Java Project Documentation

**Name:** Huszar Istvan

**Group:** 1533e/1

**Selected Theme:** Databases

**Database management software:** Microsoft Access

**Tasks:**

1. Add ‘robot’ column in the database
2. Copy Button
3. Extend query option in they GUI
4. Search option between time stamps
5. Fix the ‘close’ buttons

The base of the project had a jTable in which was inserted the data from the database, INSERT function, DELETE function, UPDATE function and query based on ordering the entries by the ‘program’ column (ascending, descending or isolating one type).

**My contributions:**

1. Adding robot column

By adding another column, I needed to restructure most of the functionalities of the initial program. I modified the database and jTable so it would support one more column. I added an extra field to the **filltable()** method of **FrDatabase** class so the new database would be compatible with the jTable. In the GUI for INSERT and UPDATE I added a new text field and label for each, and connected them accordingly to run as before, but with one more extra field

1. Copy Button

The thought process of implementing this feature was to use the scrolling table in the GUI to select the instances I would want to copy, then using an insert query with the fields of the copied rows.

We iterate through all the selected rows and for each one we generate a string that would later be used with the **runSQL()** method.

Code:

int[] selected = jTable1.getSelectedRows();

for (int i = selected.length - 1; i >= 0; --i) {

String copy = "INSERT INTO Table1" +

"([robot],[program],[instructiune],[parametri],[data],[real]) VALUES("+

"'"+ jTable1.getModel().getValueAt(selected[i], 1).toString()+"',"+

"'"+ jTable1.getModel().getValueAt(selected[i], 2).toString()+"',"+

"'"+ jTable1.getModel().getValueAt(selected[i], 3).toString()+"',"+

"'"+ jTable1.getModel().getValueAt(selected[i], 4).toString()+"',"+

"'"+ jTable1.getModel().getValueAt(selected[i], 5).toString()+"',"+

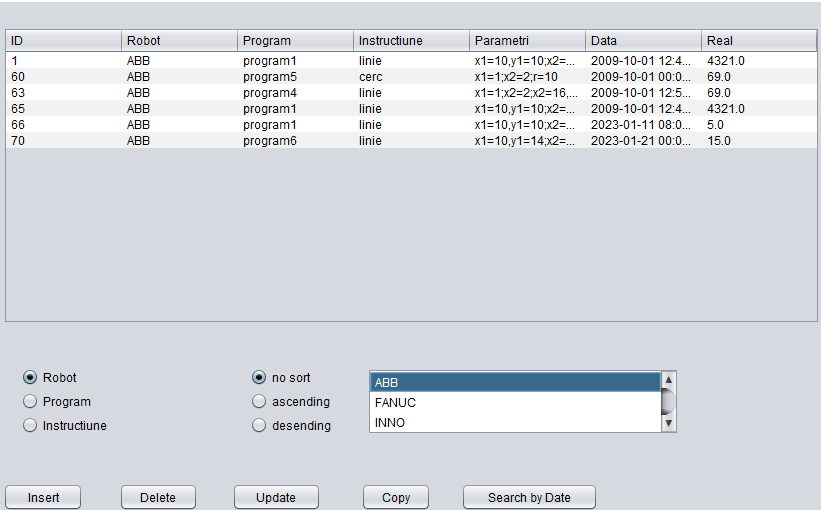
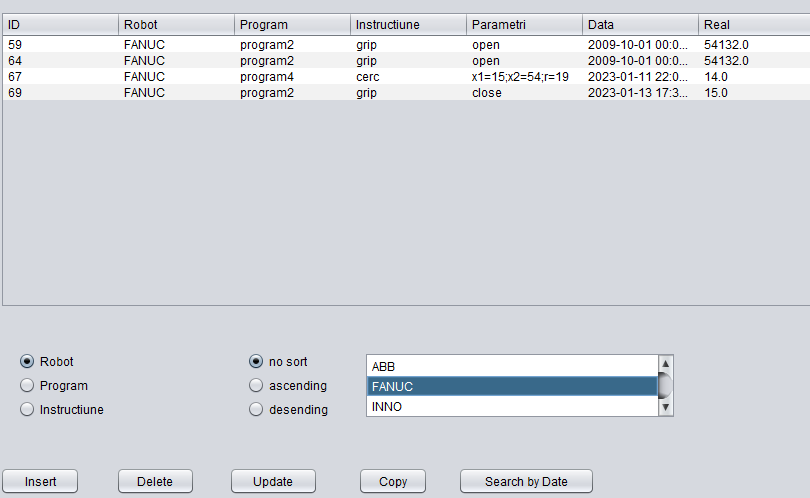
"'"+jTable1.getModel().getValueAt(selected[i], 6).toString()+"')";

runSQL(copy);

}

1. Extend query option in they GUI

What I wanted to achieve with this is to be able to more variable queries when talking about ordering the entries in the database. Because the initial code only ordered the program column I made a button group where the user would select witch column would want to order, and only then the scrolling pane would populate (with the help of the method **fillListBox()** ) with DISTINCT entries of that column, once selected it would work just like before, but now ordering based on one more parameter.



I initialized an empty string, which would change based on which radio button would be picked, and that string would be added to the general string witch will generate a query with the **fillTable()** method.

**Key code**:

String order = "";

private void jRadioButton4ActionPerformed(java.awt.event.ActionEvent evt){

order = "robot";

fillListBox("SELECT DISTINCT robot FROM Table1;");

}

private void jRadioButton6ActionPerformed(java.awt.event.ActionEvent evt) {

order = "instructiune";

fillListBox("SELECT DISTINCT instructiune FROM Table1;");

}

private void jRadioButton5ActionPerformed(java.awt.event.ActionEvent evt){

order = "program";

fillListBox("SELECT DISTINCT program FROM Table1;");

}

In the method which generates the query that fills the jTable I modified the output string so it wouldn’t contain the keyword program, instead concatenating multiple strings, one of them being the variable order.

private void jList1ValueChanged(javax.swing.event.ListSelectionEvent evt){

if (!evt.getValueIsAdjusting()) { //This line prevents double events

System.out.println(jList1.getSelectedValue().toString());

String sel = jList1.getSelectedValue().toString();

if (sel.equals("\*"))

fillTable("SELECT \* FROM Table1;");

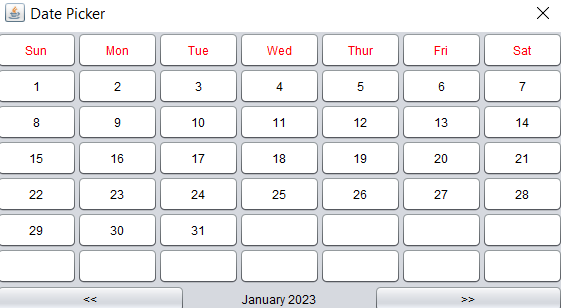
else

fillTable("SELECT \* FROM Table1 WHERE "+ order + " = '" + sel + "';");

}

1. Search options between time stamps

The easiest way to go about this was to find a library or API, that would generate a GUI from witch you could just select the date and return a string. So I found a class called **DatePicker.java** which I would use to instantiate two objects that filled two jText cassettes, i also added two more for the hours of each date.



Date Picker GUI

Graphical user interface

Description automatically generatedI created a new class because I wanted it to be a new popup menu. This I the simple design:

By pushing the add button it would execute a query to search for all the entries in the database between these 2 dates and hours.

Key code:

private void popup1(java.awt.event.ActionEvent evt) {

//just declared a new DatePicker, and because it’s parameter in the constructor is a jFrame I used the keyword **this** to refer to this jFrame

jTextField1.setText(new DatePicker(this).setPickedDate());

}

private void popup3(java.awt.event.ActionEvent evt) {

jTextField5.setText(new DatePicker(this).setPickedDate());

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

parent.fillTable("SELECT \* FROM Table1 WHERE data BETWEEN '" +

jTextField1.getText() +" "+jTextField2.getText()+"' AND '"+

jTextField5.getText()+" "+jTextField3.getText()+"'" );

}

The keyword **parent** is relevant because we couldn’t update the jTable from another jFrame except the one that it is one, but because the class **FrSearch** generates from **FrDataBase** we could use the parent method to fill the table. In order to be able to do that we needed to change the accessibility of the **fillTable()** method from **private** to **protected.**

To access the FrSearch class we instatiated an object to in FrDataBase and the make so it would popup when we pressed a button.

FrSearch sch;

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt)

sch = new FrSearch();

sch.setVisible(true);

sch.setThis(this);

}

1. Fix the close buttons

The problem with the close buttons was that it the statement that would close the jFrame was just simply missing, introducing it, an example for one of the close buttons would be, in the **FrInsert** class.

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt){

this.dispose();

}