

Programowanie w języku Java	
Temat projektu:	<b>Symulator bankomatu</b>
Zespół:	<b>Jakub Jach, Patryk Jaworski, Bartłomiej Kudła</b>
Rok studiów:	<b>2020/2021</b>
Grupa dziekańska:	<b>2ID12A</b>

## Opis projektu:

Celem projektu było stworzenie aplikacji symulującej działanie bankomatu. Podstawowym założeniem było realizowanie opcji wypłaty gotówki, zmiany kodu PIN oraz sprawdzenia stanu konta.

Projekt stworzono jako projekt Maven w środowisku IntelliJ IDEA.

Wykorzystane biblioteki:

- swing
- awt
- event.ActionEvent
- event.ActionListener
- event.WindowAdapter
- event.WindowEvent
- util.Vector
- util.Arrays
- util.Date
- util.Random
- util.Objects
- sound.sampled.AudioInputStream
- sound.sampled.AudioSystem
- sound.sampled.Clip
- io.File
- io.FileWriter
- io.IOException
- text.SimpleDateFormat

Funkcjonalność projektu (aplikacja bankomatu):

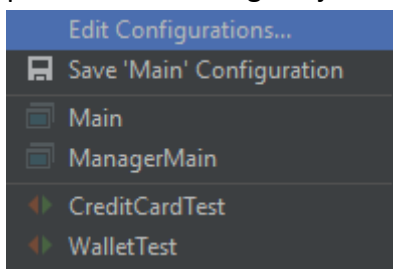
- Możliwość wprowadzenia karty do bankomatu
- Weryfikacja wprowadzonego kodu PIN (blokada karty po 3 nieudanych próbach)
- Wypłata gotówki (predefiniowane kwoty lub wprowadzone z klawiatury)
- Wpłata gotówki (każdy użytkownik posiada swój portfel wraz z odpowiednią ilością banknotów różnych nominałów)
- Sprawdzenie stanu konta
- Zapisywanie informacji o pomyślnej transakcji do pliku (drukowanie potwierdzenia)
- Zmiana kodu PIN do karty
- Przełączanie użytkowników i kart (tylko przed wprowadzeniem karty)
- Zapisywanie użytkowników, kart, pozycji okna i innych informacji do pliku *settings.xml*

Funkcjonalność aplikacji menadżera:

- Załadowanie pliku *settings.xml* z wybranego katalogu
- Zapisanie konfiguracji do pliku *settings.xml* w wybranym katalogu
- Stworzenie konfiguracji od zera (bez ładowania pliku)
- Edycja ustawień aplikacji – pozycji okna, waluty, użytkownika startowego z możliwością podglądu okna
- Edycja, dodawanie i usuwanie użytkowników
- Edycja, dodawanie i usuwanie kart dla użytkowników
- Edycja portfeli dla użytkowników
- Zabezpieczenie przed zapisaniem użytkowników bez kart

## Uruchomienie oraz obsługa projektu

Przed uruchomieniem projektu należy pobrać zawartość zdalnego repozytorium dostępnego [tutaj](#). Po pobraniu projektu należy go otworzyć za pomocą programu IntelliJ IDEA. Projekt posiada kilka konfiguracji startowych:



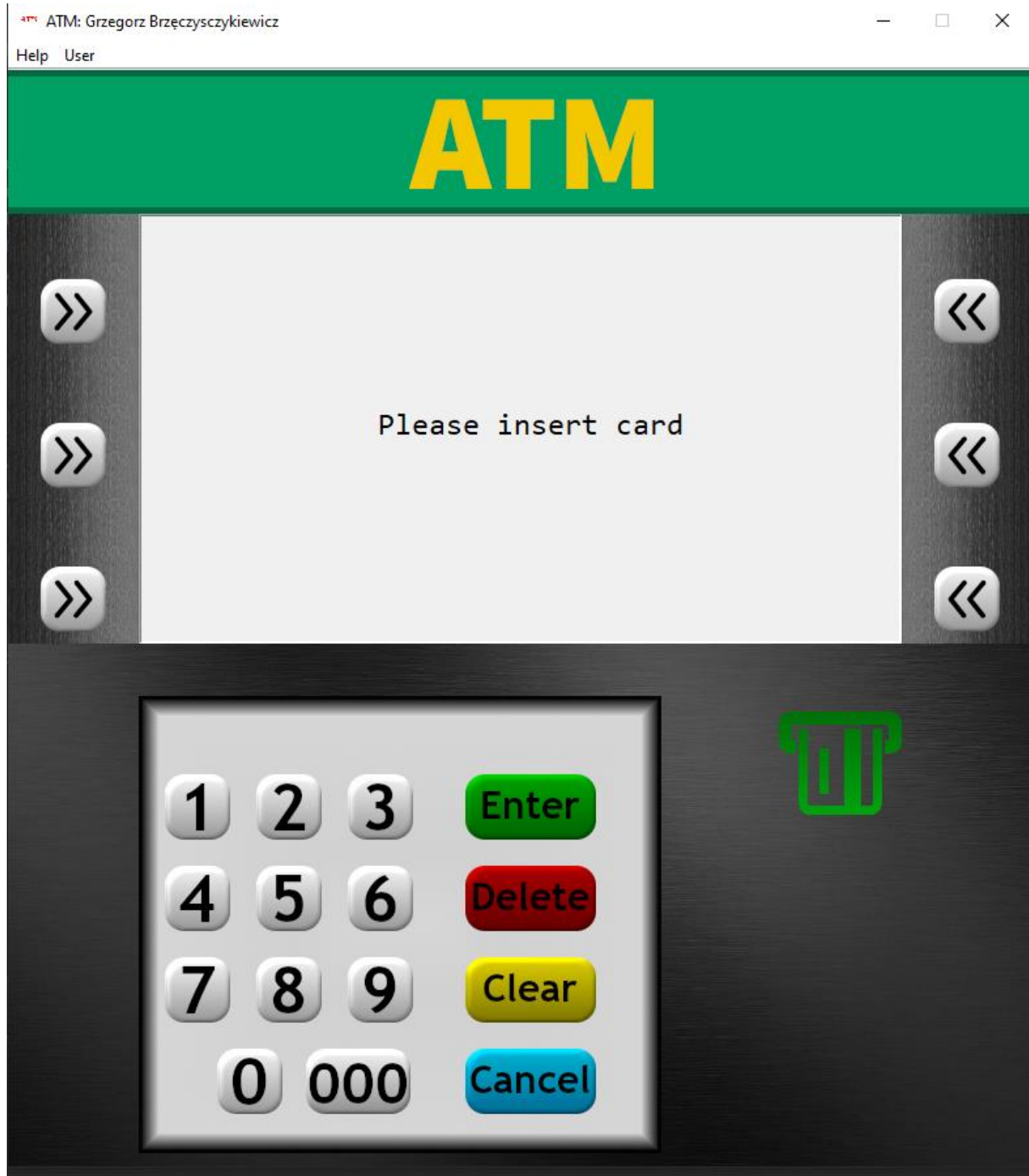
Domyślnie wybraną jest konfiguracja dla aplikacji bankomatu. Z kolei konfiguracja *ManagerMain* jest konfiguracją startową dla aplikacji menadżera. Konfiguracje z dopiskiem *Test* są konfiguracjami startowymi dla testów.

Po wybraniu interesującej nas konfiguracji możemy uruchomić aplikację poprzez *Run* lub wciśnięcie klawiszy Shift+F10.

Możliwe jest także uruchomienie aplikacji bezpośrednio z pliku \*.jar. Pliki te znajdują się w folderze %Project%/out/artifacts/. Folder ten zawiera folder *ATM\_Manager* (dla aplikacji menadżera) oraz *ATM\_Simulator\_jar* (dla aplikacji bankomatu). Aby uruchomić aplikację należy uruchomić skrypt *run.me.bat* w wybranym katalogu. Skrypt jest potrzebny w celu uruchomienia JVM z kodowaniem UTF-8 ponieważ takie domyślnie zostało założone dla całego projektu, a które zmienia się na kodowanie systemowe w przypadku uruchomienia aplikacji poprzez JRE.

## Obsługa bankomatu



Po uruchomieniu aplikacji powinno pokazać się następujące okno (tytuł okna zależny jest od wybranego użytkownika):



Częścią symulacyjną jest wszystko to, co znajduje się poniżej menu aplikacji. Menu umożliwia przełączenie użytkownika lub karty jednak tylko na etapie przed włożeniem karty do bankomatu. Później ta opcja zostaje zablokowana.

User selection	Card selection
Select user: <div>Ralph Kaminsky 2</div>	Select card: <div>Card number 1</div>
<div>OK</div> <div>Cancel</div>	<div>OK</div> <div>Cancel</div>

Okna wyboru użytkownika i karty.

Warning	Warning
<div> You cannot change the user now</div> <div>OK</div>	<div> You cannot change the card now</div> <div>OK</div>

Odmowa zmiany karty i użytkownika.

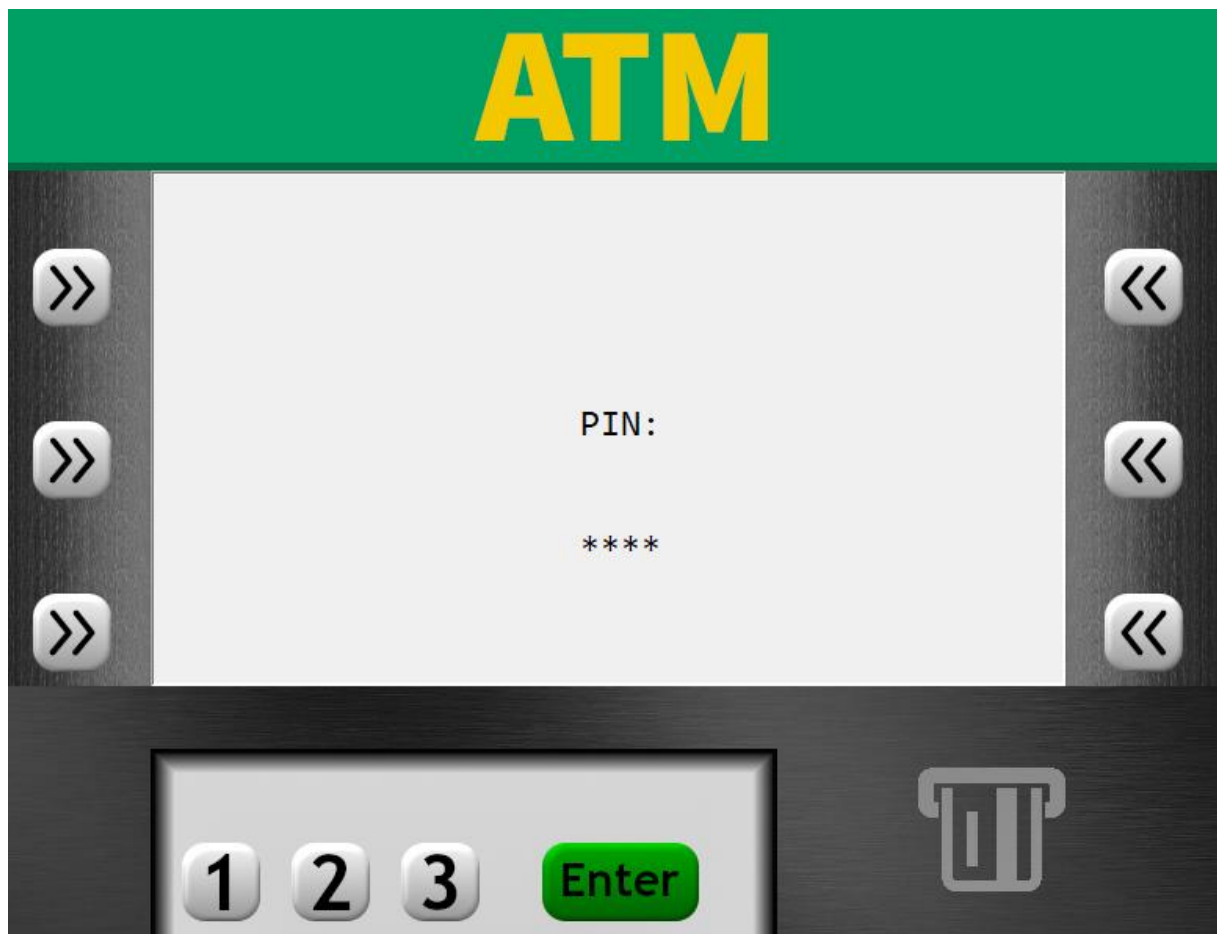
Postępując zgodnie z instrukcją na ekranie przystępujemy do włożenia karty. Bankomat sprawdzi kartę i jeśli karta nie jest zablokowana to umożliwi nam przejście do wprowadzania PINu.



W przeciwnym wypadku na ekranie pojawi się stosowny komunikat o błędzie:



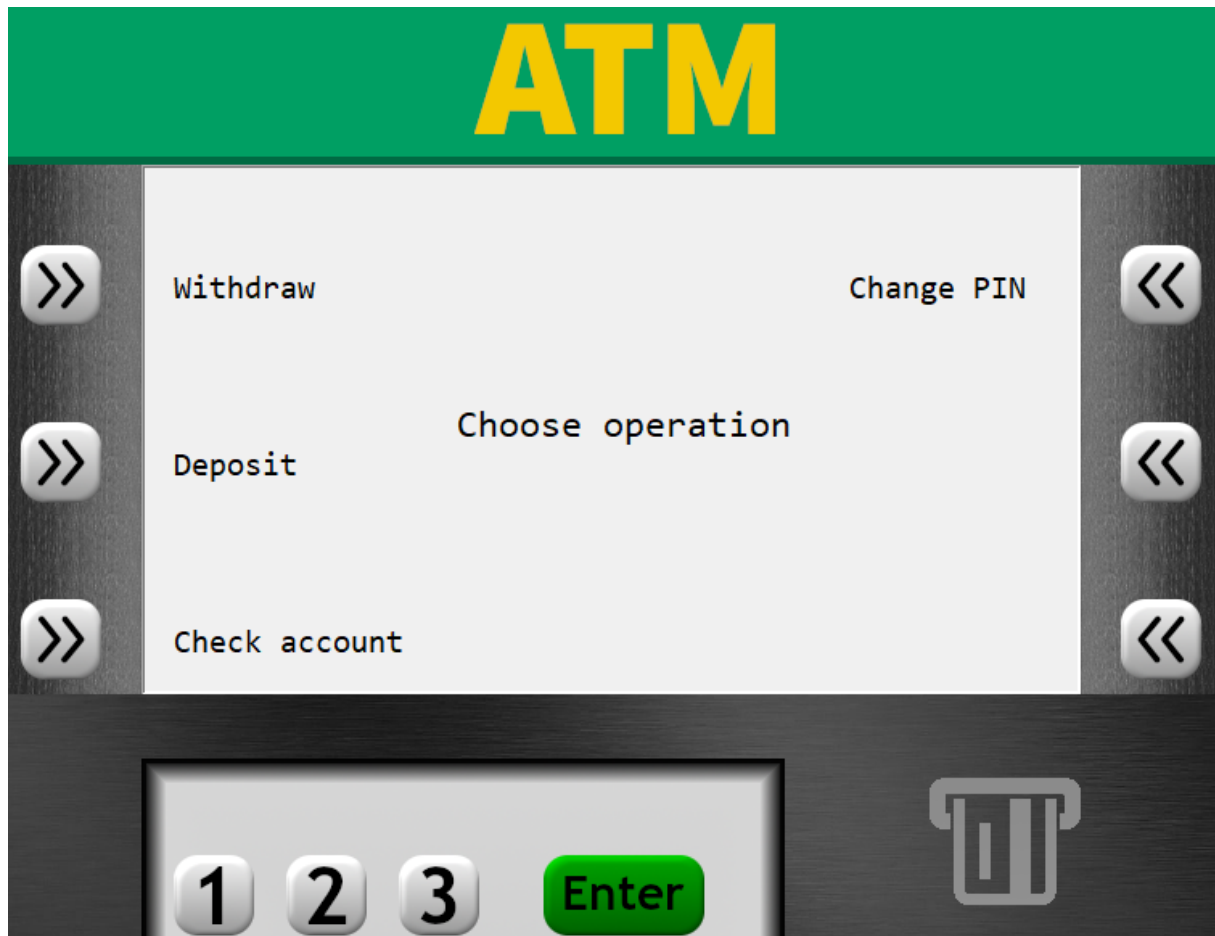
Wprowadzamy kod PIN korzystając z klawiatury poniżej ekranu (obsługiwanej myszką).



Następnie potwierdzamy wprowadzony kod naciskając przycisk Enter na części z klawiaturą.

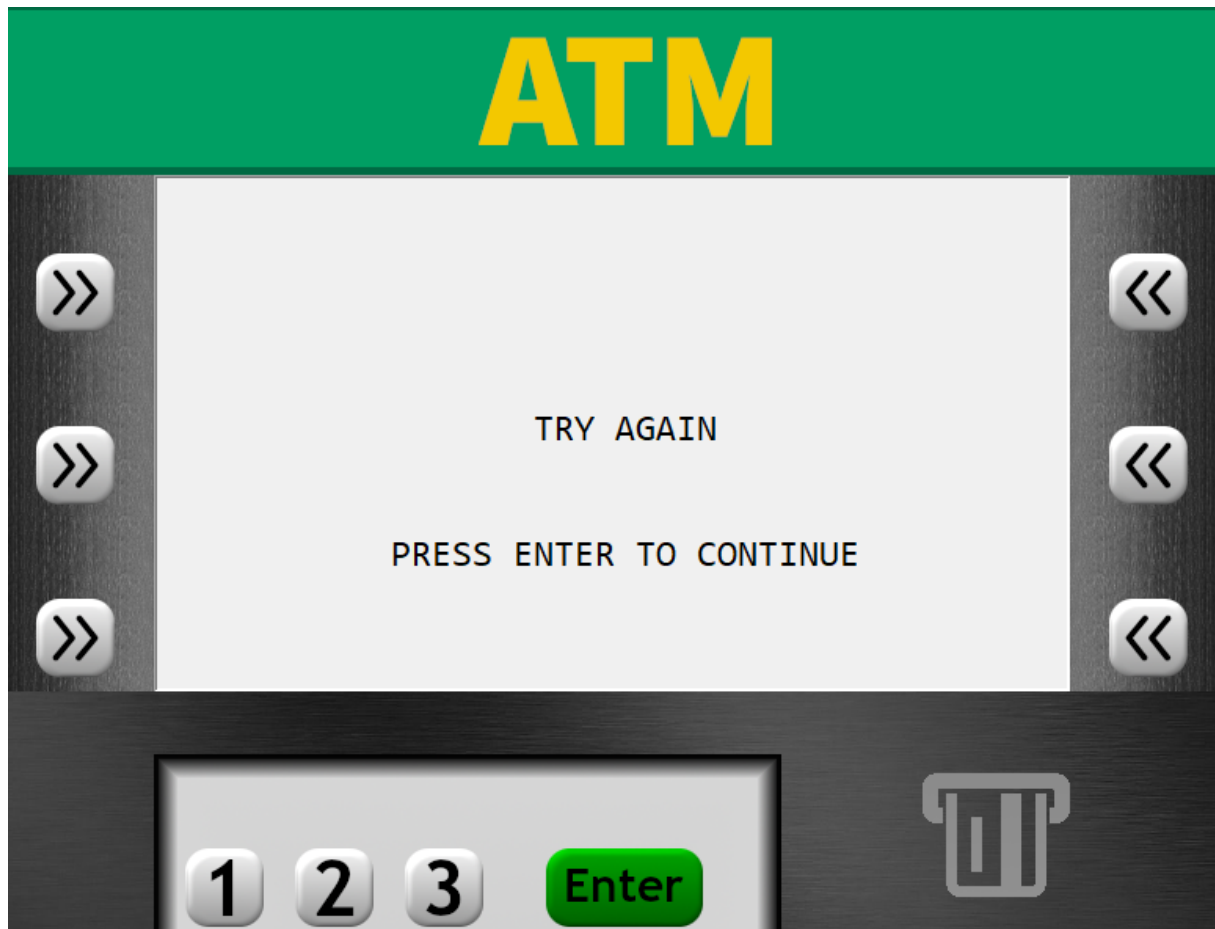
Jeżeli kod jest poprawny to zostaniemy przeniesieni do wyboru operacji:



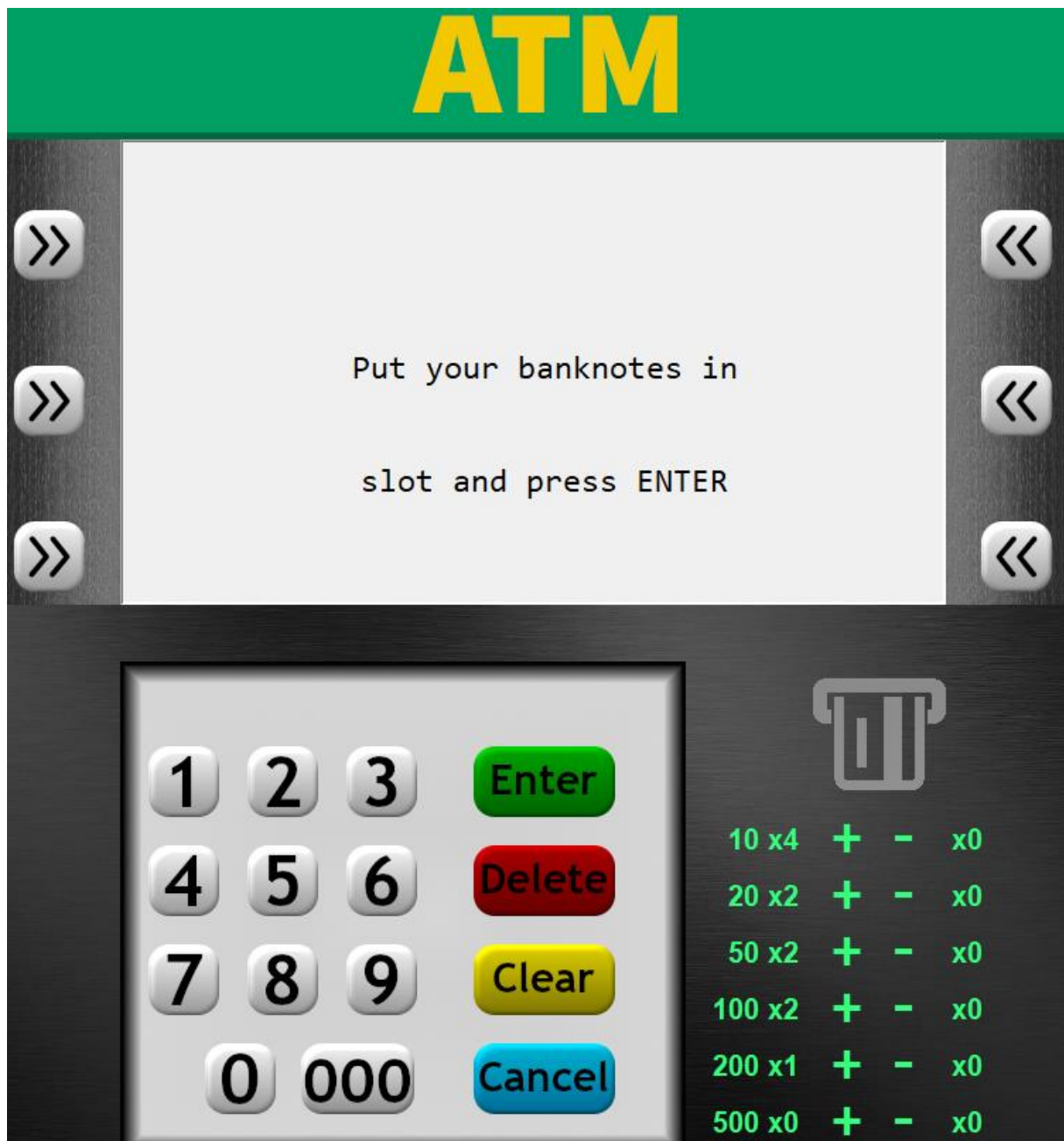


Korzystając z bocznych klawiszy koło ekranu wybieramy interesującą nas operację.

Z kolei jeżeli kod PIN okaże się niepoprawny to zostaniemy o tym poinformowani (po 3 nieudanych próbach bankomat zablokuje kartę).



W przypadku wybrania opcji depozytu gotówki poniżej przycisku do włożenia karty pojawi się zestaw przycisków do operacji z portfelem użytkownika:



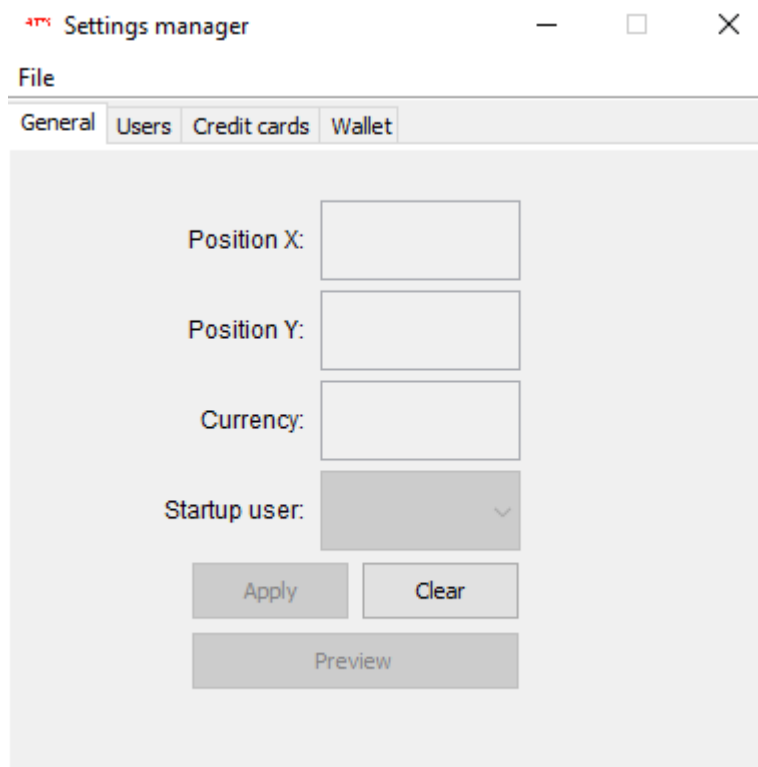
Naciskając przyciski **+** i **-** wybieramy ilość banknotów którą chcemy zdeponować. Po lewej stronie przycisków pokazane jest ile banknotów jakich nominałów znajduje się w portfelu użytkownika. Z kolei po lewej stronie pokazana jest ilość banknotów wybranego nominału do depozytu.

Po zakończeniu operacji pytani jesteśmy o wydrukowanie potwierdzenia. W przypadku odpowiedzi *Yes* stworzony zostaje raport (plik \*.html w katalogu *userdata*) z danymi o transakcji.

## Obsługa menadżera

Menadżer dostępny jest po wybraniu konfiguracji *ManagerMain* w konfiguracjach startowych lub poprzez uruchomienie aplikacji (pliku \*.jar) z katalogu *ATM\_Manager*.

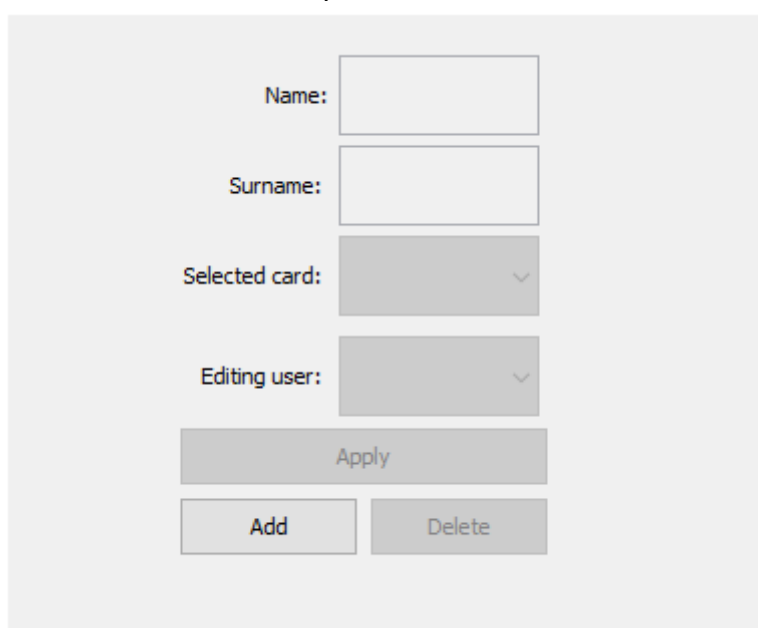
Po uruchomieniu widoczne jest następujące okno:



The screenshot shows a window titled "ATM Manager Settings manager" with standard Windows window controls (minimize, maximize, close). Below the title bar is a menu bar with "File". Underneath is a tabbed interface with four tabs: "General", "Users", "Credit cards", and "Wallet". The "General" tab is selected. The main area contains four input fields: "Position X:", "Position Y:", "Currency:", and "Startup user:". The "Startup user:" field is a dropdown menu. Below these fields are three buttons: "Apply", "Clear", and "Preview".

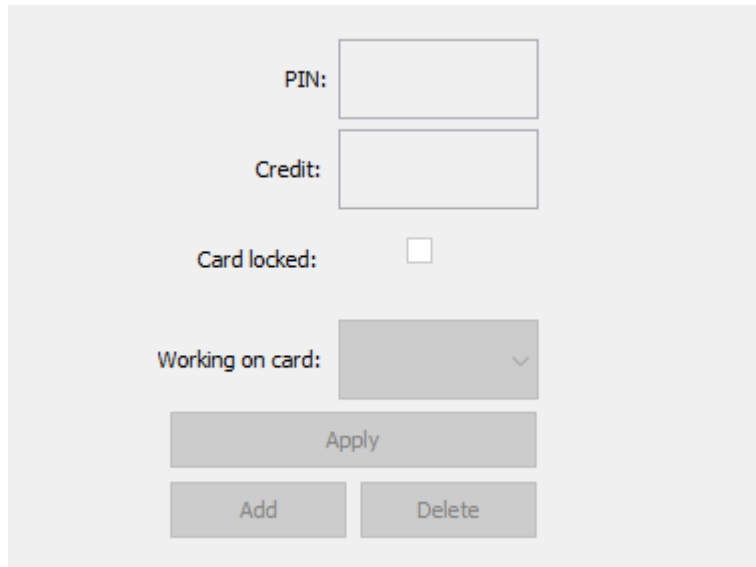
Zakładka *General* dotyczy ustawień całej aplikacji bankomatu.

Zakładka *Users* dotyczy użytkowników. Umożliwia ona edycję, dodawanie i usuwanie użytkowników.



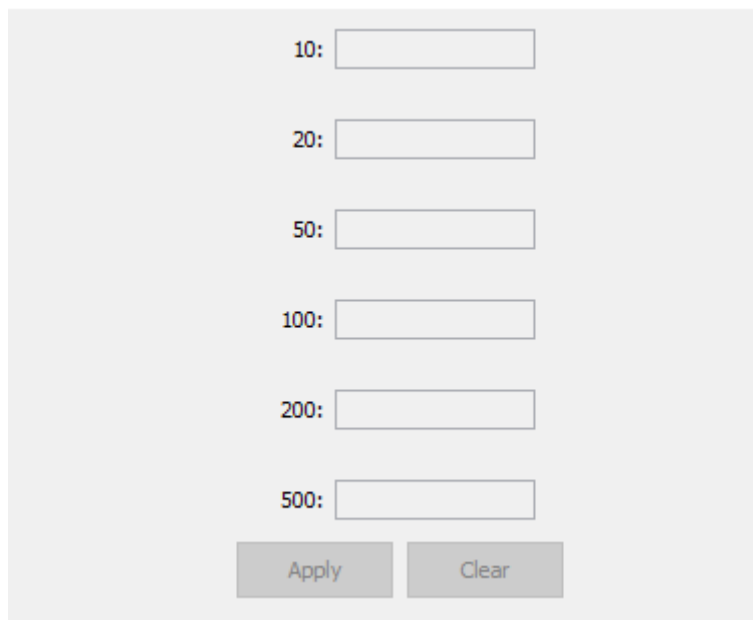
The screenshot shows the "Users" tab selected in the "ATM Manager Settings manager" window. The main area contains four input fields: "Name:", "Surname:", "Selected card:", and "Editing user:". The "Selected card:" and "Editing user:" fields are dropdown menus. Below these fields are three buttons: "Apply", "Add", and "Delete".

Zakładka *Credit cards* jest odpowiedzialna za zarządzanie kartami użytkownika (wybranego z listy *Editing user* w zakładce *Users*). Umożliwia ona dodawanie, usuwanie i edycję danych karty oraz stanu konta.

The screenshot shows a form for editing credit card details. It includes input fields for 'PIN:' and 'Credit:', a checkbox for 'Card locked:', and a dropdown menu for 'Working on card:'. Below these fields are three buttons: 'Apply', 'Add', and 'Delete'.

Zakładka edycji karty (nieaktywna)

Z kolei zakładka *Wallet* jest odpowiedzialna za edycję portfela użytkownika (wybranego z listy *Editing user* w zakładce *Users*).

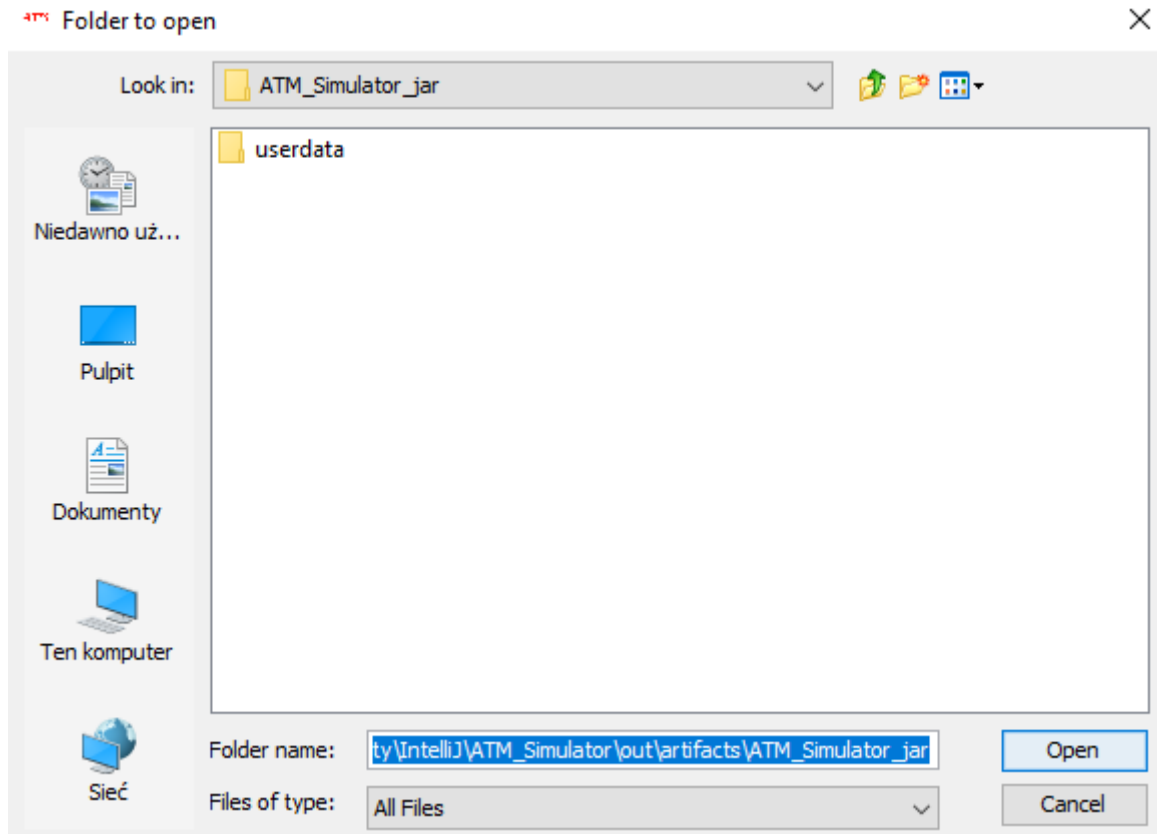
The screenshot shows a form for editing wallet details. It features six input fields labeled '10:', '20:', '50:', '100:', '200:', and '500:'. At the bottom of the form are two buttons: 'Apply' and 'Clear'.

Zakładka edycji portfela (nieaktywna)

Menadżer umożliwia zarówno ładowanie wcześniej utworzonych plików z zapisami stanu bankomatu jak również stworzenie konfiguracji od zera.

W celu utworzenia konfiguracji należy skorzystać z przycisków znajdujących się w poszczególnych zakładkach aplikacji. Z kolei w celu załadowania już istniejącej konfiguracji wybieramy *File*→*Load* a następnie wybieramy folder, w którym znajduje się plik *settings.xml*.

Otworzy się wówczas okno wyboru folderu, gdzie wybieramy folder a następnie opcję *Open*.



W przypadku pomyślnego załadowania pliku tytuł okna zmieni się, a pola we wszystkich zakładkach zostaną wypełnione danymi oraz staną się aktywne.

Settings manager: Open Success

File

General Users Credit cards Wallet

Position X: -1348

Position Y: 34

Currency: PLN

Startup user: 2

Apply Clear

Preview

Pomyślne załadowanie danych z pliku.

W przypadku niepowodzenia tytuł okna również się zmieni, oraz stworzona zostanie (domyślna) konfiguracja startowa jednak tytuł okna będzie inny niż w przypadku powodzenia.

Settings manager: Open Failed

File

General Users Credit cards Wallet

Position X: 0

Position Y: 0

Currency: -

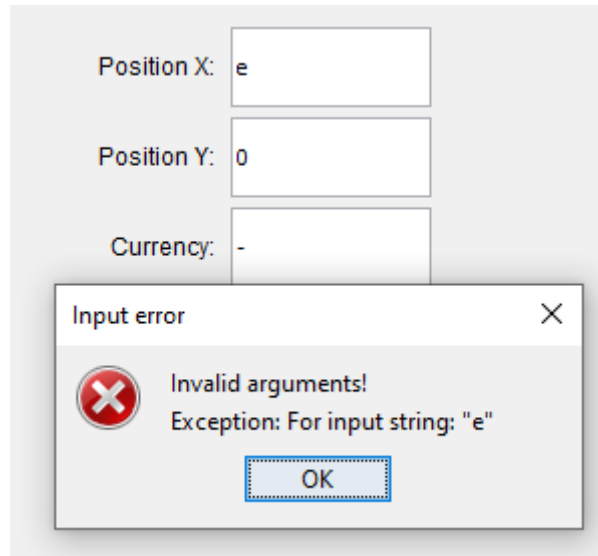
Startup user: 0

Apply Clear

Preview

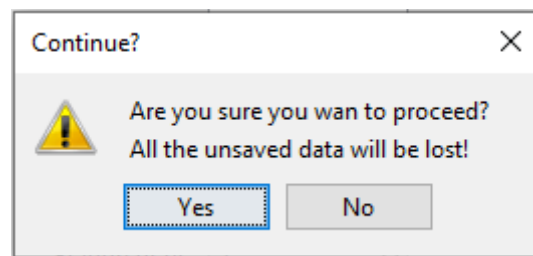
Niepowodzenie ładowania danych z pliku.

W każdej z zakładek wybranie opcji *Apply* zatwierdza wprowadzone dane (wcześniej są one weryfikowane pod kątem poprawności). Dodatkowo istnieje możliwość podglądu okna bankomatu bez potrzeby uruchamiania samej aplikacji bankomatu (opcja *Preview*), która tworzy nowe okno na pozycji ustawionej w konfiguracji oraz z tytułem będącym walutą używaną w bankomacie. Poniżej przedstawiony jest komunikat błędu w przypadku podania niepoprawnych danych.



Przykład niepoprawnych danych wejściowych.

Przy zamykaniu okna jeśli lista użytkowników nie jest pusta zostaniemy poproszeni o potwierdzenie zapisania danych. Jeśli dane były załadowane z pliku, wówczas zapisane zostaną z powrotem do tego samego pliku. Jednakże jeśli dane były tworzone od zera lub też ich ładowanie nie powiodło się to otworzone zostanie okno wyboru folderu do zapisu. W przypadku ładowania danych ponownie (tj. wybrania opcji *Load* przy już wpisanych danych) zostanie wyświetlone ostrzeżenie o możliwej utracie danych.



Ostrzeżenie przy ponownym ładowaniu danych.



## Opis klas i najważniejszych metod

Poniżej znajduje się opis klas oraz najważniejszych metod z tych klas. Dokładny opis metod znajduje się w dokumentacji stworzonej przy użyciu narzędzia Javadoc.

### Pakiet app:

`public class Main` – Główna klasa aplikacji bankomat. Zawiera metodę `main`, która jest główną metodą w programie.

`public class Window` – Klasa okna aplikacji. Jej najważniejszą metodą (oprócz konstruktora) jest metoda `actionPerformed`, która odpowiada za reagowanie na akcje użytkownika. Tworzy cały interfejs użytkownika.

`public class StateManager` – Klasa odpowiedzialna za reagowanie na akcje użytkownika – zmianę wewnętrznego stanu bankomatu. Jej najważniejszą metodą jest `sendSignal`, która reaguje odpowiednio na otrzymywane kody sygnałów. Metoda zwraca 3 różne wartości w zależności od powodzenia operacji i kodu sygnału.

Metoda może zwrócić :

- 0 – oznacza powodzenie operacji
- -1 – oznacza niepowodzenie operacji
- -2 – oznacza otrzymanie nierozpoznawalnego w danym stanie sygnału

### Pakiet settings:

`public abstract class Settings` – Klasa ta przechowuje dane dla programu – użytkowników, ich karty, pozycję okna wybranego użytkownika itd. Zawiera tylko i wyłącznie pola i metody statyczne. Najważniejszymi metodami są `saveState` oraz `loadSettings`. Pierwsza z nich zapisuje stan bankomatu do pliku. Z kolei druga odpowiada za ładowanie danych z pliku.

### Pakiet sound:

`public class Sound` – El Patricco daje opisikko

## Pakiet user:

`class Account` – klasa zawierająca stan konta użytkownika. Oprócz konstruktora posiada 3 metody:

`public double checkCredit()` – Metoda służy do sprawdzenia stanu konta

`public boolean changeCredit(double gap)` – Metoda służy do zmiany stanu konta. Zwraca `true` w przypadku powodzenia lub `false` w przypadku niepowodzenia czyli wtedy gdy saldo po operacji byłoby mniejsze od 0.

`protected void adminSetCredit(double credit)` – Metoda służy do ustawienia stanu konta przez klasy dziedziczące po tej klasie.

`public class CreditCard extends Account` – klasa zawierająca informacje o karcie oraz stan konta użytkownika. Posiada 3 konstruktory. Posiada metody umożliwiające zablokowanie i odblokowanie karty, sprawdzenie stanu konta oraz tego czy karta jest zablokowana. Posiada także chronione metody umożliwiające bezpośrednią edycję pól obiektu.

`public class User` – Klasa odpowiedzialna za użytkownika. Zawiera w sobie wektor z kartami przypisanymi do konkretnego użytkownika, portfel oraz imię i nazwisko danej osoby. Udostępnia metody umożliwiające wpłatę i wypłatę gotówki oraz przełączanie się między kartami.

`public class Wallet` – Klasa odpowiedzialna za portfel użytkownika. Udostępnia metody, które umożliwiają wprowadzanie odpowiedniej liczby banknotów lub ich wyciągnięcie z portfela, a także sprawdzenie jaka jest suma zawartości portfela.

## Pakiet xml

`public abstract class XMLTools` – Klasa odpowiedzialna za przetwarzanie danych do postaci XML oraz ich odczytywanie z tej postaci. Publiczne klasy z pakietu **user** korzystają z metod tej klasy przy importowaniu i eksportowaniu danych do pliku *settings.xml*. Udostępniają przeciążone metody:

- `public static String getData` – Metoda ta umożliwia pobranie danych ograniczonych przez znacznik przekazany w argumentach.
- `public static String toXML` – Metoda ta umożliwia obudowanie danych przekazanych jej w argumencie wywołania o znaczniki XML przekazane w argumencie.

Oprócz ww. metod klasa posiada także metody liczące liczbę wystąpień danego znacznika oraz podające indeks pierwszego napotkanego znacznika.

## Pakiet manager

`public class ManagerMain` – Główna klasa dla aplikacji menadżera. Tworzy nowe okno klasy `ManagerWindow`.

`public class ManagerWindow extends JFrame implements ActionListener, ChangeListener` – Główne okno aplikacji menadżera. Zawiera w sobie obiekty pozostałych klas *Manager* z pakietu **manager**. Jej najważniejszą metodą poza konstruktorem jest metoda `actionPerformed`.

`public class AppManager extends JPanel implements ActionListener, Manager` – Zakładka aplikacji odpowiedzialna za zarządzanie ustawieniami bankomatu. Jest to domyślna zakładka, na której uruchamia się program. Jej najważniejszą metodą jest metoda `actionPerformed` oraz `updateFields`. Druga z nich jest metodą zaimplementowaną z interfejsu `Manager`. Służy ona aktualizacji pól po zmianie karty lub operacji na danych.

`public class UserManager extends JPanel implements ActionListener, Manager` – Zakładka aplikacji odpowiedzialna za zarządzanie użytkownikami. Jest to zakładka o nazwie *Users* w finalnej aplikacji. Posiada dokładnie te same metody (poza konstruktorem) co klasa `AppManager`.

`public class CardManager extends JPanel implements ActionListener, Manager` – Zakładka aplikacji odpowiedzialna za zarządzanie kartami. Jest to zakładka *Credit cards* w finalnej aplikacji. Posiada dokładnie te same metody (poza konstruktorem) co klasa `AppManager`. Wewnątrz niej znajduje się też inna klasa umożliwiająca edycję kart płatniczych.

`public class WalletManager extends JPanel implements ActionListener, Manager` – Zakładka aplikacji odpowiedzialna za zarządzanie portfelami. Jest to zakładka *Wallet* w finalnej aplikacji. Posiada dokładnie te same metody (poza konstruktorem) co klasa `AppManager`. Wewnątrz niej znajduje się też inna klasa umożliwiająca edycję portfela.

`public interface Manager` – Interfejs posiadający jedną niezaimplementowaną metodę - `updateFields`.

## Wkład członków zespołu w projekt

Jakub Jach:

- Stworzenie klas:
  - o `StateManager`
  - o Klas z pakietu **manager**
  - o `Settings`
  - o `XMLTools`
  - o Klas z pakietu **user**
- Rozwój klas `Window`, `Main` – zaimplementowanie połączenia okna z obiektem klasy `StateManager` oraz klasami pakietu **user**.
- Napisanie dokumentacji klas i metod dla ww. klas i pakietów.
- Napisanie testów JUnit5.
- Testowanie aplikacji (menadżer i bankomat).

Patryk Jaworski:

- Stworzenie szkieletu aplikacji bankomatu – układ i rozmiar elementów oraz okna.
- Stworzenie klas
  - o `Window`
  - o `Main`
  - o `Sound`
- Dodanie dźwięków do poszczególnych akcji tj. naciśnięcia przycisku wypłaty gotówki itp.
- Napisanie dokumentacji klas i metod dla stworzonych klas.
- Testowanie aplikacji bankomatu.

Bartłomiej Kudła:

- Przygotowanie dźwięków bankomatu.
- Stworzenie tekstur dla przycisków numerycznych, funkcyjnych i bocznych oraz do wkładania karty.
- Testowanie aplikacji bankomatu.