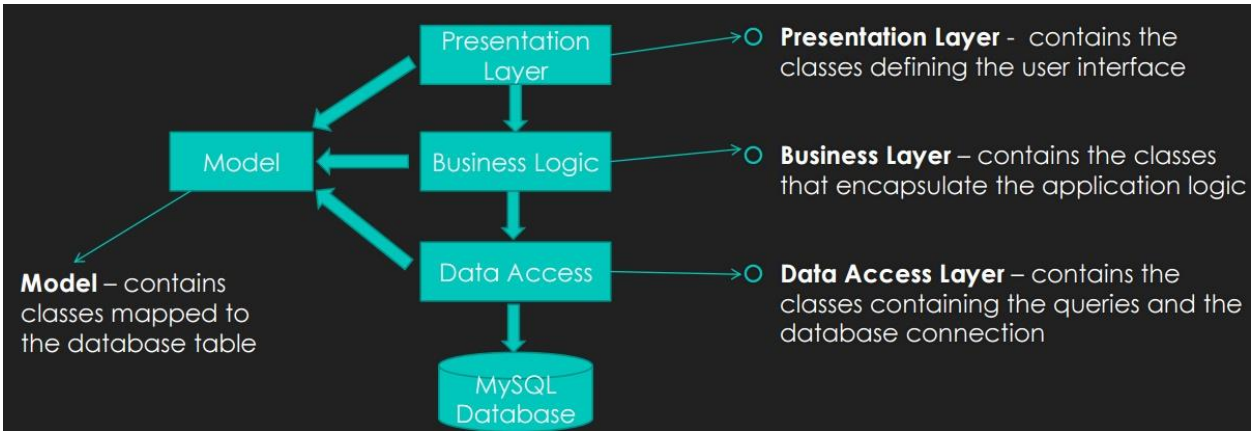
Warehouse

Decription

Acest proiect porneste de la cerinta de a realiza o aplicatie de management pentru procesarea clientilor, a produselor si a comenzilor intr-un depozit. Datele despre aceste trei entitati trebuie pastrate intr-o baza de date locala si prelucrate prin intermediul unei interfete grafice care se conecteaza la o schema. Pachetele folosite respecta ordinea de management si sunt denumite sugestiv: Presentatio Layer, Business Logic si Data Acces. Fiecare dintre acestea este conectat la Model care reprezinta maparea field-urilor din baza de date in obiecte Java. Partea de prezentare contine trei mari clase : View, Controller, GenericData. Pachetul e in stransa legatura cu pachetul de business si contine partea de prezentare, obiectele grafice, legatura dintre cine trebuie sa faca (listenere) si ce trebuie sa faca (prelucrare model) si plasarea generica a datelor(in tabele grafice). Business Layer poate accesa pachetul de date si detine clase care apeleaza metode de insert, delete, find (selectie) si update pe schema creata. Pachetul Data Access asigura conexiunea la baza de date prin folosirea unui obiect de clasa Connection Factory. In total numarul de pachete atinge cifra sapte iar toate accestea incapsuleaza in total 18 clase si o interfata. Clasicul tipar Model –View –Controller este urmat, fiind doar imbunatatit prin divizarea in pahete si limitarea accesarii globale a anumitelor obiecte. Pentru prelucrarea bazei de date s-a folosit cod SQL cu statement-uri clasice (insert, select, update si delete). MySQL Workbench retine local baza de date si printr-un “jar” se asigura conectarea la schema. Pentru acest proiect s-a folosit InteliJ Idea si varianta Java 8 cu librari precum “java.awt”, “java.swing”, “java.lang”, “java.util”,”java.beans” si “java.sql”.

Singeorzan Florin
Gr: 30227



Pachete si clase componente

Model: Customer | Order | Product

Customer

Clasa contine campurile id, name, address, phone si email alaturi de un constructor care initializeaza aceste valori la valorile trimie ca parametri constructorului. Pe langa acestea getter-ele si setter-ele aigura acesul la campurile private.

Product

Contine obiecte private: id_product, name, price, description si quantity; gettere si settere publice care asigura accesul la date si un constructor ce atribuie valori obiectelor create

Order

Aceasta clasa retine campurile id_order, id_client, id_product si quantity. Asemena ccelorlalte clase din acest pachet contine gettere, settere si un constructor pentru initializare.

DAO: AbstractDAO | CustomerDAO | ProductDAO | OrderDAO

AbstractDAO

Clasa detne un obiect generc <T> e care va imlementa oeratii de inserare,

cautare, delete si update. Contine un logger in care se vor scrie warning-urile aparute pe parcurs si un obiect specificGenercClass de tipul clasei T. Construtorul clasei aruna ,mai departe o exceptie ClassNotFoundException si obiectul de clasa T este initializat in urmatorul mod:

```
this.specificGenericClass = (Class<T>) ((ParameterizedType)
getClass().getGenericSuperclass()).getActualTypeArguments()[0];
```

O adnotatie de tipul @SuppressWornings(“unckecked”) este necesara pentru a specifica faptul ca o sa se transmita un obiect corespunzator echivalent lui T. Secventa de cod preia tipul clasei care il va inlocui pe T. Clasa contine metodedele:

- Insert(T t, String insertStatementString). Se creaza o conexiune la baza de date prin folosirea unui obiect care apeleaza metoda getConnection(), specifica clasei ConnectionFactory. Sunt create obiecte de tipul PreparedStatement si PropertyDescriptor care vor vor fi de folos pentru ececutia secventei de cod sql si pentru, obtinerea in mod generic a metodelor din clasa T. Pentru fiecare field din clasa tipului specificGenericClass (ex: id, nume, cantitate) se vor prelua getterele prin intermediul metodei getReadMethod si se vor invoca prin invoke(t,null) la nevoia de a fi inserate in locul “?” din statement-ul sql de executie. Se vor seta Stringurile si int in functie de tipul fiecarui field in parte. Dupa captarea exceptilor de tipul SQLException conexiunea la baza de date se va inchide prin chemarea metodelor de close. Ca integer de returnat se va transmite -1 daca nu s-a executat in intregime insert-ul si unu in caz de succes;
- findById va returna un obiect de tipul clasei T. Se creaza o noua instanta a obiectului specificGenericClass, se va face conexiunea la baza de date, se va insera in locul caracterului ’?’ din instructiune id-ul pe care il dorim si dupa executare se va crea un obiect resultSet unde va fi plasat rezultatul interogarii. Acest obiect va fi transformat in obiectul creat initial si va fi returnat, nu inainte de a fi inchisa conexiunea la baza de date;
- delete(String, int) este responsabila pentu executarea unui instructiuni de stergere a unui rand din baza de date, cel care are ca cheie primara (id) un numar specificat. Se deschide o conexiune la baza de date se insereaza id-ul

in statement si se executa prin executUpdate stergerea efectiva. Metoda va returna 0 in caz de esec si id-ul sters altfel;

- update(T, String) creaza o conexiunde la baza de date, pentru fiecare field preia metoda de citire (get) si o invoca pentru a completa campurile restante in statement. Se executata update-ul si se va returna 1 pentru succes si -1 in caz contrar. Conexiune este inchisa.

In cazul in care datele nu sunt de tipul celor inscrise in tabele, daca apare dublicat id-ul sau in cazul oricarei exceptii SQL se va scrie in obiectul logger ca si warning un mesaj.

CustomerDAO

Clasa contine stringurile ce trebuie executate ca insert, select, delete si update. Extinzand clasa AbstractDAO<Customer> va asigura ca un obiect de tipul Customer va fi trimis in locul parametrului generic T. Este creat un constructor care doar arunca o exceptie ClassNotFoundException si cele patru metode ce trebuie executate pe Customer, fiecare apeland metodele deja explicate din calsa pe care o extinde.

ProductDAO

Clasa AbstractDAO<Product> este extinsa, sunt create cele patru statementuri ce trebuie chemate pe tabela product, un constructor ce apeleaza super() si arunca o exeptie este creat si in mod similar metodele de insert, findById, delete si update apelaza metodele aferente din parinte.

OrderDAO

Clasa se comporta in mod similar cu celelalte clase din acest pachet cu singura exceptie ca de mai sus metoda insert este inlocuita cu place pentru plasarea unei comenzi (care din clasa parinte apeleaza tot insert).

BLL

Validator CustomerBLL | ProductBLL | OrderBLL

Validator | EmailValidator | PhoneValidator |

QuantityValidator

Subpachetul Validator are clase cu rolul de a valida (introduce niste constrangeri in baza de date) a unor campuri ce trebuie sa indeplineasca un tipar. Interfata Validator are o singura metoda, cea de validate, care v-a fi implementata de celelalte clase din acest subpachet. EmailValidator foloseste un pattern si un string regex care impreuna au rolul de a verifica daca email-ul unui cunsumer este valid. Metoda validate arunca o exceptie IllegalArgumentException in cazul in care email-ul este gresit. QuantityValidator verifica in metoda validate ca cantitate sa fie mereu pozitiva, in caz de nerespectare a cerintei o exceptie este aruncata. Ultima clasa, PhoneValidator verifica ca campul phone sa corespunda unui numar valid din Romania. Numarul trebuie sa aiba 9 cifre (0 este subinteles la introducere) si sa inceapa cu cifra 7.

CustomerBLL

Clasa extinde GenericData<Customer> si creeind o lista validators verifica in constructor campurile phone si email. Un obiect de tipul CustomerDAO este creat si folosit in metodele de findById, insertCustomer, deleteCustomer si updateCustomer. Aceste metode au ca parametru formal fie un id fie un Customer si apeleaza metodele din DAO pentru asi indeplini atriburile.

ProductBLL

Se extinde GenericData<Product> (clasa din pachetul presentation) se introduce intr-o lista un obiect de clasa QuantityValidator (in constructor). Metode de findById , insertProduct, deleteProduct si updateProduct sunt create si pe un obiect de tipul productDAO sunt apelate metode specifice de prelucrare/ interodare a bazei de date.

OrderBLL

Clasa este simiilara din punct de vedere structural cu celelate clase din pachetul BLL. Ea apeleaza pe un obiect OrderDAO metode de selectie, inserare, updatere si plasarea unui comenzi.

Presentation

View | GenericData | Controller

View

Clasa extinde JFrame si incapsuleaza tot ce o sa apara pe fereastra grafica. Aceasta parte, View, reprezinta a doua parte a modelullui MVC si este cea care da un aspect frumos (interactiune) cu utilizatorul (administratorul). Ca si panouri clasa are 7 panouri: partu JPanel si trei JScrollPane, pe langa acestea, global, clasa are : trei tabele JTable, opt butoane, trei modele pentru tabele doua tabele, unul de JtextField –uri, unul de JLabel-uri, trei siruri care tin string-uri ce reprezinta capul fiecarui tabel in parte si un integer ce tine starea panourilor ce se succed (CardLayout object). Clasa descrie 27 de metode si un constructor, toate cu rolul de a creea o aplicatie de adnimistrare a unui baze de data (warehouse). Aceste metode sunt:

- initButtons() – initializeaza toate cele opt butoane si da fiecaruia nume sugestive a atributiei pe care trebuie sa o indeplineasca;
- initPanelsContent() - initializeaza panourile tabelele si modelele si seteaza modelul fiecarui tabel obiectul de tipul DefaultTableModel aferent. Fiecare tabel descrie functia isCellEditable care returneaza false, spunand astfel ca datele nu pot fi introduse direct din tabele;
- initDataAndInfo() - creaza obiecte de tipul JLable si JtextField pentru fiecare elemet al tabloului (sase la numar) si da un nume fiecarui JLabel pentru a indica ce anume trebuie introdus in JTextField-ul cu acelasi numar.
- Toate aceste metode sunt apelate in init(), iar aceasta mai apoi in constructorul fara parametri (View()) care pe langa metoda de initializare mai apeleaza metoda setFrame;
- setFrame() – metoda seteaza caracteristicile principale ale ferestrei: dimensiune, layout, vizibilitate, titlu etc. Sunt adaugate trei panouri (cel winPanel contine prin CardLayout inca patru panouri ce vor fi adaugate explicit acestuia) si sunt apelate niste metode ce vor fi descrise in continuare;

- constructPanels() – seteaza pentru panoul de operati si cel de meniu layout-ul la null, pentru fiecare culoarea Color.WHITE si le seteaza granitele relative la dimensiunea fereastrei;
- linkPanels() – seteaza layout-ul panoului de contine cele trei tabele la un obiect de tipul CardLayout, seteaza dimensiunea tabelelor si le da o constrangere de nume fiecaruia dintre tabelele componente;
- addButtons() – seteaza limite in plan, pe fereastra pentru toate butoanele si le adauga panoului ce trebuie sa le contina menuPanel / operationPanel;
- addDataAndInfo () – seteaza fiecarui container de text si fiecarei etichete pozitia si le adauga in panoul de opereatii;
- tabels() – seteaza inaltimea fiecarei coloane din tabel la 25 si seteaza tabelelor culoarea de background la white;
- Metodele setPermissions(), permissionLevelOne() si permissionLevelTwo seteaza containarele care se pot modifica (edita) si care nu. Nivelul unu de permisiune va fi folosit pentru tabelele Customer si Order, nivelul doi pe tabela Order si prima metoda va fi apelata in constructor pentru un panou initial blank;
- Sase metode de get sunt descrise pentru a prelua anumite date private(ex: stare, modele, date)
- Fiecare buton are cate o metoda corespunzatoare care primese un ActionListener si il atribuie butonului. Acesea vor fi folosite in urrmartoarea parte, cea de conrol.

GenericData

Este o clasa ce primeste un parametttru generic T care in pachetul BLL ese inlocuit pe rand cu Customer,Product si Order. In constructor este preluat un obiect de tipul clasei T, genericType si este preluata clasa tipul de la convertire. Din clasa fac parte doua metode : insertToTable si deleteFromTable. Prima metoda, preia field-urile obiectului T si invoca metoda aferenta de get pentru a obtine valori ce vor fi adaugate intr-un tablou de obiecte si prin metoda addRow vor fi adugate in tabela. Cealalta

metoda deleteFromTable cauta cu un for (pana la Integer.MAX_VALUE) daca identificatorul coloanei coincide cu parametrul formal primit. Daca id este gasit se va apela removeRow() care va sterge linia din tabela;

Controller

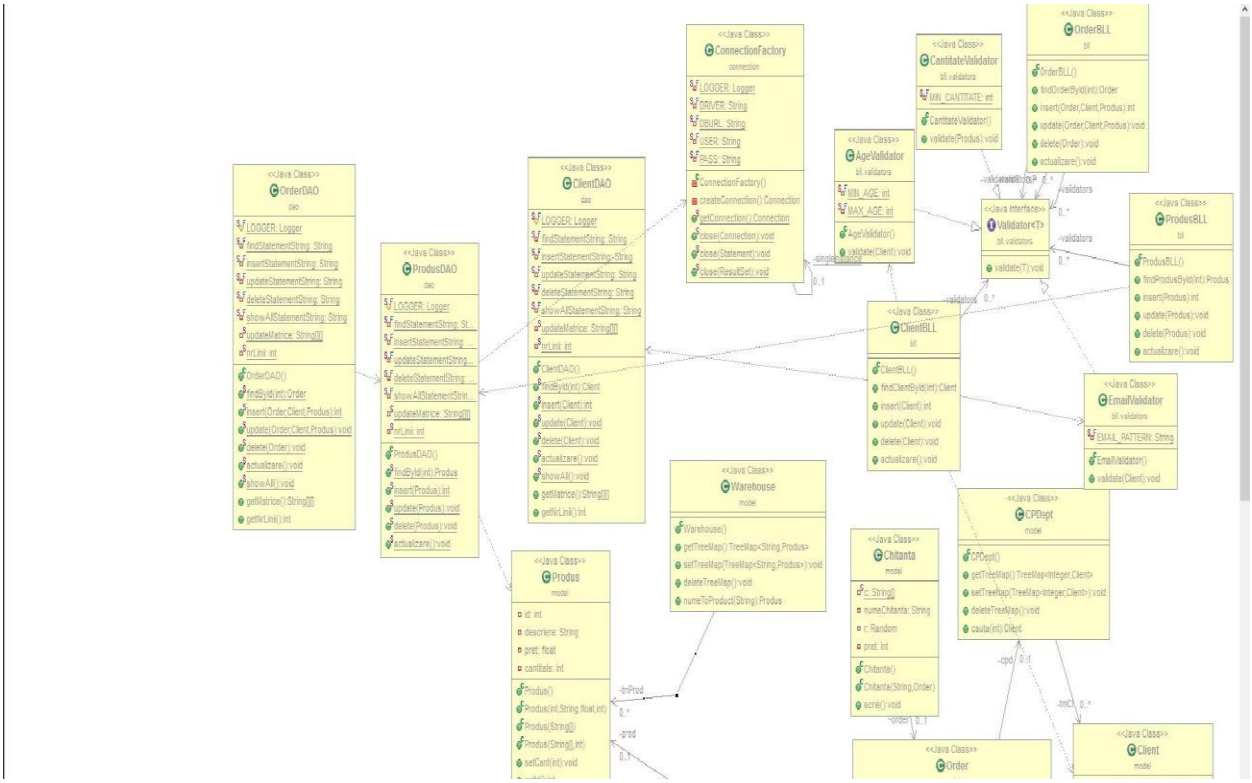
Clasa controller uneste View cu partea de model, astfel ca incapsuleaza fereastra grafica si trei obiecte BLL de tipul tabelelor din baza de date. Constructorul acestei clase initializeaza aceste obiecte la parametri formali si prin metoda de attach din partea de view, fiecare obiect primeset o noua sarcina, de tipul unor clase ce vor fi descrise ca inner class in inteiorul clasei. Pentru butoanele de sus, panoul de meniu, la o actiunde (ActionListener) schimba starea panoului ce contine tabelele selectand astfel cel dorit prin setShowPanel(). Pentru customer se selecteaza 1, pentru product doi si pentru order 3. Tot aici este setat si permisiunea “containers enable” pentru fiecare. Butonul de insert contine un switch pt a sti in ce tabela sa insereze (Customer/ Product) si depinzand de starea panourilor se creaza un nou obiect care primeste drept parametri valorile trimise in containarele pentru text. Se introduce in tabela din MySQL W (locala) si in cazul in care nu s-a intampinat nici o eroare de sintaxa tabelul (din fereastra grafica) este updatata cu datele introdusa, prin metoda insertToTable. Butonul de delete, pentru fiecare caz (tabela) in parte, preia id-ul si apeleaza metodele delete<T> si deleteFromTable pe obiectul global (model) declarat.

Update face aproximativ acelasi operatii: verifica starea panoului, trateaza fiecare caz in parte, creaza un obiect <T> care primeste ca si parametri datele din containerele text si in cazul in care update-ul se face in baza de date fara probleme, noile valori sunt afisate si in tabel (delete + insert). Listenarul atribuit butonului de order creaza un obiet nou si il adauga in baza ded date si in tabelele JTable.

Start

Clasa instantiaza in metoda de main obiecte de tipul model (bll),

fereastra grafica si o noua instanta a unui obiect Controller care primeste ca si parametri obiectele create.



Singeorzan Florin
Gr: 30227