Tehnici de programare

Documentație pentru tema 3

Miklós Balázs

Grupa 30224

UTCN, Calculatoare

Order Management

1. Obiectivul temei

Programul este folositor pentru a exemplifica o conexiune simpla intre limbajul de programare Java si sistemul de gestionare a bazelor de date MySql. Este folosita o baza de date relationala pentru a stoca datele importante, inerfata grafica de utiizator ajuta ca operatiile bazei de date sa fie mai usor facuta de catre utilizator(adunare, stergere, editare).

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

In prima vedere a problemei m-am dat seama ca in tema aceasta trebuie facut, legat o conexiune intre Java si o baza de date relationara (in cazul nostru MySql). O baza de date reprezinta o modalitate de stocare a datelor, un dispozitiv de stocare a datelor, unde datele pot fi usor regasite, modificate sau inserate. In baza de date relationale inseram datele in tabele, numele tabelelor reprezita un fel de trasatura comuna a datelor. Intre datele pot exista diferite relatii, legaturi pe care pot fi usor de realizate sau modificate.

Ca si cum cerinta spune, aici trebuie sa folosim minim 4 pacheturi pentru a realiza aplicatia, un pachet ModelClasses care reprezinta modelele de date in aplicatia, un pachet BusinessLogicClasses care reprezinta logica aplicatiei, PresentationClasses care reprezzinta clasele pentru interfata grafica de utilizator (GUI) si in final DataAccessClasses care contine un acces la baza de date. In cazul meu pachetul businessLogicClasses contine clasele pentru a valida un client, un produs sau o comanda. Pachetul dataAccessClasses contine clasa DBConect care face coenxiunea cu baza de date, si mai contine 3 clase pentru a accesa separat tabelul Client, Product sau Order din baza de date.

Tabelele sunt construine un felul urmator. Fiecare client are un ID, un nume , o varsta, un email si un numar de telefon. Tabelul Product are camparile ID, nume si Quantite specifica cate bucati mai exista din product in depozit si pret. Mai este o tabela numit Order care au urmatoarele campuri: Nr, client_id, client_name, product_id, product_name si quantitie. Tabelul aceasta de fapt face o legatura intre tabelele Client si Product. Daca vrem sa descriem in enunt pentru ce e buna tabelul aceasta putem sa zicem asa: tabelul contine date despre un client valid care a comandat un product valid. Un client este valid daca exista in tabelul Cient. Un product este valid atunci cand exista in tabelul Product, comandarea, orderea este valida daca clientul si productul este valid si cand au fost destul de producte in depozit (in stock).

Operatiile facute asupra tabelelor bazei de date sunt inserarea, editarea, stergerea. Fiecare operatie este facuta in modul urmator: Iau un stirng cu sintaxa scrisa in mysql doar ca aici folosesc metodele a limbajului Java astel incat stringul care este de fapt instructiunea, sa aiba o sintaxa ca si asi fi scris o insturctiune in MySql. Dupa de folosesc functiile predefinite statement.executeQuerry() sau statement.executeUpdate(). Primul folosesc in cazul in care vreau sa accesez datele la un tabel si al doilea cand vreau sa modific sau sa fac un update la un tabel in baza de date.

3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

Programul contile 4 pachete: dataAccessClasses, businessLogicClasses, modelClasses si presentationClasses. Fiecare pachet are propria sa rol in realizarea aplicatiei. Primul, dataAccessClasses face conexiunea cu baza de date, si ofera niste functii pentru a accesa tabelele in baza de date si pentru a lua datele stochate de tabele. Pachetul businessLogicClasses descrie niste operatii efectuate asupra tabelelor in baza de date, cum ar fi inserarea, stergerea sau editarea tabelelor, mai ofera acces la tabele la BD. Pachetul modelClasses contine clasele pentru cele 3 tabele din baza de date (Client, Product, Order). In plus pachetul aceasta mai contine si clasa cu metoda main. In final pachetul presentationClasses contine clasa pentru interfata grafca de utilizator (Window).

La crearea diagramei de UML am folosit modalitatile oferite de Microsoft Word, mai ales functia insertShapes si addText pentru a adauga text in dreptungiuri. Intre clasele diagramei poate sa apara relatie de tip agregare si relatia de tip „foloseste”.

Interfata grafica de utilizator contine urmatoarele elemente. Prima data am trei butoane numit Show Client Table, Show Product Table si Show Order Table pe care sunt facute pentru a arata tabelele actualizate dupa anumite modificare sau fara modificari. Deci de fapt dupa orice opratie se face click la butoanele acestea si tabelele vor fi actualizate.

Interfata mai contine niste fielduri de texte JtextFields, in aceste texte inseram datele pentru a face un update la un tabel. Butoanele AddClient EditClient si DeleteClient adauge, modifice sau sterg un cleint dupa datele inserate. Butoanele AddProduct, EditProduct sau DeleteProduct sunt facute pentru a adauga, sterge sau a edita un product. Mai este facuta o optiune createOrder unde daca dam click, dupa Idul clientului ce am introdui si dupa Id-ul produsei ce am introdus si dupa cantitate, face o comanda, dar doar daca clientul si produsul introdus a fost valid si daca produsil mai este in stoc.

4. Implementare

<u>Pachetul busenessLogicClasses</u>

Clasa **ClientValidation** contine o metoda statica:

-public static boolean valid(Client c)- returneaza true daca clientul pe care vrem sa inseram e valid, in caz contrar false

Clasa **ProductValidation** contine o metoda statica:

-public static boolean valid(Product p)- returneaza true daca produsul pe care vrem sa inseram e valid, in caz contrar false

Clasa **OrderValidation** contine o metoda statica:

-public static boolean valid(Order o)- returneaza true daca comanda pe care vrem sa facem e valid adica exista in tabelul client clientul la care vrem sa facem comanda si mai exista produsul in tabelul produs si mai este in stoc din el.

Clasa **Reflection** are o metoda stacia:

-public static JTable createTableWithReflection(ArrayList<Object> objects) returneaza un Jtable din obiectele primite. In cazul meu merge pentru obiecte de tip Client si Product.

Pachetul dataAccesClasses

Clasa **DBConnect** are variabilele de instanta Connection con, Statement st, ResultSet rs.

Constructorul public DBConnect() face conexiunea la serverul local la baza de date unde tin tabelul Cliennt, Product si Order. Clasa mai are niste gettere si settere.

Clasa **ClientAcces** are urmatoarele functii statice:

-public static ArrayList<Client> getClientTableData()- returneaza lista clientilor din tabelul DB

-public static void insertClient(Client client)- returneaza void, insereaza un client in tabelul DB daca au fost valide datele clientului.

-public static void deleteClient(int clientID)- reuturneaza void, sterge un client dupa ID

-public static void editClient(int clientID, String newName, int newVarsta, String newEmail, String newPhoneNumber)- returneaza void, modifica un client dupa ID.

Clasa **ProductAcces** are urmatoarele functii statice:

-public static ArrayList<Product> getProductTableData()- returneaza lista produselor din tabelul DB

-public static void insertProduct(Product product)- returneaza void, insereaza un client in tabelul DB daca au fost valide datele produsului.

-public static void deleteProduct(int productID)- reuturneaza void, sterge un produs dupa ID

-public static void editProdus (int productID, String newName, int newQuantity, int newPret)- returneaza void, modifica un produs dupa ID.

Clasa **OrderAcces** are urmatoarele functii statice:

-public static ArrayList<Order> getOrderTableData()- returneaza lista comenzilor din tabelul DB

-public static void insertOrder(Order o)- returneaza void, creaza o comanda daca clientul si produsul exista si daca din produs mai este in stoc

-public static void deleteOrder(int orderID)- reuturneaza void, sterge un order dupa ID

-public static void writeBill()- returneaza void, face un nou fisier text in care scrie comenzile facute

Pachetul modelClasses

Clasa App este clasa care are metoda main, aici apelez new Window() pentru ca interfata de grafica sa apare.

Clasa Client are variabilele de instanta ID, nume, varsta, email, PhoneNumber, si mai are niste constructori public Client() pentru a putea crea Clientul in mai multe feluri. In plus clasa aceasta mai are niste gettere si settere pentru variabilele de instanta.

Clasa Product are variabilele de instanta ID, Name, quantite, pret si mai are niste constructori public Product() pentru a putea crea Produsul in mai multe feluri. In plus clasa aceasta mai are niste gettere si settere pentru variabilele de instanta.

Clasa Order are variabilele de instanta ID, client_id, client_name, product_id, product_name, quantitie si mai are niste constructori public Order() pentru a putea crea Produsul in mai multe feluri. In plus clasa aceasta mai are niste gettere si settere pentru variabilele de instanta.

Pachetul presentationClasses

Clasa Window este singura clasa a pachetului, clasa cea mai complexa care de fapt leaga toata programul. Are urmatoarele variablie de instanta: DBConect db, mai are trei butoane pentru vederea celor trei tabele din baza de date, am trei Jtable pentru cele trei tabele, si mai am si un JscrollPane sp. Mai sunt trei ArrayListuri statice pentru a putea accea Clientii, Produse si Ordere in DB. In plus mai am niste butoane pentru modificarea, adaugarea sau stergerea clientilor respectiv mai am niste JtextFielduri unde trebuie introduse datele, si dupa datele introduse daca facem click la butoane, modificarile o sa se intample. Mai am doua paneluri principale(JPanel) si 9 panele secundare. Aceste sunt facute pentru aranjarea chestilor in interfata grafcia.

Clasa aceasta extinda Jframe, la constructorul public Jframe() creez un nou fereastra si incep sa agaug chestile (butoanele si textFieldurile) in el respectiv si panelele. Metodele clasei:

-public void addShowClientsListener()- returneaza void, adauga un ActionListener pentru butonul Show Client Table.

- public void addShowProductsListener()- returneaza void, adauga un ActionListener pentru butonul Show Product Table.

- public void addShowOrdersListener()-returneaza void, adauga un ActionListener pentru butonul Show Order Table.

- public void addAddClientListener()- returneaza void, adauga un ActionListener pentru butonul Add Cleint (adaugare client).

- public void addRemoveClientListener()- returneaza void, adauga un ActionListener pentru butonul Delete Cleint (stergere client).

- public void addEditClientListener()- returneaza void, adauga un ActionListener pentru butonul Edit Cleint (editare client).

- public void addAddProductListener()- returneaza void, adauga un ActionListener pentru butonul Add Product (adaugare produs).

-public void addDeleteProductListener()- returneaza void, adauga un ActionListener pentru butonul Delte Product (stergere produs).

- public void addEditProductListener()-returneaza void, adauga un ActionListener pentru butonul Edit Product (editare produs).

- public void addMakeOrderListener()- returneaza void, adauga un ActionListener pentru butonul Make Order (creaza o comanda).

- public void addRemoveOrderListener()- returneaza void, adauga un ActionListener pentru butonul Delete Order (sterge o comanda).

-public JTable makeClientJTable()- returneaza un nou Jtable facut dupa tabelul Client din DB.

-public JTable makeProductJTable()- returneaza un nou Jtable facut dupa tabelul Produs din DB.

-public JTable makeOrderJTable()- returneaza un nou Jtable facut dupa tabelul Order din DB.

5. Rezultate

Programul cel mai usor poate fi testat folosind interfata grafica de utiliator (GUI). Acolo apar niste butoane, pe fiecare butoan scrie functia lui, inaintea butoanelor de modificare a tabelelor din baza de data sunt niste textfield-uri in care utilizatorul trebuie sa introduca datele necesare pentru a face modificarile. Modificarile pot fi inserare, editare si stergere. In tabelele client si produs pot fi utilizate aceste modificari.

In tabelul order putem adauga o comanda dupa id-ul clientului si id-ul produsului. Produsul va fi adaugat numai in cazul in care clientul si produsul este valid (exsta in tabele client si produs) si din produs mai este in depozit.

Mai este implementata o metoda care primeste o lista de obiecte (in cazul meu Client si Product) si face un tabel din obiecte primite.

Testarea de aceeapoate fi facut foarte usor, fiindca sunt implementate trei butoane de actualizarea a tabeleor din baza de date cu care putem sa vedem cum s-au modificat tabelele dupa orice update sau delete. Operatiile sunt facute corecte, aceasta afirmatie este demostrata si prin faptul ca in baza de date in interiorul programului daca verific continutul unui tabel atunci si acolo asa apar datele modificate cum trebuie.

Mai este implementata o functie care primeste o lista de obiecte si cu metoda reflectiei face un nou Jtable analizand datele si trazaturile obiectelor date ca parametru. Valoarea returnata este unJTable, Jtableul returnat putem sa vedem si la interfata grafica.

6. Concluzii:

Aplicatiile de genu ceea ce este cerut in tema aceasta putem sa primim la orice firma care cere sa conectam o baza de date la un program Java sau daca se cer sa facem un GUI pentru o baza de date relationala. De aceea am invatat aici cum trebuie sa leg Java la MySql si cum pot sa accesez datele la baza de date, cum pot sa modific datele.

7. Bibliografie:

-http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/HW3_Tema3/Tema3_HW3_Indications.pdf

-http://tutorials.jenkov.com/java-reflection/index.html

8. Diagrama UML