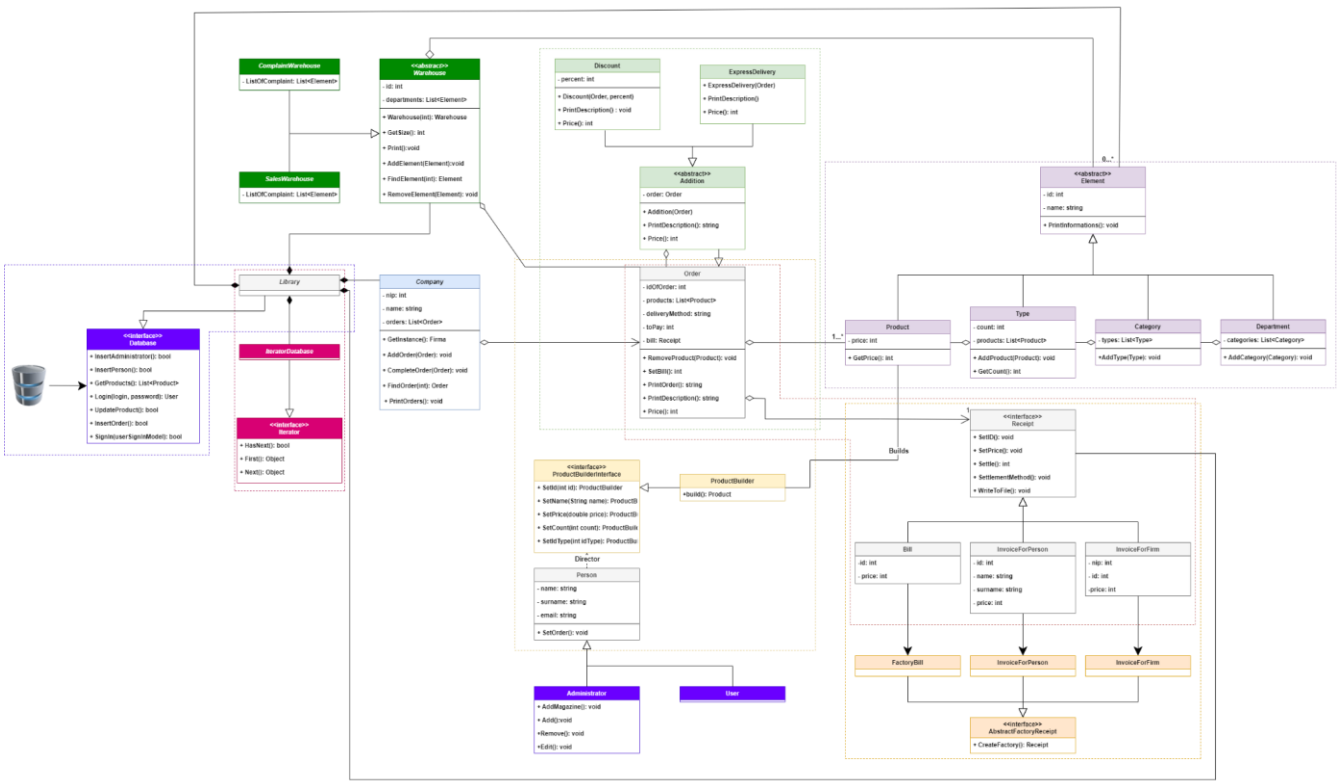
Wydział Informatyki Politechnika Białostocka Zaawansowane techniki programistyczne	Data 23.01.2023
Dokumentacja projektu Temat: Przepływ dokumentów w firmie zajmującej się sprzedażą. 1.Michał Wołosewicz 2.Patryk Wójtowicz 3.Magda Zaborowska	Prowadzący: mgr inż. Daniel Reska

1. Diagram klas budujących użyte wzorce



2. Krótkie opisy każdego wzorca

- **Singleton**
  - **Cel użycia**

    Zastosowano wzorzec Singleton dla klasy Firm, aby istniała tylko jedna jej instancja. Dodatkowo dzięki temu wzorcowi dostęp do tej instancji jest globalny.

  - **Przyporządkowanie klas do wszystkich ról wzorca**

    Klasa Singleton-Firm

  - **Lokalizację wzorca w kodzie**

    src/Firm

  - **Określenie wektora zmian tego wzorca w programie**
  - **Pozwala**

- **Strategia**
  - **Cel użycia**

    Wzorzec strategia wykorzystano do realizacji zamówień. Ułatwił on wybór sposobu rozliczenia zamówienia, dzięki czemu są one wymienialne. Kontekstem jest Order oraz metody jego rozliczenia

  - **Przyporządkowanie klas do wszystkich ról wzorca**

    Interfejs Strategii:Recipt
    Klasy konkretnych Strategii: Bill, InvoiceForPerson, InvoiceForFirm
    Kontekst: Order

  - **Lokalizację wzorca w kodzie**

    src/Order, src/Recipt, src/Bill, src/InvoiceForPerson, src/InvoiceForFirm

  - **Określenie wektora zmian tego wzorca w programie**

    sposób realizacji pewnego zadania

- ## Builder

  - **Cel użycia**

    Wzorzec Builder wykorzystany został do tworzenia obiektu Order. Ułatwiło to tworzenie własnego, spersonalizowanego zamówienia. Użytkownik aplikacji generuje scenariusz tworzenia zamówienia. Proces tworzenia zamówienia jest podzielony na mniejsze etapy, a każdy z nich może być implementowany na wiele sposobów.

  - **Lokalizację wzorca w kodzie**

    src/`Order`, src/`PersonBuilder`

  - **Określenie wektora zmian tego wzorca w programie**

    Wektor zmian we wzorcu Strategii może obejmować zmianę samego interfejsu algorytmów, dodanie nowych algorytmów lub usunięcie już istniejących, zmianę sposobu wyboru odpowiedniego algorytmu czy też rozszerzenie kontekstu w jakim ma działać algorytm.

- ## Dekorator

  - **Cel użycia**

    Wzorzec dekorator pozwolił na łatwe zmiany w cenie np. uwzględnienie zniżek lub wyboru sposobu dostawy co odpowiednio zmniejsza lub zwiększa finalną kwotę. Dodatkowo wzorzec ten pozwala na łatwą rozbudowę projektu o kolejne modyfikatory finalnych kosztów.

  - **Przyporządkowanie klas do wszystkich ról wzorca**

    Klasa Konkretnego Komponentu:Addition
    Klasa Dekoratora: Discount, ExpressDelivery

  - **Lokalizację wzorca w kodzie**

    src/Discount, src/ExpressDelivery, src/Addition,src/Order

  - **Określenie wektora zmian tego wzorca w programie**

    - dodanie nowych klas dekoratorów, które będą oferować nowe funkcjonalności dla obiektów
    - zmiana interfejsu klas dekoratorów lub obiektów, które są dekorowane
    - usunięcie już istniejących klas dekoratorów
    - zmiana sposobu tworzenia i konfiguracji zestawu dekoratorów dla obiektu
    - rozszerzenie kontekstu w jakim mają działać dekoratory

- ## Kompozyt

  - **Cel użycia**

    Wzorzec kompozyt został użyty do stworzenia drzewiastej struktury obiektów co pozwala grupować obiektów na działy, typy oraz kategorie. Do tworzenia nowych działów, typów lub kategorii nie wymagane jest utworzenie nowej klasy, a należy stworzyć tylko obiekt danej klasy.

  - **Przyporządkowanie klas do wszystkich ról wzorca**

    Klasa Komponentu podstawowego:Element
    Klasa Kompozytu: Product, Type, Category, Department

  - **Lokalizację wzorca w kodzie**

    src/Element, src/Product, src/Type, src/Category, src/Department

  - **Określenie wektora zmian tego wzorca w programie**

- Dodanie nowych klas komponentów, które będą oferować nowe funkcjonalności dla struktury drzewiastej
- Zmiana interfejsu klas komponentów lub sposobu tworzenia i konfiguracji struktury drzewiastej
- Usunięcie już istniejących klas komponentów
- Zmiana sposobu dostępu do poszczególnych elementów struktury drzewiastej
- Rozszerzenie kontekstu w jakim ma działać kompozyt

- ## Proxy

  - **Cel użycia**

W projekcie stworzono bazę danych poprzez interfejs Database i klasę implementującą - Library. Klasa Library pełni rolę pośrednika przekazując wszystkie wywołania do docelowego obiektu. Ułatwia to wykonywanie zapytań do bazy danych.

  - **Przyporządkowanie klas do wszystkich ról wzorca**

    Klasa Proxy:Library

  - **Lokalizację wzorca w kodzie**

    src/Library

  - **Określenie wektora zmian tego wzorca w programie**

- ## Fabryka Abstrakcyjna

    - **Cel użycia**

Wzorzec fabryka abstrakcyjna pozwala na tworzenie różnych obiektów należących do tej samej rodziny. Obiekty różnych rodzajów implementują jeden interfejs (Receipt), ale są różne. Metoda CreateFactory() jest metodą bezparametrową i w zależności od rodzaju fabryki tworzy odpowiedni rodzaj rachunku dzięki czemu nie trzeba robić tego samodzielnie.

    - **Przyporządkowanie klas do wszystkich ról wzorca**

Interfejs Fabryki:AbstractFactoryRecipt
Klasy konkretnych Fabryk:FactoryBill, FactoryInvoiceForPerson, FactoryInvoiceForFirm
Interfejs Produktu:Recipt
Klasy konkretnych Produktów:Bill, InvoiceForPerson, InvoiceForFirm

    - **Lokalizację wzorca w kodzie**

src/Recipt, src/Bill, src/InvoiceForPerson, src/InvoiceForFirm, src/FactoryBill, src/FactoryInvoiceForPerson, src/FactoryInvoiceForFirm, src/AbstractFactoryRecipt

    - **Określenie wektora zmian tego wzorca w programie**

- Dodanie nowych typów proxy, które będą oferować różne mechanizmy kontroli dostępu do obiektu
- Zmiana interfejsu proxy lub obiektu, który jest udostępniany przez proxy
- Usunięcie już istniejących typów proxy
- Zmiana sposobu tworzenia i konfiguracji proxy
- Rozszerzenie kontekstu w jakim ma działać proxy, np. obsługa wielowątkowości

- ## Iterator

    - **Cel użycia**

Wzorzec iterator odpowiada za poruszanie się po bazie danych oraz za odczytywanie jej poszczególnych rekordów. Metoda Next() pobiera element i przechodzi do kolejnego

    - **Lokalizację wzorca w kodzie**

src/Library

- **Określenie wektora zmian tego wzorca w programie**

- Dodanie nowych typów iteratorów, które będą oferować różne mechanizmy przeglądania elementów zbioru danych
- Zmiana interfejsu iteratorów lub klas obiektów, które mają być przeglądane
- Usunięcie już istniejących typów iteratorów
- Zmiana sposobu tworzenia i konfiguracji iteratorów
- Rozszerzenie kontekstu w jakim ma działać iterator, np. obsługa wielowątkowości

3. Opis rozwiązania specyficznego dla użytej technologii

Java Swing jest biblioteką do tworzenia interfejsu graficznego użytkownika (GUI) dla aplikacji Java. Jest to rozszerzenie biblioteki Java AWT (Abstract Window Toolkit) i umożliwia tworzenie bardziej zaawansowanych i atrakcyjnych interfejsów. Swing oferuje wiele komponentów, takich jak przyciski, listy, tabele, menu i inne, które pozwalają na tworzenie interfejsu. Aplikacje napisane z użyciem Java Swing są natywne dla systemu operacyjnego i działają płynnie.

4. Opis podziału pracy w zespole

· Michał Wołosewicz:

implementacja wzorca Iterator
implementacja wzorca Singleton
implementacja wzorca Proxy
implementacja wzorca Dekorator
implementacja wzorca Strategia
debugowanie

· Patryk Wójtowicz:

implementacja wzorca Fabryka Abstrakcyjna
implementacja wzorca Strategia
implementacja wzorca Proxy
implementacja wzorca Dekorator
implementacja wzorca Kompozyt
implementacja wzorca Builder
debugowanie

· Magda Zaborowska:

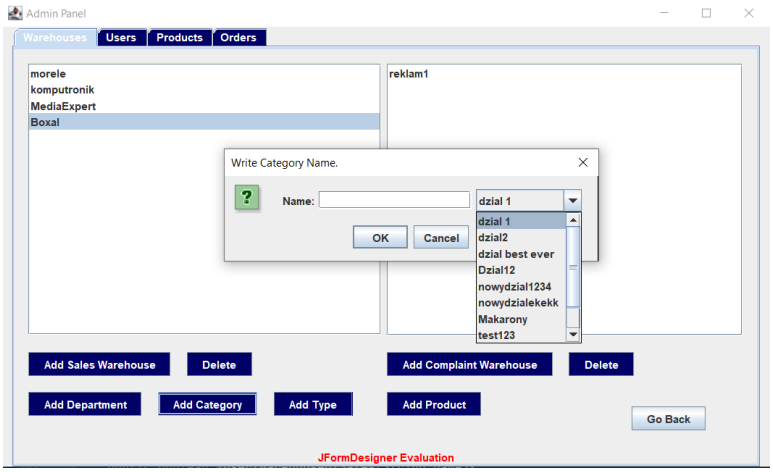
implementacja wzorca Singleton
implementacja wzorca Iterator

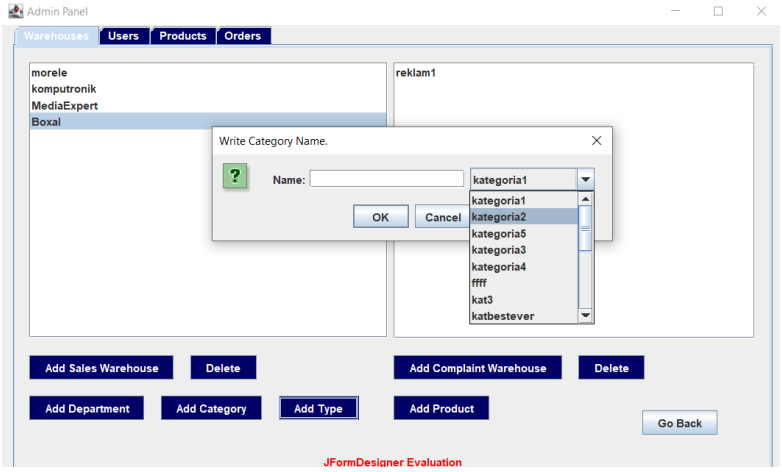
implementacja wzorca Kompozyt
implementacja wzorca Fabryka Abstrakcyjna
implementacja wzorca Builder
dokumentacja

5. Instrukcja użytkownika

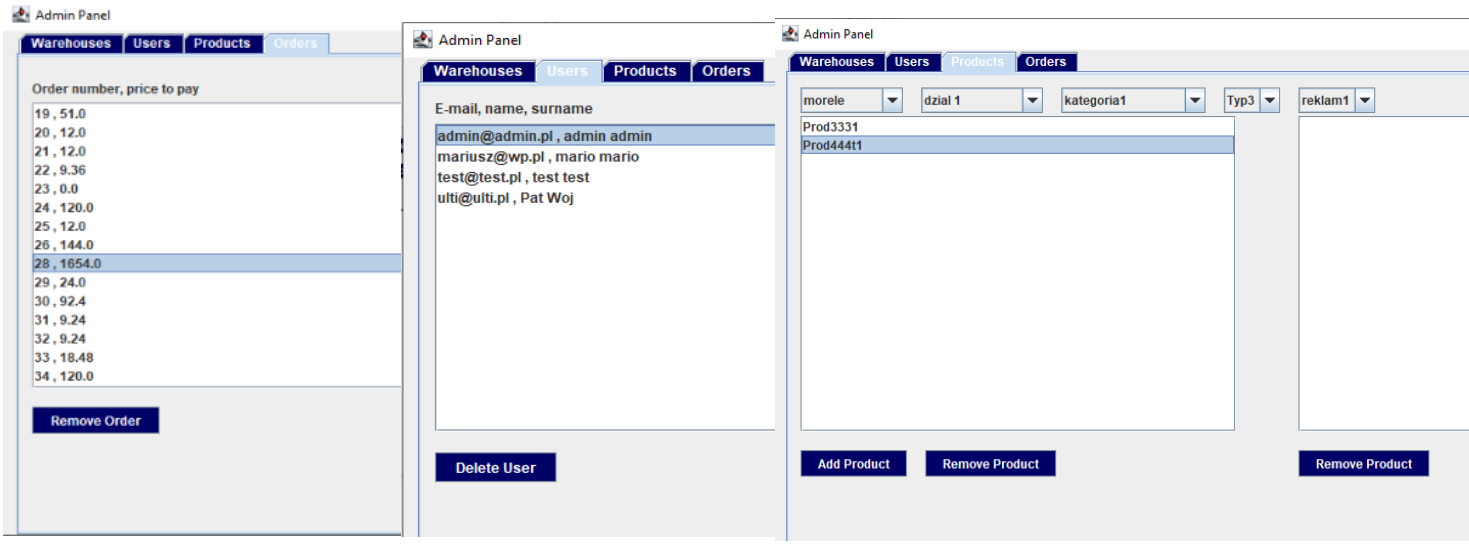
Po włączeniu aplikacji wyświetla się okno logowania. W przypadku gdy zaloguje się admin musi on wybrać czy chce przejść do panelu administracyjnego czy do sklepu.

W panelu administracyjnym na dole wyświetlają się przycisku otwierające okna do dodania nowego magazynu,oddziału, kategorii, typu i produktu. W każdym z przypadków wyświetli się okno z formularzem. Jeśli chcemy usunąć któryś rekord wystarczy go kliknąć i wybrać przycisk Delete.





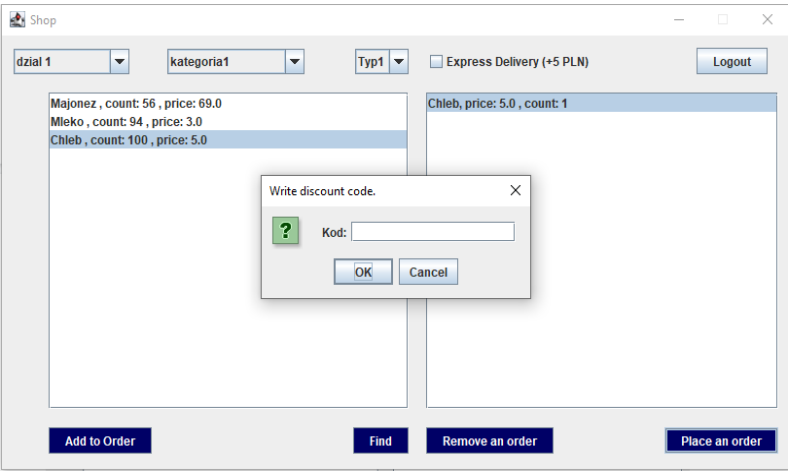
Na górze panelu administracyjnego mamy nawigację. Możemy przełączać się pomiędzy widokiem magazynów, użytkowników(wyświetla), produktów i zamówień.

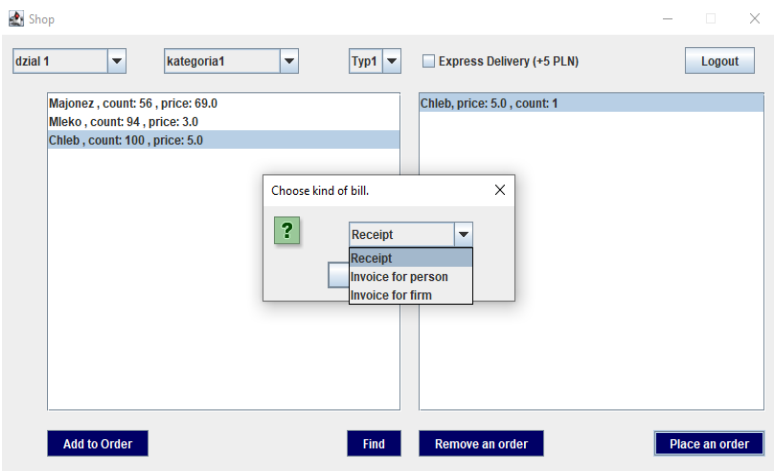


W panelu Users wyświetlają się dane użytkowników. W panelu Orders wyświetlają się numery zamówień wraz z ich wartością. W panelu Products mamy podgląd na produkty z danego magazynu, działu, kategorii, typu

Oprócz panelu Administracyjnego mamy też panel sklepu. Można w nim filtrować produkty po dziale, kategorii i typie. Aby dodać produkt do zamówienia, należy wybrać go i wcisnąć AddToOrder.

Po wybraniu Place an order wyświetla się formularz do wpisania kodu rabatowego. Jeżeli takowego nie znamy należy wcisnąć Cancel. Następnie wyświetli się wybór między paragonem a fakturą. W przypadku faktury, należy wprowadzić dane do faktury. Następnie tworzone jest zamówienie i wyświetla się jego podsumowanie.





6. Instrukcja instalacji

Należy pobrać SQLite oraz środowisko InteliJ wraz z jego bibliotekami. Następnie uruchomić aplikację w trybie debuger.