylogowania po okresie bezczynności użytkownika

1. Klasa AppTimer
    a. Korzysta z klasy Timer() biblioteki swing
    b. W konstruktorze na wejściu podajemy obiekt typu JLabel oraz obiekt klasy PreScreen
    c. W konstruktorze tworzymy obiekt klasy Timer i implementujemy ActionListener() w taki sposób, aby obiekt typu JLabel (odpowiada za wyświetlanie aktualnego czasu pozostałego do wylogowania) był na bieżąco aktualizowany, aż dojdzie do 0 sekund. Rozpoczyna od 2 minut i idzie do tyłu. Od 10 sekund zmienia kolor obiektu JLabel na czerwony, aby ostrzec użytkownika o kończącym się czasie bezczynności. Jeżeli osiągniemy czas 0 sekund, timer zostaje zatrzymany funkcją stop() i zmienia aktualny ekran, na którym następuje odliczanie, na ekran logowania.
    d. Metoda start() rozpoczyna odliczanie timera
    e. Metoda stop() zatrzymuje odliczanie timera
    f. Metoda reset() resetuje bieżące wartości w timerze spowrotem do 2 minut
2. Klasa MouseAction
    a. Klasa dziedziczy po klasie MouseMotionListener biblioteki java.awt.event.
    b. Odpowiada za wykrywanie ruchu myszką na ekranie
    c. Jeżeli ruch zostanie wykryty, metoda mouseMoved() wywołuje odpowiednie metody (szczegóły poniżej)

d. Jako argument konstruktora, dostarczamy obiekt klasy AppTimer
e. Implementujemy tylko metodę mouseMoved(), gdyż tylko ona jest nam potrzebna w programie
f. Metoda mouseMoved()
g. Wywołuje na obiekcie klasy AppTimer metody reset() oraz stop()

Pakiet

transfers

Obszerny pakiet zawierający klasy odpowiedzialne za wszystkie rodzaje przelewów (interfejsy GUI), klasy odpowiedzialne za drukowanie potwierdzeń pdf, klasy odpowiedzialne za historie przelewów (interfejsy GUI) oraz klasy pomocnicze.
W pakiecie dwukrotnie wykorzystywany jest **wzorzec projektowy** Factory (Fabryka)

1. Typ wyliczeniowy enym AccountChoosed
    a. Zawiera dwie składowe: ORDINARYACCOUNT, SAVINGSACCOUNT
    b. Pomaga w identyfikacji typu konta, z jakiego przelew jest wykonywany
2. Klasa CurrenciesExchangeRate
    a. Klasa pomocnicza dla przelewu zagranicznego
    b. Najpierw ustawiamy bazową walutę jako „PLN”
    c. W konstruktorze inicjalizujemy mapę currencies, gdzie kluczem jest kraj waluty, a wartością kod waluty (np. „EUR”, „USD”, etc.) oraz mapę currenciesWithAmounts, gdzie kluczem jest waluta, a wartością wartość jednostkowa waluty w stosunku do waluty bazowej. Następnie wywołujemy metodę setCurrencies()
    d. Metoda setCurrencies()
        i. Rozpakowujemy dostarczony plik .zip przechowujący plik .csv z dostępnymi walutami
        ii. Dalej otwieramy plik .csv i wczytujemy do mapy currencies walut odczytane waluty. Ważna jest tutaj pomocnicza klasa CurrencyEntry, która reprezentuje pojedynczy wiersz pliku .csv. W klasie CurrencyEntry wybieramy interesujące nas elementy wiersza z pliku .cvs i wypełniamy nimi mapę currencies
    e. Metoda getCurrencies()
        i. Zwraca mapę currencies

    f. Metoda getCurrencyAmount()
        i. Korzysta z zewnętrznej, zainstalowanej biblioteki „money” oraz „moneta”, przy pomocy metod wbudowanych w tą bibliotekę przelicza wartość odpowiedniej waluty na złotówki (walutę bazową). Co istotne, metoda getExchangeRate pobiera dane za pośrednictwem połączenia internetowego z europejskiego banku centralnego (ECB) i mamy gwarancję, że kurs walut jest aktualny
3. Klasa CurrencyEntry
    a. Pomocnicza klasa dla klasy CurrenciesExchangeRate, opisana powyżej
4. Klasa FileSystem
    a. Implementuje interfejs TreeModel
    b. Odpowiada za system plików użytkownika
    c. Przechodzi po kolejnych plikach użytkownika
    d. Jako, że system plików przypomina strukturą drzewo, to metody zaimplementowane w klasie przypominają przechodzenie drzewa
    e. Jest wykorzystywana do wyświetlania systemu plików użytkownika
    f. W konstruktorze ustawiamy fileRoot (korzeń, od którego rozpoczynamy przeszukiwanie) na system.getProperty(„user.home”) <-zwraca ścieżkę do katalogu domowego użytkownika
    g. Metoda getRoot() zwraca korzeń systemu plików
    h. Metoda getChild() zwraca dziecko rodzica podanego jako argument
    i. Metoda getChildCount() zwraca liczbę naliczonych dzieci
    j. Metoda isLeaf() zwraca prawdę, jeśli węzeł podany na wejściu jest liściem
    k. Metoda getIndexOfChild() zwraca indeks dziecka
    l. Metoda addTreeModelListener() dodaje do listy modeli (modele są dla obiektu JTree) nowy model
    m. Metoda removeTreeModelListener() usuwa -||-
5. Klasa FileElement
    a. Dziedziczy po klasie File
    b. Konstruktor tworzy obiekt typu File reprezentujący plik
    c. Metoda toString() zwraca ścieżkę do pliku
6. Klasa PdfFactory
    a. Realizuje wzorzec projektowy fabryki
    b. W konstruktorze przyjmuje parametry konieczne do utworzenie obiektów klas produkowanych
    c. Metoda getPdfGenerator() produkuje odpowiedni obiekt wybranej klasy na bazie typu wyliczeniowego PdfType
        i. STANDARD zwraca obiekt PdfGeneratorStandard
        ii. BLIK zwraca obiekt PdfGeneratorBLIK

iii. ZLECENIESTALE zwraca obiekt PdfGeneratorStandingOrder
iv. WLASNY zwraca obiekt PdfGeneratorOwn

d. Interfejs PdfGenerator
   i. Deklaruje metodę generatePdf
e. Klasa PdfGeneratorStandard
   i. Implementuje interfejs PdfGenerator
   ii. Deklaruje czcionki, jakie będą używane w pliku pdf
   iii. Deklaruje inne niezbędne elementy do generacji pliku pdf
   iv. W konstruktorze pobiera przekazane informacje o przelewie i wypełnia nimi mapę „data", której klucz zawiera informację o wartości, jaka jest do tego klucza przypisana
f. Metoda generatePDF()
   i. Odpowiada za generowania pliku pdf
   ii. Jako argument podajemy ścieżkę, gdzie chcemy zapisać plik
   iii. Przy wykorzystaniu zewnętrznej biblioteki PdfBox tworzymy dokument w formacie .pdf, wypełniamy przekazanymi danymi (data wygenerowania, data przelewu i inne dane dotyczące przelewu)

7. Klasa PdfGeneratorStandingOrder
   a. Dziedziczy po klasie PdfGeneratorStandard i implementuje interfejs PdfGenerator
   b. Różni się tylko konstruktorem, a dokładniej innymi danymi przelewu
8. Klasa PdfGeneratorBLIK
   a. Dziedziczy po klasie PdfGeneratorStandard i implementuje interfejs PdfGenerator
   b. Różni się tylko konstruktorem, a dokładniej innymi danymi przelewu
9. Klasa PdfGeneratorOwn
   a. Dziedziczy po klasie PdfGeneratorStandard i implementuje interfejs PdfGenerator
   b. Różni się tylko konstruktorem, a dokładniej innymi danymi przelewu

10. Interfejs Transfer
    a. Jest implementowany przez klasy przedstawione poniżej
11. Klasa TransferFactory
    a. Realizuje wzorzec projektowy fabryki (factory)
    b. Zawiera typ wyliczeniowy TransferType zawierający elementy:KRAJOWY, ZAGRANICZNY, ZLECENIESTALE, TELEFONBLIK, WLASNY
    c. Konstruktor TransferFactory
       i. Przyjmuje argumenty niezbędne do przekazania konstruktorom klas produkowanych obiektów

d. Metoda getTransfer()
    i. Produkuje i zwraca obiekt wybranej poprzez typ wyliczeniowy TransferType klasy:
        1. KRAJOWY-> StandardTransfer
        2. ZAGRANICZNY -> ForeginStandardTransfer
        3. ZLECENIESTALE -> StandingOrder
        4. TELEFONBLIK -> BlikPhoneTransfer
        5. WLASNY OwnTransfer

12. Klasa LimitJTextfield
a. Dziedziczy po klasie PlainDocument
b. Odpowiada za ustawienie maksymalnego limitu znaków wpisywanych z klawiatury
c. Jest stosowana w obiektach typu JTextField przy wykorzystaniu metody setDocument()

13. Klasa OnlyNumbers
a. W konstruktorze tworzy obiekt typu KeyAdapter
b. Służy do blokowania wpisania z klawiatury znaków innych, niż cyfry
c. Jest stosowana w obiektach typu JTextField

14. Klasa StandardTransfer
a. Wraz z StandardTransfer.form odpowiada za utworzenie interfejsu GUI przelewu krajowego oraz przy drobnych modyfikacjach szablon wykorzystywany jest też w przelewie zagranicznym i zleceniu stałym
b. Wyświetla użytkownikowi następujące informacje
    i. Dostępne środki na koncie
    ii. Pole tekstowe do wpisania numeru konta
        1. Pole to korzysta z klas OnlyNumbers oraz LimitJTextfield, dzięki czemu jedyne znaki, jakie można wpisać, to cyfry 0-9 oraz ich liczba nie może przekroczyć 26
        2. W przypadku niewypełnienia pola pojawia się stosowny komunikat
        3. W przypadku zbyt krótkiego numeru konta pojawia się komunikat – komunikat pojawia się na bieżąco podczas wpisywania do momentu uzyskania 26 cyfr
        4. W przypadku podania numeru konta, który nie istnieje w bazie danych, pojawia się komunikat
    iii. Pole wyboru między firmą (Company)- wówczas pojawia się jedno pole tekstowe na nazwę firmy; a osobą (Person)- wówczas pojawiają się dwa pola tekstowe na imię oraz na nazwisko
        1. Są to pola obowiązkowe- niewypełnienie kończy się wyskoczeniem komunikatu pod odpowiednim, niewypełnionym polem
    iv. Opcja wpisania adresu- nie są to pola obowiązkowe, natomiast w przypadku zaznaczenia opcji adresu

wypełnienie wszystkich pól jest konieczne, w innym wypadku pod każdym niewypełnionym polem wyskoczy komunikat

1. Po naciśnięciu opcji „Receiver address” pojawiają się pola tekstowe, kolejno na:
    a. Miejscowość
    b. Ulicę
    c. Kod pocztowy
        i. Akceptuje wyłącznie cyfry
        ii. Pierwsza część pola ograniczona do 2 cyfr, druga do 3 cyfr
        iii. Weryfikowana jest poprawność formatu kodu pocztowego XX-XXX
    d. Numer domu/ numer mieszkania
        i. Akceptują wyłącznie cyfry
        ii. Niewpisanie numeru domu kończy się komunikatem, natomiast pole tekstowe na numer lokalu jest opcjonalne
2. Odznaczenie opcji „Receiver address” powoduje, że nie trzeba wypełniać pól adresu

v. Pole tekstowe „Amount”, do którego należy wpisać kwotę przelewu
    1. Pola ograniczone tylko do cyfr 0-9, wpisanie innych znaków niemożliwe
    2. W przypadku niewpisania żadnej wartości wyskoczy stosowny komunikat
    3. Pole po przecinku ograniczone do 2 cyfr
    4. Jeżeli podana kwota przekracza dostępną na koncie, wyświetlany jest komunikat

vi. Pole tekstowe „Title”
    1. Pole wymagane
    2. Należy w nim podać tytuł przelewu
    3. Niewypełnienie kończy się komunikatem

vii. Przycisk „Cancel”
    1. Odpowiada za powrót do ekranu głównego
    2. Powraca wywołując obiekt klasy MainScreen

viii. Przycisk „Next”
    1. Odpowiada za przejście do kolejnego ekranu- wywołując obiekt klasy StandardTransferNextStep
    2. Po naciśnięciu, o ile jakieś dane nie zostały wypełnione (zgodnie z powyższym opisem), komunikaty zostają wyświetlane po odpowiednimi polami i blokowane jest przejście do kolejnego ekranu do momentu wypełnienia prawidłowo pól

c. Metoda setCurrency()
    i. Ustawia domyślną walutę, jako „PLN”
d. Metoda setLabels()

        i. Ustawia wszystkie obiekty JLabel do ustawień początkowych (czyli wszelkie ostrzeżenia ukryte)
    e. Metoda setReceiverNameCombo()
        i. Ustawia obiekt JComboBox – pole wyboru między „Company”, a „Person”, zgodnie ze specyfikacją powyżej
    f. Metoda setTransferAmountTxt()
        i. Odpowiada za obsługę pól tekstowych do wpisywania kwoty przelewu
    g. Metoda setPostcodeTxt()
        i. Odpowiada za obsługę pól tekstowych kodu pocztowego
    h. Metoda setAccountNumberTxt()
        i. Odpowiada za obsługę pola tekstowego na numer konta odbiorcy
    i. Metoda setReceiverAddressRadioButton()
        i. Odpowiada za obsługę pola wyboru adresu odbiorcy oraz pól tekstowych odpowiedzialnych za adres
    j. Metoda setNextButton()
        i. Odpowiada za ustawienie przycisku uruchamiającego kolejny ekran. Jeśli któraś z wymaganych danych została błędnie wpisana, lub nie została w ogóle wpisana, metoda ta nie pozwoli na przejście dalej i wyświetli pod odpowiednimi polami komunikaty o błędach
15. Klasa ForeignStandardTransfer
    a. Dziedziczy po klasie StandardTransfer
    b. Korzysta ze StandardTransfer.form
    c. Korzysta z klasy CurrenciesExchangeRate- dostarcza ona walut oraz przeliczników na złotówki
    d. W konstruktorze klasy zmieniane są niezbędne informacje wyświetlane na ekranie, aby przystosować ekran do przelewu zagranicznego (korzystamy z klasy i form’a klasy odpowiedzialnej za przelew krajowy, więc konieczne są pewne zmiany)
    e. Metoda setTransferData()
        i. ustawia walutę przekazywaną do kolejnych klas na walutę wybraną przez użytkownika
    f. Metoda setAccountNumberCountryComboBox()
        i. odpowiada za wyświetlenie na ekranie przy numerze konta pola wyboru zawierającego zbiór kodów IBAN (kodów krajów), domyślnie ustawione na „PL”
    g. Metoda setCurrencyComboBox()
        i. ustawia pole wyboru waluty, korzysta ze wspomnianej wcześniej klasy CurrenciesExchangeRate
    h. Metoda setTransferAmountTxt()
        i. Metoda z klasy macierzystej StandardTransfer w tym przypadku wymaga zmian. Przede wszystkim jeśli użytkownik wpisze kwotę i wybierze walutę, należy zadbać, aby sprawdzić kwotę przeliczoną na złotówki po

odpowiednim przeliczniku i dopiero ta kwota musi być porównywana z kwotą dostępną na koncie i ewentualnie powinien zostać wyświetlony komunikat o zbyt małej ilości środków na koncie
    ii. Powyższe realizuje opisywana metoda
i. W interfejsie GUI po wyborze waluty wyświetlany jest odpowiedni przelicznik, nazwa waluty oraz kraj waluty, według wzoru: [1 PLN = x kod waluty (kraj waluty)]

16. Klasa StandingOrder
    a. Dziedziczy po klasie StandardTransfer i korzysta z jej interfejsu GUI, wprowadzając odpowiednie poprawki
    b. Zmodyfikowana jest metoda setNextButton() z klasy macierzystej StandardTransfer- w tym przypadku przy pomyślnym wpisaniu danych wywołujemy obiekt klasy StandingOrderNextStep, a nie TransferNextStep, jak w przypadku klas StandardTransfer i ForeignTransfer, gdyż przed ekranem tworzonym przez klasę TransferNextStep konieczny jest jeszcze jeden ekran pośredni
17. Klasa StandingOrderNextStep
    a. Wraz z StandingOrderNextStep.form tworzy interfejs GUI pośredni między pierwszym ekranem zlecenia stałego, a ekranem tworzonym przez klasę TransferNextStep
    b. W tym interfejsie GUI zaimplementowane są następujące elementy:
        i. Pole tekstowe do wpisania liczby jednostek czasu z przedziału [0,999) [zostały wykorzystane klasy OnlyNumbers i LimitJTextfield]
        ii. Pole wyboru jednostki czasu ze zbioru: {dzień, tydzień, miesiąc}
            1. Wyżej wymienione pola są obowiązkowe, niewypełnienie kończy się odpowiednim komunikatem i blokadą przejścia dalej
        iii. Pole typu JCalendar – mini interfejs GUI kalendarza, w którym użytkownik może wybrać datę rozpoczęcia obowiązywania zlecenia stałego
            1. Nałożone zostało ograniczenie, które nie pozwala na wybranie nieprawidłowej (czyli wcześniejszej, niż data wypełniania danych) daty rozpoczęcia obowiązywania zlecenia.
            2. W przypadku wyboru daty wcześniejszej niż data wypełniania danych przez użytkownika, zostanie wyświetlony komunikat o błędnej dacie
        iv. Możliwość wyboru daty zakończenia obowiązywania zlecenia- jest opcjonalne, ale jeśli użytkownik zaznaczy tą opcję, to wybór daty jest konieczny
            1. Jest zaimplementowane zabezpieczenie, które nie dopuszcza do sytuacji, w której data zakończenia będzie wcześniejsza niż data rozpoczęcia zlecenia

+ co najmniej jeden cykl czasu zadany wcześniej przez użytkownika
2. Data zakończenia nie jest jednocześnie rzeczywistą datą kiedy ostatni raz powinien zostać wykonany przelew- wskazuje ona tylko na prawą stronę przedziału czasowego (innymi słowy nie można wyjść poza jej obręb)

v. Przycisk Cancel umożliwia powrót do ekranu klasy StandingOrder
vi. Przycisk Next umożliwia przejście do kolejnego ekranu- klasy TransferNextStep

c. Metoda setLabels()
   i. Ustawia wartości początkowe obiektów JLabel
   ii. Ukrywa ostrzeżenia
d. Metoda setNextButton()
   i. Umożliwia przejście dalej po spełnieniu wyżej wymienionych warunków
   ii. W tej metodzie sprawdzana jest poprawność wszelkich wybranych przez użytkownika danych
e. Metoda setStartPaymentPanel()
   i. Ustawia obsługę kalendarza dla ustawienia daty rozpoczęcia obowiązywania zlecenia
f. Metoda setEndPaymentPanel()
   i. Ustawia obsługę kalendarza dla ustawienia daty zakończenia obowiązywania zlecenia
g. Metoda setEndDateRadioButton()
   i. Ustawia pole wyboru „czy chcesz ustawić datę zakończenia?”
h. Metoda setCancelButton()
   i. Ustawia przycisk powrotu do poprzedniego ekranu
i. Metoda setTimeUnitsTxt()
   i. Ustawia pole tekstowe wyboru liczby jednostek czasu

18. Klasa BlikPhoneTransfer

a. Wraz z BlikPhoneTransfer.form dostarcza interfejs GUI przelewu blik na telefon
b. Interfejs stworzony na podobieństwo StandardTransfer, jedyna różnica w interfejsie, to:
   i. W tym przypadku podajemy numer telefonu, a nie numer konta
      1. Możliwość wpisania tylko cyfr (podobnie jak dla numeru konta), limit znaków to 9 i sprawdzana jest poprawność numeru telefonu- wymogiem jest 9 cyfr, zakładamy, że podawany jest polski numer telefonu komórkowego
      2. Jeżeli użytkownik o podanym numerze konta nie istnieje w bazie danych, wyświetlany jest stosowny komunikat, a przelew jest uniemożliwiany

ii. W tym przypadku pole wyboru czy chcemy wpisać adres odbiorcy, nie istnieje, gdyż w przelewach BLIK na telefon to pole nie ma zastosowania
iii. Podobnie, jak w przypadku poprzednich klas, sprawdzana jest kwota przelewu podana przez użytkownika pod kątem przekroczenia dostępnego budżetu

c. Metoda setLabels()
   i. Ustawia obiekty JLabel na początkowe ustawienia (czyli ostrzeżenia chowamy)
d. Metoda setCurrency
   i. ustawia domyślnie walutę na „PLN”
e. Metoda setReceiverNameCombo
   i. ustawia pole wyboru między osobą a firmą (podobnie jak w poprzednich, opisanych wyżej klasach)
f. Metoda setTransferAmountTxt()
   i. ustawia pole wpisywania ilości gotówki (podobnie, jak wcześniej)
g. Metoda setPhoneNumberTxt()
   i. ustawia pole tekstowe wpisywania numeru telefonu
h. Metoda setNextButton()
   i. sprawdza poprawność wprowadzonych danych oraz wypełnienie wszystkich wymaganych pól oraz dostępność użytkownika o podanym numerze telefonu (korzystając z bazy danych)
   ii. w przypadku pełnej poprawności pozwala przejść dalej do ekranu klasy TransferNextStep

19. Klasa OwnTransfer

a. Wraz z OwnTransfer.form dostarcza interfejsu GUI przelewu własnego między kontami użytkownika
b. W interfejsie:
   i. Obiekt typu jMenuBar, w którym użytkownik ma wybór między kontem głównym, a oszczędnościowym. Po najechaniu myszką na dany typ konta, wyświetlane są podstawowe informacje o tym koncie, czyli numer konta oraz dostępna na danym koncie kwota
   ii. Po wybraniu jednego z kont w panelu wyboru opisanego powyżej, poniżej wyświetlane jest konto „przeciwne” do wybranego- czyli jeśli użytkownik chce przelać pieniądze z konta oszczędnościowego na główne, wyświetlone zostanie poniżej konto główne i vice versa
   iii. Dalej standardowo pole tekstowe do wpisania kwoty, na jaką użytkownik chce wykonać przelew (ze sprawdzeniem czy kwota nie przekracza stanu konta)
   iv. Pole tekstowe na tytuł przelewu
   v. Przyciski Cancel oraz Next odpowiedzialne odpowiednio za wycofanie się do ekranu głównego lub przejście do następnego ekranu klasy TransferNextStep

c. Metoda setjMenu()
    i. odpowiada za ustawienie menu wyboru konta
d. Metoda setFonts()
    i. Odpowiada za pobranie i ustawienie czcionek z pliku Futura.ttc
e. Metoda setLabels()
    i. Odpowiada za ustawienia początkowe obiektów jLabel
f. Metoda setTransferAmountTxt()
    i. Odpowiada za ustawienie pola tekstowego kwoty przelewu
g. Metoda setNextButton()
    i. Odpowiada za przejście do następnego ekranu w przypadku pełnej prawidłowości danych
h. Klasa korzysta z poniższych klas: MenuInfo, SampleMenuListenerAccount1, SampleMenuListenerAccount2

20. Klasa MenuInfo
a. Klasa pomocnicza do ustawienia menu wyboru kont (głównego i oszczędnościowego)
b. Zawiera wyłącznie składowe ustawiane z poziomu innych klas

21. Klasa SampleMenuListenerAccount1
a. Implementuje MenuListener
b. Przeznaczona dla menu konta głównego
c. Odpowiada za pokazywanie i ukrywanie szczegółowych informacji o koncie głównym
d. Metoda menuSelected()
    i. Pokazuje szczegółowe informacje o koncie
e. Metoda menuDeselected()
    i. Ukrywa szczegółowe informacje o koncie
f. Metoda menuCanceled()
    i. Niepotrzebna dla aplikacji, stąd ciało metody puste

22. Klasa SampleMenuListenerAccount2
a. Klasa bliźniacza do tej z punktu 21., ale dla konta oszczędnościowego

23. Klasa TransferNextStep
a. Wraz z TransferNextStep.form dostarcza interfejsu GUI ekranu następującego po wybraniu parametrów każdego z opisanych wyżej typów przelewów
b. Ma za zadanie wyświetlać tylko informacje, jakie podał wcześniej użytkownik (ma charakter informacyjny, oznajmia użytkownikowi co zamierza zrobić)
c. Zawiera przyciski Cancel i Next odpowiednio powracający do ekranu wyboru parametrów przelewu i przechodzący do ostatniego ekranu w klasie TransferConfirm
d. Ma charakter uniwersalny- dostosowana do przyjęcia informacji z klasy StandardTransfer, StandingOrder i StandingOrderNextStep, OwnTransfer, BlikPhoneTransfer, ForeignTransfer

24. Klasa TransferConfirm

a. Wraz z TransferConfirm.form dostarcza interfejsu GUI zatwierdzania wykonania danego typu przelewu
b. Interfejs GUI zawiera
    i. 4 pola tekstowe, każde mieszczące po jednym znaku, wyłącznie cyfry 0-9, jako całość odpowiadają za kod pin do aplikacji
    ii. W przypadku podania błędnego kodu aplikacji wyświetlane ostrzeżenie i blokada możliwości wykonania przelewu
    iii. Opcjonalny przycisk „Print transfer confirmation (pdf)”
        1. Po jego naciśnięciu ukazuje się drzewko systemu plików użytkownika, w którym może wybrać folder, w którym zapisane ma zostać potwierdzenie wykonania przelewu
    iv. Przyciski Cancel i Confirm, odpowiednio wycofuje się do poprzedniego ekranu, albo wykonuje przelew i powraca do ekranu głównego
c. W konstruktorze następuje przechwycenie odpowiednich danych i wywołanie pomocniczych metod
d. Metoda setPrintTransferConfirmationRadioButton() ustawia przycisk wyboru chęci wykonania potwierdzenia pdf przelewu
e. Metoda setCancelButton() ustawia przycisk powrotu do ekranu w klasie TransferNextStep
f. Metoda setConfirmTransferButton()
    i. Weryfikuje poprawność wprowadzonych danych
    ii. Identyfikuje rodzaj przelewu, jaki chcemy wykonać
    iii. Wprowadza wszelkie dostarczone z poprzednich klas informacje do bazy danych
    iv. W przypadku wyboru potwierdzenia przelewu pdf wywołuje fabrykę PdfFactory dla zidentyfikowanego typu przelewu, podając w argumentach dane przelewu
g. Metoda setBalanceSender() aktualizuje stan konta nadawcy przelewu w bazie danych
h. Metoda setBalanceReceiver() aktualizuje stan konta odbiorcy przelewu w bazie danych
i. Drzewko plików wyświetlane na ekranie korzysta z klasy FileSystem

25. Klasa OrdinaryHistory
a. Wraz z OrdinaryHistory.form dostarcza interfejsu GUI historii przelewów dla konta głównego
b. Interfejs zawiera dwie tabele: pierwsza zawiera wszystkie informacje o wszystkich przelewach wychodzących, druga (poniżej) zawiera wszystkie informacje o wszystkich przelewach przychodzących
c. Tabele wypełniane są w konstruktorze klasy przy pomocy metody klasy Database: getHistoryFrom() oraz getHistoryTo()
d. Przycisk Cancel pozwala na powrót do ekranu głównego

26. Klasa SavingsHistory działa na tej samej zasadzie, jak powyższa klasa OrdinaryHistory, z tym, że wyświetla tabele dla konta oszczędnościowego.