Programowanie w języku Java

Temat: Hurtownia RTV/AGD

Autorzy: Tomasz Muciek, Patryk Mendak, Mateusz Miernik

Drugi rok, Powtarzanie przedmiotu.

Opis Aplikacji

Aplikacja Hurtownia RTV/AGD to program pozwalający na zarządzanie i prowadzenie magazynu sprzętów codziennego użytku. Jest ona napisana w całości w języku Java(J2SE 1.8), przy wykorzystaniu takich technologii jak:

- Eclipse IDE,

- standardowa biblioteka graficzna Swing,

- baza danych Oracle,

- framework do mapowania obiektowo- relacyjnego Hibernate,

- system automatyzacji budowania projektów Gradle,

- system kontroli wersji Git(Gitlab),

- wzorzec projektowy MVC.

Projekt uruchamia się poprzez dwukrotne kliknięcie na ikonę paczki uruchamialnej .jar.

Przed rozpoczęciem korzystania z aplikacji, należy wybrać z menu w oknie głównym opcję „Pomoc", a następnie „Ustawienia". W otworzonym okienku należy podać dane dostępu do bazy danych Oracle. Po poprawnym wprowadzeniu danych można przetestować połączenie(przycisk „Wypróbuj Połączenie") i zapisać dane(przycisk „Zapisz"). Można rozpocząć pracę z aplikacją.

Interfejs graficzny

Interfejs składa się z:

- Okna głównego(MainView)
- Okienka powitalnego(GreetingsView)
- Okienka do wyświetlania pracowników(EmployeeView)
- Okienka do wyświetlania kontrahentów(ContractorView)
- Okienka do wyświetlania produktów(ProductView)
- Okienka do wyświetlania zamówień(OrderView)
- Panelu ustawień(SettingsView)
- Okienek formularzy do wprowadzania danych(Dialog).

MainView

Jest to okno składające się z dwóch elementów: belki menu i panelu roboczego. W menu dostępne są dwie opcje: Zarządzaj i Pomoc. Pierwsza opcja rozwija się na 4 elementy, służące do otwarcia w obszarze panelu roboczego okna do zarządzania: zamówieniami, kontrahentami, magazynem(produktami) oraz pracownikami. Opcja „Pomoc" rozwija się na dwa elementy: ustawienia i dokumentacja. Ustawienia otwierają panel ustawień(patrz: SettingsView), natomiast dokumentacja odwołuje się do tego dokumentu.

Okna, będące w panelu roboczym, można dowolnie przenosić, minimalizować, maksymalizować, zmieniać rozmiar ręcznie w obrębie panelu.

GreetingsView

GreetingsView to okno powitalne służące do szybkiego dostępu do zarządzania encjami z bazy danych. Jest to pierwsze okno uruchamiane natychmiast to otworzeniu aplikacji. Nie da się go wywołać z menu.

Posiada 4 przyciski:

- Zamówienia,
- Kontrahenci,
- Magazyn,
- Personel.

EmployeeView

Okno to służy do zarządzania zatrudnionymi pracownikami. Wyświetla zatrudnionych pracowników w formie tabeli. Tabelę można dowolnie dostosowywać, tj. zmieniać kolejność wyświetlanych danych(poprzez przeciąganie nazw kolumn), zmieniać rozmiar kolumn.

Pod tabelą znajdują się przyciski do zarządzania danymi. Za ich pomocą można kolejno: usuwać wpisy(po zaznaczeniu wpisu, trzeba kliknąć przycisk Usuń Rekord), dodawać nowe wpisy w bazie(przycisk Dodaj Rekord) oraz edytować wyświetlane wpisy(po zaznaczeniu wpisu, trzeba kliknąć przycisk Edytuj Dane). Dwie ostatnie opcje wywołują okienko dialogowe z formularzem do manipulowania danymi.

Dialog składa się z nazw pól klasy encji Employee i pól tekstowych do wprowadzania/modyfikowania danych. Dialog posiada także dwa przyciski: „OK"(zatwierdza zmiany) i „Anuluj"(odrzuca je). Po ich kliknięciu okno formularza zostaje zamknięte. Jeśli dane będą wprowadzone błędnie, program zaznaczy je na czerwono. Należy zatem pamiętać, aby nazwy były rozpoczynane z wielkich liter, a daty wpisywane w formacie dd.mm.rrrr(np. 20.06.2018).

ContractorView

Okno to służy do zarządzania kontrahentami. Wyświetla kontrahentów hurtowni w formie tabeli. Tabelę można dowolnie dostosowywać, tj. zmieniać kolejność wyświetlanych danych(poprzez przeciąganie nazw kolumn), zmieniać rozmiar kolumn.

Pod tabelą znajdują się przyciski do zarządzania danymi. Za ich pomocą można kolejno: usuwać wpisy(po zaznaczeniu wpisu, trzeba kliknąć przycisk Usuń Rekord), dodawać nowe wpisy w bazie(przycisk Dodaj Rekord) oraz edytować wyświetlane wpisy(po zaznaczeniu wpisu, trzeba kliknąć przycisk Edytuj Dane). Dwie ostatnie opcje wywołują okienko dialogowe z formularzem do manipulowania danymi.

Dialog składa się z nazw pól klasy encji Contractor i pól tekstowych(z wyjątkiem checkboxa przy dostawcy) do wprowadzania/modyfikowania danych. Dialog posiada także dwa przyciski: „OK”(zatwierdza zmiany) i „Anuluj”(odrzuca je). Po ich kliknięciu okno formularza zostaje zamknięte. Jeśli dane będą wprowadzone błędnie, program zaznaczy je na czerwono. Należy zatem pamiętać, aby nazwy były rozpoczynane z wielkich liter, a daty wpisywane w formacie dd.mm.rrrr(np. 20.06.2018).

ProductView

Okno to służy do zarządzania produktami znajdującymi się w ofercie hurtowni. Wyświetla produkty w formie tabeli. Tabelę można dowolnie dostosowywać, tj. zmieniać kolejność wyświetlanych danych(poprzez przeciąganie nazw kolumn), zmieniać rozmiar kolumn.

Pod tabelą znajdują się przyciski do zarządzania danymi. Za ich pomocą można kolejno: usuwać wpisy(po zaznaczeniu wpisu, trzeba kliknąć przycisk Usuń Rekord), dodawać nowe wpisy w bazie(przycisk Dodaj Rekord) oraz edytować wyświetlane wpisy(po zaznaczeniu wpisu, trzeba kliknąć przycisk Edytuj Dane). Dwie ostatnie opcje wywołują okienko dialogowe z formularzem do manipulowania danymi.

Dialog składa się z nazw pól klasy encji Product i pól tekstowych do wprowadzania/modyfikowania danych. Dialog posiada także dwa przyciski: „OK”(zatwierdza zmiany) i „Anuluj”(odrzuca je). Po ich kliknięciu okno formularza zostaje zamknięte. Jeśli dane będą wprowadzone błędnie, program zaznaczy je na czerwono. Należy zatem pamiętać, aby nazwy były rozpoczynane z wielkich liter.

ProductView

Okno to służy do zarządzania zleceniami wykonanymi przez hurtownię. Wyświetla zamówienia w formie tabeli. Tabelę można dowolnie dostosowywać, tj. zmieniać kolejność wyświetlanych danych(poprzez przeciąganie nazw kolumn), zmieniać rozmiar kolumn.

Pod tabelą znajdują się przyciski do zarządzania danymi. Za ich pomocą można kolejno: usuwać wpisy(po zaznaczeniu wpisu, trzeba kliknąć przycisk Usuń Rekord), dodawać nowe wpisy w

bazie(przycisk Dodaj Rekord) oraz edytować wyświetlane wpisy(po zaznaczeniu wpisu, trzeba kliknąć przycisk Edytuj Dane). Dwie ostatnie opcje wywołują okienko dialogowe z formularzem do manipulowania danymi.

Dialog składa się z etykiet oraz: pól tekstowych, list rozwijanych oraz specjalnego pola z listą produków powiązaną z danym zamówieniem. Dialog posiada także przyciski:

„dodaj”(dodaje wybrany produkt we wpisanej ilości do listy),

„usuń”(usuwa zaznaczony produkt z listy),

„OK”(zatwierdza zmiany)

„Anuluj”(odrzuca je).

Po ich kliknięciu okno formularza zostaje zamknięte. Jeśli dane będą wprowadzone błędnie, program zaznaczy je na czerwono. Należy zatem pamiętać, aby data wpisana była w formacie dd.mm.rrrr(np. 20.06.2018) oraz lista produktów nie była pusta. W przypadku próby dodania produktu istniejącego już na liście, program wyświetli komunikat o istniejącym produkcie i zignoruje polecenie. W razie potrzeby zmiany ilości produktu, należy usunąć wpis i dodać od nowa z poprawną ilością.

SettingsView

Okno to służy do ustawienia danych potrzebnych do połączenia programu z bazą. Wymagane dane to: nazwa użytkownika bazy danych Oracle, hasło, url połączenia. Istnieje możliwość przetestowania danych połączenia. W tym celu należy kliknąć przycisk „Wypróbuj Połączenie” i sprawdzić komunikat.

Po wprowadzeniu zmian, należy zapisać je, klikając przycisk „Zapisz”.

Uwaga! Po zmianie danych połączenia z bazą danych może być wymagany restart aplikacji do jej poprawnego działania.

Dokumentacja kodu

Kod został napisany we wzorcu projektowym MVC. Każdy obiekt, w szczególności encje, posiada swój model- przechowujący stan, łączący się z bazą danych(podwzorzec Observer), widok- wyświetlający dane modelu użytownikowi oraz kontroler- reagujący na akcje użytkownika i zmieniający stan modelu.

Opis klas i metod:

Niniejsza dokumentacja jest uzupełnieniem informacji zawartych w javadoc’ach.

Klasa: Main

Posiada wyłącznie metodę główną powodującą rozruch aplikacji, a więc: ustawienie wyglądu „look and feel” oraz stworzenie modelu, widoku i kontrolera obiektu Main. Stworzenie tych instancji powoduje wyświetlenie się okienka głównego(MainView).

Klasa: entities.Contractor

Klasa przedstawiająca encję kontrahenta. Posiada następujące właściwości:

nip- podstawowy identyfikator, dziesięciocyfrowy

name – nazwa

phoneNumber- numer telefonu kontrahenta

city- miasto

street- ulica

houseNumber- nr domu

apartmentNumber- nr mieszkania/lokalu

isSupplier- czy jest dostawcą

Klasa: entities.Employee

Klasa przedstawiająca encję pracownika. Posiada następujące właściwości:

id- identyfikator, automatycznie generowany

name- imię

surname- nazwisko

position- stanowisko

salary- pensja

city- miasto

street- ulica

houseNumber- nr domu

apartmentNumber- nr mieszkania/lokalu

employmentDate- data zatrudnienia

Klasa: entities.OrderProduct

Klasa przedstawiająca połączenie informacji klasy Order oraz Product, służąca tworzeniu list produktów w Zamówieniach. Posiada następujące właściwości:

id- identyfikator, automatycznie generowany

product- pole przechowujące Produkt(Join na kod produktu)

numberOfProducts- ilość produktu

Klasa: entities.Product

Klasa przedstawiająca encję produktu. Posiada następujące właściwości:

code- identyfikator, 6 cyfrowy kod

manufacturer- producent

name- nazwa

specification- specyfikacja produktu, miejsce na szczegółowe dane

amount- ilość produktu na magazynie

price- cena za sztukę

category- kategoria, dział do którego należy na magazynie

weight- waga jednej sztuki, podawana w kilogramach, z dokładnością do 3 miejsc po przecinku(gram)

Klasa: entities.PurchaseOrder

Klasa przedstawiająca zamówienie. Zamówienie składa się z

orderId- identyfikator, automatycznie generowany

orderDate- data realizacji zamówienia

productsList- lista produktów z ich ilościami(OrderProduct)- każdy element(produkt) nie może występować więcej niż raz

contractor- kontrahent kupujący, sprzedający- w zależności od wartości pola idSupplier(join na nip kontrahenta)

Klasa model.ContractorModel

Klasa odpowiadająca za komunikację z bazą oraz informowanie obserwujących o zmianie jej stanu.

Implementuje interfejs „ModelInterface”

Metody:

Konstruktor bezparametrowy powoduje zainicjalizowanie listy obserwujących.

Oprócz tego znajdziemy tutaj metody służące do:

- pobrania Kontrahenta z bazy, z wykorzystaniem identyfikatora;
- pobrania listy(ArrayList) wszystkich Encji
- dodania nowego wpisu do bazy
- zmiany danych istniejącego wpisu
- usunięcia wpisu
- dodania obserwującego
- usunięcia obserwującego
- powiadomienia wszystkich obserwujących o zmianie stanu

Klasa model.EmployeeModel

Klasa odpowiadająca za komunikację z bazą oraz informowanie obserwujących o zmianie jej stanu.

Implementuje interfejs „ModelInterface”

Metody:

Konstruktor bezparametrowy powoduje zainicjalizowanie listy obserwujących.

Oprócz tego znajdziemy tutaj metody służące do:

- pobrania Kontrahenta z bazy, z wykorzystaniem identyfikatora;
- pobrania listy(ArrayList) wszystkich Encji
- dodania nowego wpisu do bazy
- zmiany danych istniejącego wpisu
- usunięcia wpisu
- dodania obserwującego
- usunięcia obserwującego
- powiadomienia wszystkich obserwujących o zmianie stanu

Klasa model.OrderProductModel

Klasa odpowiadająca za komunikację z bazą oraz informowanie obserwujących o zmianie jej stanu.

Reprezentuje połączenie Produktu z Zamówieniem z dodatkową informacją o ilości produktów.

Implementuje interfejs „ModelInterface”

Metody:

Konstruktor bezparametrowy powoduje zainicjalizowanie listy obserwujących.

Oprócz tego znajdziemy tutaj metody służące do:

- pobrania ZamówieniaProduktu z bazy, z wykorzystaniem identyfikatora;
- pobrania listy(ArrayList) wszystkich wpisów w bazie
- dodania nowego wpisu do bazy
- zmiany danych istniejącego wpisu
- usunięcia wpisu
- dodania obserwującego
- usunięcia obserwującego
- powiadomienia wszystkich obserwujących o zmianie stanu

Klasa model.OrderModel

Klasa odpowiadająca za komunikację z bazą oraz informowanie obserwujących o zmianie jej stanu.

Implementuje interfejs „ModelInterface”

Metody:

Konstruktor bezparametrowy powoduje zainicjalizowanie listy obserwujących.

Oprócz tego znajdziemy tutaj metody służące do:

- pobrania Zamówienia z bazy, z wykorzystaniem identyfikatora;
- pobrania listy(ArrayList) wszystkich Encji
- dodania nowego wpisu do bazy
- zmiany danych istniejącego wpisu
- usunięcia wpisu

- dodania obserwującego
- usunięcia obserwującego
- powiadomienia wszystkich obserwujących o zmianie stanu

Klasa model.ProductModel

Klasa odpowiadająca za komunikację z bazą oraz informowanie obserwujących o zmianie jej stanu.

Implementuje interfejs „ModelInterface”

Metody:

Konstruktor bezparametrowy powoduje zainicjalizowanie listy obserwujących.

Oprócz tego znajdziemy tutaj metody służące do:

- pobrania Produktu z bazy, z wykorzystaniem identyfikatora;
- pobrania listy(ArrayList) wszystkich Encji
- dodania nowego wpisu do bazy
- zmiany danych istniejącego wpisu
- usunięcia wpisu
- dodania obserwującego
- usunięcia obserwującego
- powiadomienia wszystkich obserwujących o zmianie stanu

Klasa model.SettingsModel

Klasa odpowiadająca za zmianę i przechowywanie danych połączeniowych z bazą Oracle. Dane te są zapisywane w pliku tekstowym db.properties.

Implementuje interfejs „ModelInterface”.

Metody:

Konstruktor bezparametrowy powoduje zainicjalizowanie zmiennych oraz załadowanie danych z pliku do obiektu typu Properties.

Oprócz tego znajdziemy tutaj metody służące do:

- Pobrania wyłącznie url połączenia
- Pobrania nazwy użytkownika bazy danych

- Pobrania hasła użytkownika bazy danych
- Zapisania ustawień
- Sprawdzenia połączenia(zwraca tekst wiadomości do użytkownika)
- dodania obserwującego
- usunięcia obserwującego
- powiadomienia wszystkich obserwujących o zmianie stanu

Klasa model.MainModel

Klasa przechowująca informacje o następnym wolnym położeniu okna.

Implementuje interfejs „ModelInterface".Posiada stałą FREE_POSITION_JUMP zawierającą o długości przeskoku okien w pikselach, w poziomie.

Metody:

Konstruktor bezparametrowy powoduje zainicjalizowanie zmiennej freePosition.

Oprócz tego znajdziemy tutaj metody służące do:

Zmiany wolnej pozycji i pobrania wolnej pozycji.

Klasa model.HibernateUtil

Jest to to klasa umożliwiająca obsługę bazy dzięki frameworkowi mapowania obiektowo-relacyjnego Hibernate.

Posiada metody:

BuildSessionFactory- konfiguruje sesję oraz metodę do pobrania dostępu do sesji.

Interfejs model.ModelInterface

interfejs służący do implementacji wzorca Observer dla strony obserwowalnej

Klasa view.ContractorView

Klasa odpowiadająca za wyświetlanie wpisów Kontrahenta z bazy danych. Dziedziczy po JinternalFrame, implementuje ModelObserver

Przechowywane przez nią dane we właściwościach to przede wszystkim: nazwy kolumn tabeli, lista wyświetlanych encji, model, dialog, kontroler oraz klasy potrzebne do rysowania tabeli.

Posiada konstruktor z parametrami: ContractorModel model, ActionListener controller, int x, int y. Są one potrzebne do ustwienia pól klasy, dzięki którym możliwe jest pobieranie danych z bazy, przekazywanie reakcji na akcje użytkownika do kontrolera oraz do określenia położenia okna na panelu głównym.

Metody, które posiada:

- pobranie nazw kolumn tabeli
- pobranie panelu z przyciskami
- pobranie zaznaczonego wpisu jako instancję klasy Contractor
- pokazanie dialogu
- pobranie instancji dialogu
- ustawienie danych tabeli
- odświeżenie widoku- pobranie danych z modelu

Klasa view.EmployeeView

Klasa odpowiadająca za wyświetlanie wpisów Pracownika z bazy danych. Dziedziczy po JinternalFrame, implementuje ModelObserver

Przechowywane przez nią dane we właściwościach to przede wszystkim: nazwy kolumn tabeli, lista wyświetlanych encji, model, dialog, kontroler oraz klasy potrzebne do rysowania tabeli.

Posiada konstruktor z parametrami: EmployeeModel model, ActionListener controller, int x, int y. Są one potrzebne do ustwienia pól klasy, dzięki którym możliwe jest pobieranie danych z bazy, przekazywanie reakcji na akcje użytkownika do kontrolera oraz do określenia położenia okna na panelu głównym.

Metody, które posiada:

- pobranie nazw kolumn tabeli
- pobranie panelu z przyciskami
- pobranie zaznaczonego wpisu jako instancję klasy Employee
- pokazanie dialogu
- pobranie instancji dialogu

- ustawienie danych tabeli
- odświeżenie widoku- pobranie danych z modelu

Klasa view.OrderView

Klasa odpowiadająca za wyświetlanie wpisów Zamówienia z bazy danych. Dziedziczy po JinternalFrame, implementuje ModelObserver

Przechowywane przez nią dane we właściwościach to przede wszystkim: nazwy kolumn tabeli, lista wyświetlanych encji, model, dialog, kontroler oraz klasy potrzebne do rysowania tabeli. Dodatkowo przechowuje modele dla encji: Produktu, Kontrahenta i ZamówienieProdukt, gdyż są one niezbędne do poprawnego działania podwidoku- dialogu.

Posiada konstruktor z parametrami: OrderModel model, ActionListener controller, int x, int y. Są one potrzebne do ustwienia pól klasy, dzięki którym możliwe jest pobieranie danych z bazy, przekazywanie reakcji na akcje użytkownika do kontrolera oraz do określenia położenia okna na panelu głównym.

Metody, które posiada:

- pobranie nazw kolumn tabeli
- pobranie panelu z przyciskami
- pobranie zaznaczonego wpisu jako instancję klasy PurchaseOrder
- pokazanie dialogu
- pobranie instancji dialogu
- ustawienie danych tabeli
- odświeżenie widoku- pobranie danych z modelu

Klasa view.ProductView

Klasa odpowiadająca za wyświetlanie wpisów Produktów z bazy danych. Dziedziczy po JinternalFrame, implementuje ModelObserver

Przechowywane przez nią dane we właściwościach to przede wszystkim: nazwy kolumn tabeli, lista wyświetlanych encji, model, dialog, kontroler oraz klasy potrzebne do rysowania tabeli.

Posiada konstruktor z parametrami: ProductModel model, ActionListener controller, int x, int y. Są one potrzebne do ustwienia pól klasy, dzięki którym możliwe jest pobieranie danych z bazy, przekazywanie reakcji na akcje użytkownika do kontrolera oraz do określenia położenia okna na panelu głównym.

Metody, które posiada:

- pobranie nazw kolumn tabeli
- pobranie panelu z przyciskami
- pobranie zaznaczonego wpisu jako instancję klasy Product
- pokazanie dialogu
- pobranie instancji dialogu
- ustawienie danych tabeli
- odświeżenie widoku- pobranie danych z modelu

Klasa view.SettingsView

Klasa odpowiedzialna za wyświetlanie okienka z ustawieniami połączenia.

Konstruktor klasy odpowiedzialny jest za wyrysowanie okna, wraz z zawartością i danymi, pobranymi z modelu. Jedynym parametrem konstruktora jest model obiektu ustawień(SettingsModel).

Klasa dziedziczy po Jframe i implementuje interfejs ModelObserver.

Posiada metody służące do pobrania instancji obiektów wyświetlanych przycisków oraz pól tekstowych. Ponadto posiada metodę do ustanowienia kontrolera dla przycisków.

Klasa view.MainView

Klasa MainView jest oknem głównym aplikacji. Posiada panel roboczy, na którym wyświetlane są okna oraz menu, z którego wybieramy, nad czym chcemy pracować.

Jedyny konstruktor posiada jeden parametr controller(ActionListener) i powoduje rozpoczęcie działania aplikacji- pokazanie widoku użytkownikowi.

Posiada pola: modele, widoki i kontrolery wszystkich encji i obiektów oraz widok powitalny, instancję klasy belki menu.

Metody:

- do ustawienia modelu głównego
- do pokazania okienka: ustawień, produktów, kontrahentów, zamówień, pracowników oraz powitania
- do pobrania okienka powitalnego
- do sprawdzenia czy okienko jest już wyświetlane
- do pobrania itemów z belki menu

Klasa view.GreetingsView

Klasa podwidoku widoku głównego służąca do szybkiego dostępu do okien zarządzania encjami.

Konstruktor powoduje wyrysowanie okienka powitalnego.

Posiada pola: komponentów tekstowych, głównego panelu wewnętrznego i 4 przycisków encji.

Metody: do pobierania przycisków oraz metoda do ustawienia kontrolera dla przycisków.

Klasa view.MenuBar

Klasa przedstawiająca belkę menu. W niej przyciski: Zarządzanie i Pomoc oraz podprzyciski: do zarządzania encjami , wyświetlania dokumentacji i ustawień połączenia. Każdy przycisk menu posiada swój przycisk mnemoniczny, po którym można za pomocą klawiatury wywołać go.

Konstruktor powoduje inicjalizację komponentów graficznych i stworzenie menu.

Metody: do pobierania komponentów i do ustanowienia kontrolerów dla nich.

Klasa view.TableButtonsPanel

Posiada przyciski tabeli: dodaj rekord, usuń i edytuj.

Konstruktor tworzy panel z przyciskami i ustanawia kontroler dla nich.

Metody: do pobierania przycisków

Klasa view.ProductsListPanel

Służy do wyświetlania Listy ZamówienieProdukt, którą przechowuje Zamówienie.

Konstruktor tworzy komponent graficzny.

Metody służące do:

- pobierania listy
- ustawienia listy
- usuwania zaznaczonego ZamówienieProdukt
- dodawania ZamówienieProdukt

Klasa view.OkCancelButtonsPanel

Klasa OkCancelButtonsPanel to jak nazwa wskazuje, panel z przyciskami „OK” i „Anuluj”, wykorzystywana przez każdy dialog.

Interfejs view.ModelObserver

interfejs implementujący wzrorzec Observer od strony obserwującego, posiada metodę update.

Klasa controller.EmployeeController

Klasa odpowiadająca za reakcję na akcje użytkownika, wsyłane przez przyciski. Kontroler modyfikuje model i zmusza go do powiadomienia obserwujących o zmianie. Implementuje interfejs ActionListener.

Posiada właściwości: model i widok.

Metoda actionPerformed zawiera reakcje na naciśnięcie przycisków z okna Employee View i Dialog.

Klasa controller.ProductController

Klasa odpowiadająca za reakcję na akcje użytkownika, wsyłane przez przyciski. Kontroler modyfikuje model i zmusza go do powiadomienia obserwujących o zmianie. Implementuje interfejs ActionListener.

Posiada właściwości: model i widok.

Metoda actionPerformed zawiera reakcje na naciśnięcie przycisków z okna Product View i Dialog.

Klasa controller.ContractorController

Klasa odpowiadająca za reakcję na akcje użytkownika, wsyłane przez przyciski. Kontroler modyfikuje model i zmusza go do powiadomienia obserwujących o zmianie. Implementuje interfejs ActionListener.

Posiada właściwości: model i widok.

Metoda actionPerformed zawiera reakcje na naciśnięcie przycisków z okna Contractor View i Dialog.

Klasa controller.OrderController

Klasa odpowiadająca za reakcję na akcje użytkownika, wsyłane przez przyciski. Kontroler modyfikuje model i zmusza go do powiadomienia obserwujących o zmianie. Implementuje interfejs ActionListener.

Posiada właściwości: model, widok i modelZamówienieProdukt.

Metoda actionPerformed zawiera reakcje na naciśnięcie przycisków z okna Order View i Dialog.

Klasa controller.SettingsController

Klasa odpowiadająca za reakcję na akcje użytkownika, wsyłane przez przyciski. Kontroler modyfikuje model(plik db.properties). Implementuje interfejs ActionListener.

Posiada właściwości: settingsModel i settingsView.

Metoda actionPerformed zawiera reakcje na naciśnięcie przycisków z okna Contractor View i Dialog.

Klasa controller.MainController

Klasa odpowiadająca za reakcję na akcje użytkownika, wysłane przez przyciski w GreetingsView i MenuBar. Kontroler modyfikuje model i zmusza go do powiadomienia obserwujących o zmianie. Implementuje interfejs ActionListener.

Posiada właściwości: view i model;

Klasa view.dialog.ContractorDialog

Klasa dialogowa reprezentująca w graficzny sposób formularz, służący do dodawania i edytowania Kontrahenta.

Pola:

- contextValues- tablica z wartościami pól encji, ułatwiająca wypełnianie formularza,
- buttonsPanel- obiekt panelu z przyciskami „OK” i „Anuluj”
- jtxtField- tablica zawierająca komponenty formularza umożliwiające modyfikację danych
- context- instancja encji tworzonej/edytowaniej
- isNew- czy encja jest nowa czy podlega edycji

Konstruktor rysuje okno formularza i przypisuje wartości polom. Parametry:

- parent- View dla danej encji,
- context- instancja encji,
- isNew- czy instancja reprezentuje nowy obiekt czy już istniejący,
- controller- controler do obsługi akcji przycisków.

Zadeklarowane metody służą do:

- sprawdzenia poprawności wprowadzonych danych
- pobrania instancji encji zawierającej dane z formularza
- pobranie kontextu podanego w konstruktorze
- pobranie panelu z przyciskami „OK” i „Anuluj”
- pobranie wartości pola isNew

Klasa view.dialog.EmployeeDialog

Klasa dialogowa reprezentująca w graficzny sposób formularz, służący do dodawania i edytowania Pracownika.

Pola:

- contextValues- tablica z wartościami pól encji, ułatwiająca wypełnianie formularza,
- buttonsPanel- obiekt panelu z przyciskami „OK” i „Anuluj”
- jtxtField- tablica zawierająca komponenty formularza umożliwiające modyfikację danych
- context- instancja encji tworzonej/edytowaniej
- isNew- czy encja jest nowa czy podlega edycji

Konstruktor rysuje okno formularza i przypisuje wartości polom. Parametry:

- parent- View dla danej encji,
- context- instancja encji,
- isNew- czy instancja reprezentuje nowy obiekt czy już istniejący,
- controller- controler do obsługi akcji przycisków.

Zadeklarowane metody służą do:

- sprawdzenia poprawności wprowadzonych danych
- pobrania instancji encji zawierającej dane z formularza
- pobranie kontextu podanego w konstruktorze
- pobranie panelu z przyciskami „OK” i „Anuluj”
- pobranie wartości pola isNew

Klasa view.dialog.ProductDialog

Klasa dialogowa reprezentująca w graficzny sposób formularz, służący do dodawania i edytowania Produktu.

Pola:

- contextValues- tablica z wartościami pól encji, ułatwiająca wypełnianie formularza,
- buttonsPanel- obiekt panelu z przyciskami „OK” i „Anuluj”
- jtxtField- tablica zawierająca komponenty formularza umożliwiające modyfikację danych
- context- instancja encji tworzonej/edytowaniej
- isNew- czy encja jest nowa czy podlega edycji

Konstruktor rysuje okno formularza i przypisuje wartości polom. Parametry:

- parent- View dla danej encji,
- context- instancja encji,
- isNew- czy instancja reprezentuje nowy obiekt czy już istniejący,
- controller- controler do obsługi akcji przycisków.

Zadeklarowane metody służą do:

- sprawdzenia poprawności wprowadzonych danych
- pobrania instancji encji zawierającej dane z formularza
- pobranie kontextu podanego w konstruktorze
- pobranie panelu z przyciskami „OK" i „Anuluj"
- pobranie wartości pola isNew

Klasa view.dialog.OrderDialog

Klasa dialogowa reprezentująca w graficzny sposób formularz służący do dodawania i edytowania Zamówienia. Za jej pomocą dodawane są także ZamówienieProdukt

Pola: wymieniaone w konstruktorze+ obiekty pól tekstowych, panel z produktami, listy rozwijane Kontrahentów i Produktów, przyciski, lista wyświetlanych ZamówienieProdukt.

Konstruktor rysuje okno formularza i przypisuje wartości polom. Parametry:

- parent- View dla danej encji, za pomocą tej referencji pobierane są dane z Modeli,
- context- instancja encji,
- isNew- czy instancja reprezentuje nowy obiekt czy już istniejący,
- controller- controler do obsługi akcji przycisków.

Zadeklarowane metody służą do:

- sprawdzenia poprawności wprowadzonych danych
- sprawdzenia poprawności wprowadzonej ilości przy dodawaniu ZamówienieProdukt
- pobrania instancji encji zawierającej dane z formularza
- pobranie kontextu podanego w konstruktorze
- pobranie panelu z przyciskami „OK" i „Anuluj"
- pobrania panelu z listą ZamówienieProdukt
- pobranie wartości pola isNew

- pobrania przycisków i list rozwijanych
- pobrania danych zaznaczonego produktu w liście rozwijanej z modelu rodzica
- pobrania danych o produkcie z rozwijanej listy
- utworzenie listy ciągów znaków na podstawie wyświetlanych ZamówienieProdukt
- sprawdzenie czy Produkt o podanym kodzie został już dodany do listy
- pobranie ZamówienieProdukt na podstawie wygenerowanego ciągu znaków

Podział obowiązków:

Patryk Mendak:

1. Zarządzanie systemem kontroli wersji
2. Stworzenie logiki programu(model klas, ich relacji, potrzebnych danych-encji)
3. Stworzenie kontrolerów
4. Dokumentacja(javadoc)

Tomasz Muciek:

1. Stworzenie Graficznego Interfejsu Użytkownika
2. Tworzenie potrzebnych klas widoków
3. Dokumentacja(pdf)

Mateusz Miernik:

1. Obsługa Hibernate
2. Napisanie klas encji
3. Stworzenie Modelu
4. Zarządzanie gradlem
5. Ogólna konfiguracja projektu w Eclipse