TEHNICI FUNDAMENTALE DE PROGRAMARE

Manager de comenzi

*-JavaSE15, Java Swing-*

Realizat de:

Tanul Gabriel-Ștefan

an 2 grupa 30222

1. Problemă și soluție
2. Obiective
3. Analiză
   1. Pachete
      1. Model
      2. Business logic
      3. Data access
      4. Presentation Layer
      5. Date choosers
4. Dezvoltare grafică
5. Interogări MySQL
6. Posibilități de dezvoltare ulterioară

1. Problemă și soluție

Consider an application OrderManagement for processing clientorders for a warehouse. Relational databases are used to store the products, the clients and the orders

Se considera o aplicație de distribuire și furnizare a unor comenzi pentru un depozit. Se cere utilizarea bazelor de date relationale pentru stocarea datelor legate de **produse,** **clienti și comenzi.**

Este costisitor din punct de vedere al timpului să ținem evidența multor date pe foaie, cum se obișnuia acum mulți ani când nu aveam la dispoziție dispozitive de calcul și sisteme hardware avansate ca în prezent, iar o soluție bună la această problemă ar fi implementarea unei aplicații cu interfață grafică într-un limbaj de programare cunoscut care să folosească o bază de date unde se pot reține informațiile despre un anumit subiect sau în cazul de față despre comenzile unui depozit mult mai ușor. În cazul de față am folosit limbajul de programare Java și interfață grafică din Java Swing folosind mediul de dezvoltare **IntelliJ IDEA, iar baza de date utilizată a fost implementată în Navicat pe un server de MySQL.**

2. Obiective

Scopul principal al acestui proiect a fost de a dezvolta și implementa o aplicație unde cei desemnați să se ocupe cu gestiunea comenzilor pot introduce niste detalii ca text legați de datele unei **comenzi, clienți sau produse** disponibile iar apoi prin simpla apăsare a unui buton, aceste date se vor stoca pe server într-o bază de date iar accesul acestora, nu doar că va fi mult mai ușor, dar și mai securizat.

**Cum să implementez soluția?**



**Utilizare**: modificare date, adăugare date, ștergere date

**Principalul actor: utilizatorul**

**Scenariul cel mai probabil:**

1. Utilizatorul introduce de la tastatură parametrii corespunzători simulării sub formă de text.
2. Actorul principal confirmă introducerea datelor.
3. Calculatorul verifică corectitudinea datelor și le adaugă în baza de date.

**Scenariul mai puțin probabil:** Date introduse incorect

* Utilizatorul introduce datele greșit (format incorect, litere, caractere speciale etc.)

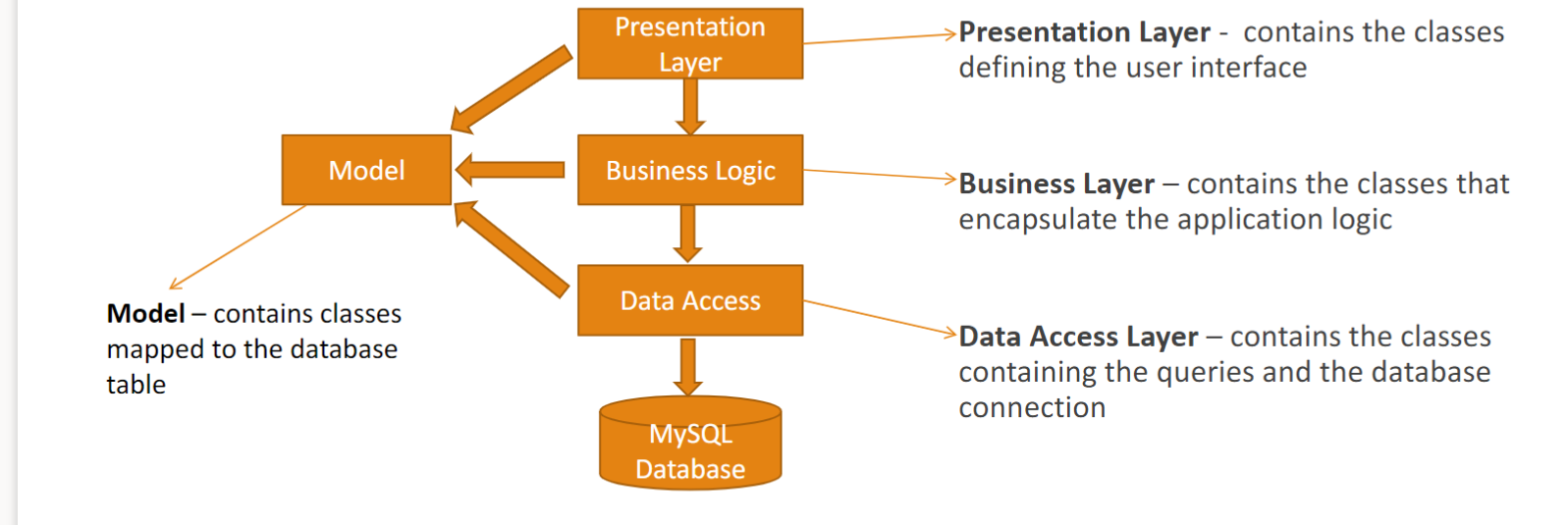
Atenție!!! Formatul parametrilor trebuie sa fie numeric în cazul atributelor de genul( **cantitatea unui produs, id-ul unui produs).**

* În acest caz, se revine la pasul 1.

1. Analiză

Pentru a reliza o conexiune în timp real cu o bază de date este nevoie de un conector JDBC care să permită transmiterea datelor înspre server folosindu-se de user, parola și url-ul serverului.

De asemenea am ales o structurare a pachetelor din java după cum urmează.

A a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a

**și Table.**

În **Model** am ales ca variabile instanță clasele implementate pentru frame, url, e-mail și parolă de la server și alte frame-uri legate de erori.

**ClockPane** este un panel care afișează în timp real data în colțul frame-urilor.

**Table** este o clasa implementată de mine care conține mai multe funcționalități ale JTable-ului.

**În Business logic** am adăugat toate clasele legate de **ExtragereDate, ExtragereAntet, BusinessLogic, ExecProc, ExecHeader și Main.** Sunt toate clasele responsabile de controlul datelor și informațiilor care intră și ies din sistemul aplicației respectiv din interfața grafică. **ExtragereDate** este o clasa ce conține metode care extrag datele dintr-un anumit tabel transmis ca parametru funcției, funcție care folosește biblioteca java.SQL și care prin intermediul conectorului JDBC comunică cu baza de date MySQL și preia informațiile dorite. De asemenea **ExtragereAntet** este bazată pe același principiu însă aici este returnat antetul unui tabel pentru folosirea acestuia în interfața grafică. **ExecProc și ExecHeader** execută o anumită interogare MySQL și în funcție de **result set-ul** acesteia se transmite la un ***String***care se poate folosi mai departe în aplicație ( indiferent că este vorba de antetul unui **result set** sau datele acestuia).

Pachetul **Data access** am adăugat clasele **DBConnection și DBUpdate** care realizează conexiunea propriu-zisă a limbajului Java cu serverul bazei de date MySQL. În **DBConnection** se realizează prin **DriverManager** conexiunea cu serverul MySQL având URL-ul și parola serverului folosit iar separat este declarat numele bazei de date utilizate. **DBUpdate** are toate action-listener-ele butoanelor de introducere, editare și stergere a datelor din tabelele existente iar după cum spune și numele, aici se realizează actualizarea bazei de date și a tabelelor din interfața grafică. Se sterg, introduc sau actualizează datele prin intermediul **PreparedStatement-urilor sau CallableStatement-urilor** care accesează proceduri existente din baza de date sau execută un query dat ca parametru funcțiilor.

**PresentationLayer** este pachetul care conține toate clasele interfeței grafice **ClientsFrame, EroareInsert, EroareLogare, EroareStergere, MeniuPrincipal, OrdersFrame, ProductsFrame, UsersFrame.** Fiecare dintre acestea extinde clasa JFrame și creează câte o fereastră pentru fiecare cu o funcționalitate diferită și interfață diferită. În **ClientsFrame** se află tabelul cu datele despre clienți, **OrdersFrame** conține tabelul care realizează o nouă comandă dar și un tab cu detalii despre clienți în cazul în care se dorește informația unui client. **ProductsFrame** deține detaliile despre produsele depozitului așezate într-un tabel, detalii privind cantitatea, denumirea și id-ul produselor.

1. Dezvoltarea grafică

* SWING API
  1. Parte din clasele fundamentale java (JFC)
  2. Facilitează scrierea grafică a aplicațiilor java
  3. Include 17 pachete standard printre care cel mai folosit:
* **javax.swing** - Is the most important package from Swing

|  |  |
| --- | --- |
| **Tipul componentei** | **Examples** |
| Componente atomice | JLabel, JButton, JCheckBox, JRadioButton, JToggleButton, JScrollBar, JSlider |
| Componente complexe | JTable, JTree, JComboBox, JList, JFileChooser, JColorChooser, JOptionPane |
| Componente de tip text | JTextField, JPasswordField, JTextArea, JEditorPane, JTextPane |
| Meniuri | JMenuBar, JMenu, JPopupMenu, JMenuItem, CheckboxMenuItem , JRadioButtonMenuItem |
| Cointainere intermediare | JPanel, JTabbedPane, JDesktopPane |
| Containere principale | JFrame, JDialog |

Am folosit un container principal de tip JFrame pentru a ilustra fereastra principală a calculatorului iar apoi am creat un fel de ferestre de tip pop-up care apar în cazul excepțiilor și anume a erorilor prezentate mai sus.

Managerul de aspect este folosit pentru aranjarea într-o anumită structură a containerelor sau a componentelor.

a) BorderLayout – plasează componentele în cinci zone ale containerului principal: sus, jos stânga, dreapta și centru.

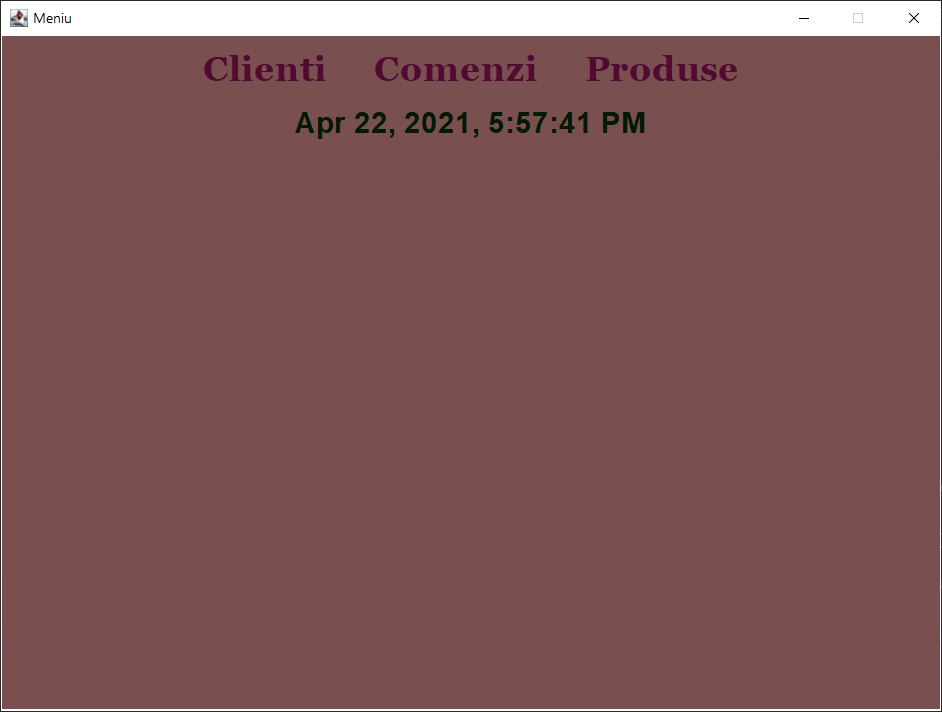
b) BoxLayout – plasează componentele pe o linie sau pe o coloană

c) FlowLayout – plasează componentele într-o singură linie dreaptă

d) GridLayout – plasează componentele într-un aspect matricial în mai multe celule

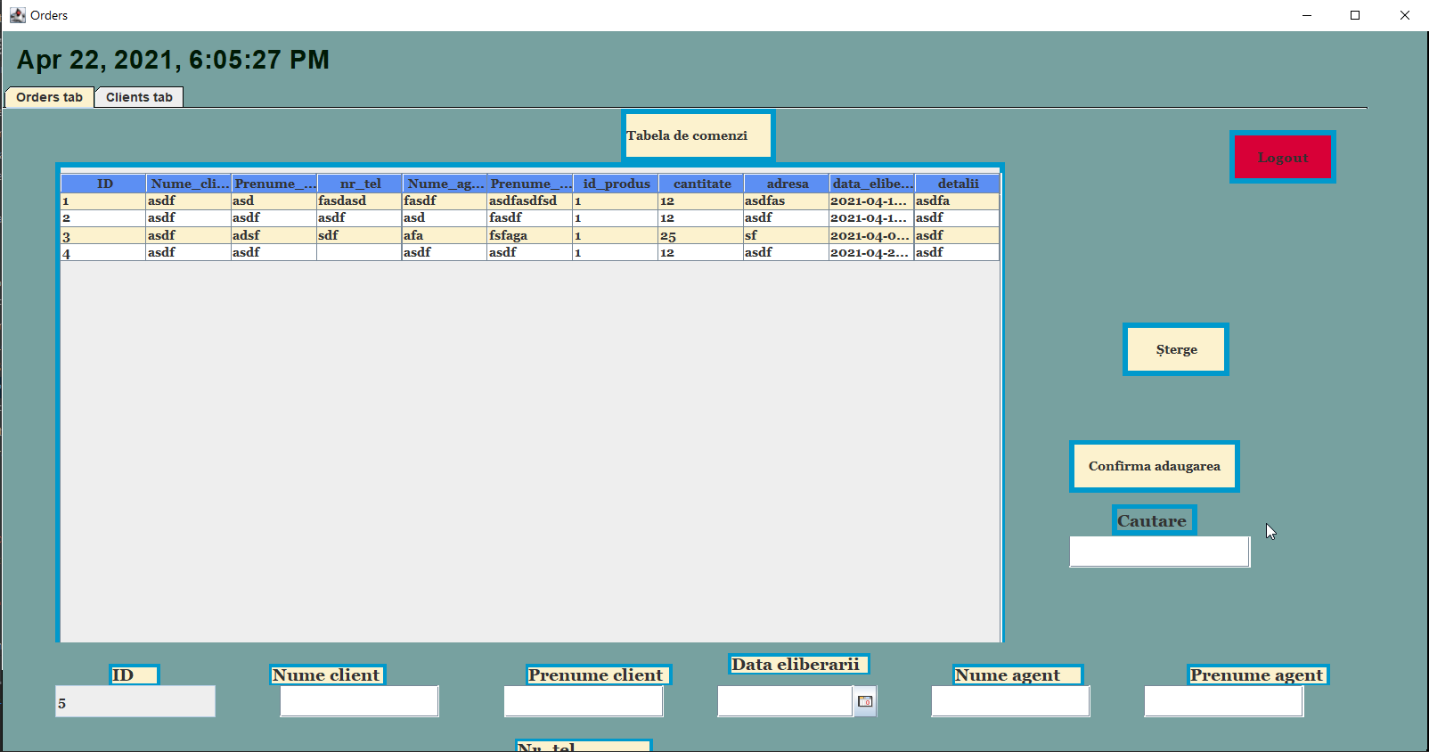
Am ales un GridLayout însă nu l-am folosit pe cel implicit ci am preferat să aleg eu coordonatele pentru componente deoarece a fost mai simplu să le așez exact în locul dorit.

Așa arată partea de grafică a calculatorului. Au fost implementate butoane pentru a alege operația dorită, pentru a confirma introducerea polinoamelor și etichete pentru a ilustra rezultatul sau restul polinomului final.



a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a

Butoane ferestre



Butoane de decizie

Date tabel

Text-field-uri date intrare

1. Interogări în MySQL

Pentru a modifica datele din server m-am folosit și de anumite proceduri stocate pe server care prin intermediul **CallableStatement-ului** din java se accesau procedurile și se manipulau datele din server ( stergere, actualizare și introducere). De exemplu, proceduri precum **insertOrder, deleteOrder** preiau datele transmise ca parametru și se actualizează în tabel.

1. Posibilități de dezvoltare ulterioară

Ulterior se pot alege anumite elemente grafice care sa usureze utilitatea acestei aplicatii. Se pot face triggere și anumite listene-re pentru a ușura alegerea datelor din text-field-uri dar și pentru a modifica automat datele din baza de date. Se poate crea și o pagina de login unde se autentifică utilizatorii, cei responsabili de distribuirea comenzilor și administratorul care poate modifica datele utilizatorilor. De asemenea se pot adauga tabele care să stocheze datele despre chitante, facturi, cheltuieli și profit în cazul depozitului.

Bibliografie

[1] https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

[2] https://docs.oracle.com/javase/8/docs/api/java/util/Timer.html

[3] http://www.tutorialspoint.com/java/util/timer\_schedule\_period.html

[4] http://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-andthreadpoolexecutor.html

[5] B. Goetz et al., Java Concurrency in Practice, Addison-Wesley Professional; 1 edition (May 19, 2006)

[6] https://toedter.com/jcalendar/