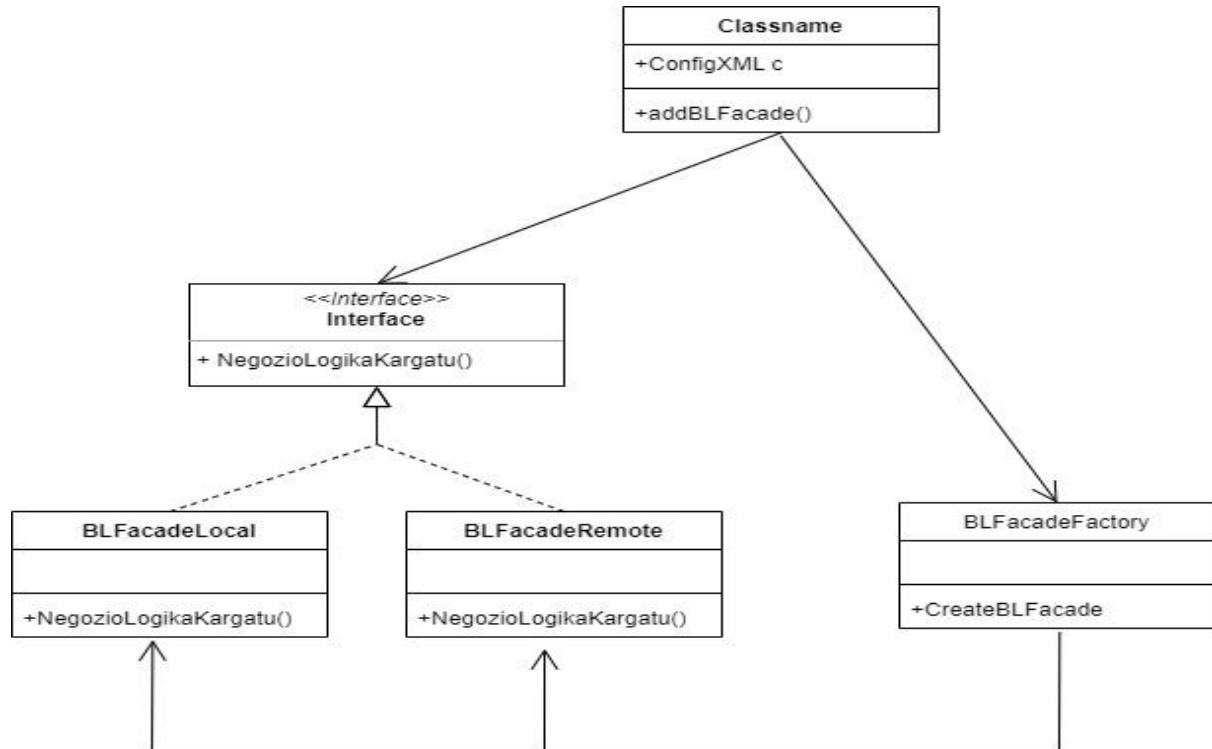


PATROIAK PROIEKTUA-BETS

SIMPLE FACTORY

Klase diagrama



Aldatutako Kodea

· 3 klase eta interfaze berri abt sortu ditut (BLFacadeLocal, BLFacadeRemote, BLFacadeFactory eta BLFacadeInterface) Hona hemen bere kodea:

BLFacadeFactory

```
package gui;

public class BLFacadeFactory {
    public static BLFacadeInterface createBLfacade( boolean isLocal) {
        if(isLocal) {return new BLFacadeLocal();}
        if(!isLocal) {return new BLFacadeRemote();}
        return null;
    }
}
```

BLFacadeRemote

```
package gui;

import java.net.MalformedURLException;
import java.net.URL;

import javax.xml.namespace.QName;
import javax.xml.ws.Service;

import businessLogic.BLFacade;
import configuration.ConfigXML;

public class BLFacadeRemote implements BLFacadeInterface {
    public void negozioLogikaKargatu() {
        try {
            ConfigXML c=ConfigXML.getInstance();
            BLFacade appFacadeInterface;
            String serviceName= "http://" + c.getBusinessLogicNode() + ":" +
c.getBusinessLogicPort() + "/ws/" + c.getBusinessLogicName() + "?wsdl";

            //URL url = new URL("http://localhost:9999/ws/ruralHouses?wsdl");
            URL url = new URL(serviceName);

            //1st argument refers to wsdl document above
            //2nd argument is service name, refer to wsdl document above
            // QName qname = new QName("http://businessLogic/",
```

BLFacadeLocal

```
package gui;

import businessLogic.BLFacade;
import businessLogic.BLFacadeImplementation;
import configuration.ConfigXML;
import dataAccess.DataAccess;

public class BLFacadeLocal implements BLFacadeInterface {

    public void negozioLogikaKargatu() {
        //In this option the DataAccess is created by FacadeImplementationWS
        //appFacadeInterface=new BLFacadeImplementation();

        //In this option, you can parameterize the DataAccess (e.g. a Mock DataAccess
object)
        ConfigXML c=ConfigXML.getInstance();
        BLFacade appFacadeInterface;
        DataAccess da= new DataAccess(c.getDataBaseOpenMode().equals("initialize"));
        appFacadeInterface=new BLFacadeImplementation(da);
    }
}
```

BLFacadeInterfaze

```
package gui;

public interface BLFacadeInterface {
    public void negozioLogikaKargatu();
}
```

Baita Application launcher- en zegoen kodea aldatu egin izan behar dut klase berri hauek implementatzeko

```
package gui;

import java.awt.Color;
import java.net.URL;
import java.util.Locale;

import javax.swing.UIManager;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;

import businessLogic.BLFacade;
import businessLogic.BLFacadeImplementation;
import configuration.ConfigXML;
import dataAccess.DataAccess;

public class ApplicationLauncher {

    public static void main(String[] args) {

        ConfigXML c=ConfigXML.getInstance();
        //isgsiu

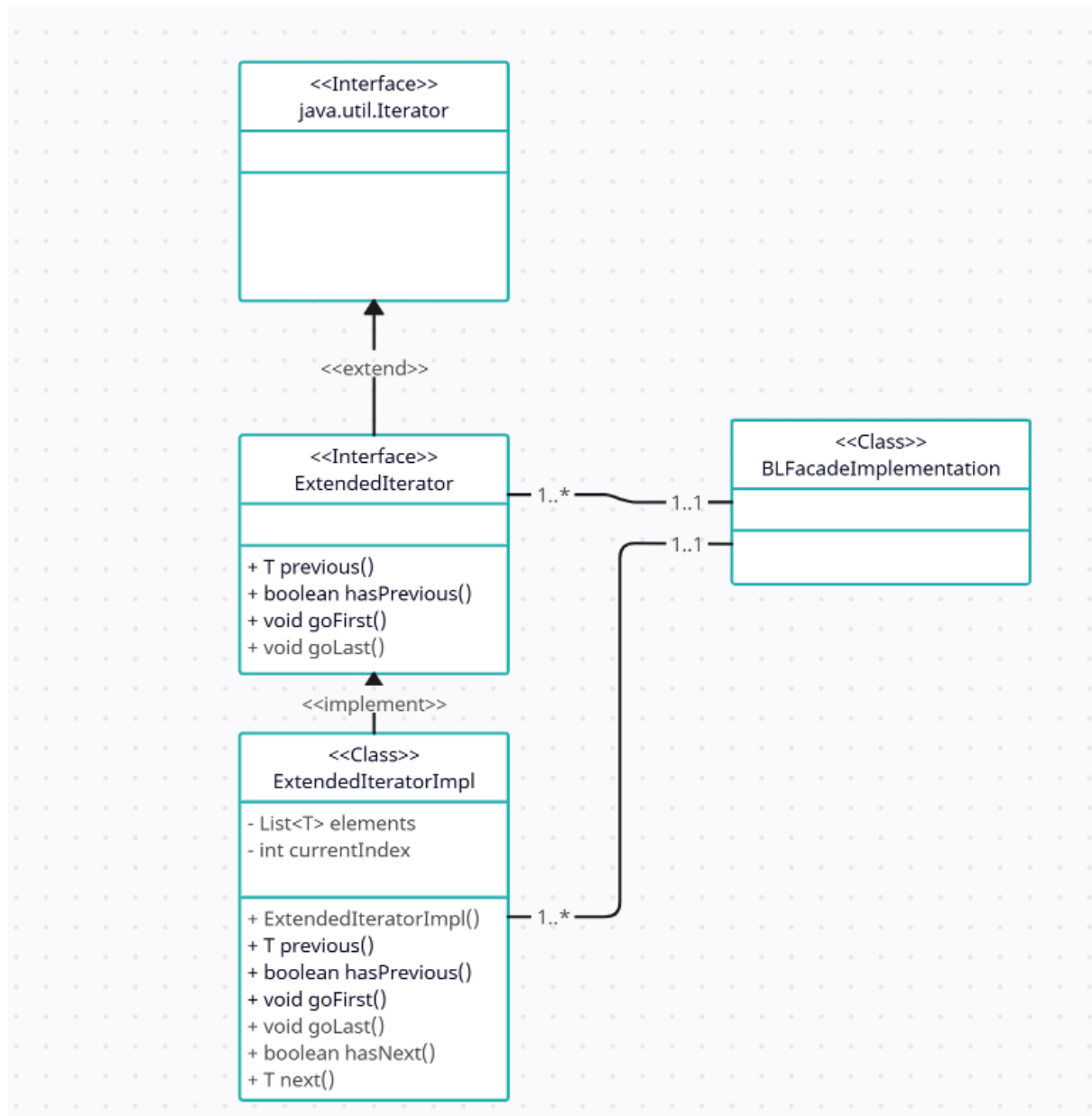
        System.out.println(c.getLocale());
    }
}
```

Egindako aldaketak hauek izan dira: BLFacade objektua sortzen zute kode lerroak BLFacadeRemote eta BLFacadeLocal-en idatzi ditut, NegoziologikaKargatu() metodoan, metodo hau interfazetik heredatzen dute. Eta objektu hauen instantziak sortzeko factory klasea sortu dut. Beraz gure bezeroak(ApplicationLauncher) factory klasea deituko du

Aldaketa haukein BLFacaderen kreazioa BLFacadeFactory klaseane giten dugu, modu honetan ApplicationLauncher klaseak ez du hainbeste erresponsabilitate izango eta etorkizun batean beste BLFacade mota bat sortu nahi badugu ez dugu ApplicationLauncher klasea aldatu behar.

ITERATOR

Klase diagrama



Kode aldaketak

'java.util.Iterator' klasetik heredatzen duen interfaze bat sortu dut, "ExtendedIterator" izenekoa. Interfaze hori "ExtendedIterator" klaseak inplementatzen du eta, hortaz, klase horrek 'java.util.Iterator' eta ExtendedIteratoreko metodoak inplementatu beharko ditu. Hortaz, hemen dago interfaze berriaren eta klase berriaren kodea:

```

package businessLogic;
import java.util.Iterator;
public interface ExtendedIterator<T> extends Iterator<T> {
    T previous();
    boolean hasPrevious();
    void goFirst();
    void goLast();
}

```

```

package businessLogic;
import java.util.List;
public class ExtendedIteratorImpl<T> implements ExtendedIterator<T> {
    private List<T> elements;
    private int currentIndex = 0;
    public ExtendedIteratorImpl(List<T> elements) {
        this.elements = elements;
    }
    public T previous() {
        if (hasPrevious()) {
            return elements.get(currentIndex--);
        }
        return null;
    }
    public boolean hasPrevious() {
        return currentIndex > 0;
    }
    public void goFirst() {
        currentIndex = 0;
    }
    public void goLast() {
        currentIndex = elements.size() - 1;
    }
    public boolean hasNext() {
        return currentIndex < elements.size();
    }
    public T next() {
        if (hasNext()) {
            return elements.get(currentIndex++);
        }
        return null;
    }
}

```

Aurrekoaz gain, getEvents() metodoa aldatu da, honela utziz:

```

@WebMethod
    public ExtendedIterator<Event> getEvents(Date date) {
        dbManager.open(false);
        Vector<Event> events=dbManager.getEvents(date);
        dbManager.close();
        ExtendedIterator<Event> it = new
ExtendedIteratorImpl<Event>(events);
        return it;
    }

```

Gauzak horrela, metodo hau erabiltzen zuten lerroak eta metodoak aldatu behar izan dira, metodoak bueltatzen zuen lista modu ezberdinean korritzen baita.

DEMO:

Hurrengo klasea sortu da iterator berria probatzeko:

```

public class getEventsDemo {

    public getEventsDemo() {
    }
    @WebMethod
    public ExtendedIterator<Event> getEvents() {
        Vector<Event> events= this.getDemoEvents();
        ExtendedIterator<Event> it = new
ExtendedIteratorImpl<Event>(events);
        return it;
    }

    private Vector<Event> getDemoEvents(){
        Vector<Event> eventVector = new Vector<Event>();
        for (int i = 0; i < 10; i++) {
            Event event = new Event();
            event.setEventNumber(i + 1); // Asignar números de evento únicos
(puedes ajustar esto según tus necesidades)
            event.setDescription("Event Description " + (i + 1));
            event.setEventDate(new Date()); // Utilizar la fecha actual o
proporcionar una fecha específica
            eventVector.add(event);
        }
        return eventVector;
    }

    public static void main(String[] args) {
        getEventsDemo demo = new getEventsDemo();
        ExtendedIterator<Event> it = demo.getEvents();

        //Lehenengotik azkenengora:
        while(it.hasNext()) {
            System.out.println(it.next());
        }
        it.goLast();
        System.out.println("\n\nBREAK\n\n");

        //Azkenengotik lehenengora:

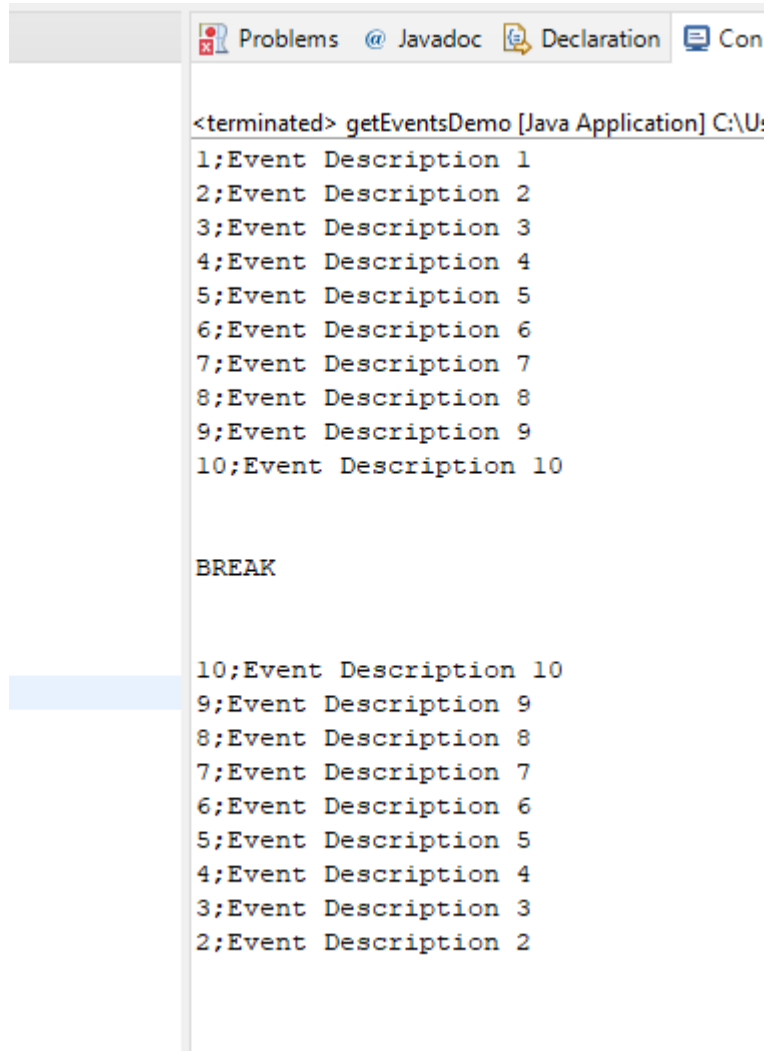
```

```

        while(it.hasPrevious()) {
            System.out.println(it.previous());
        }
    }
}

```

Eta hauxe da exekuzioaren emaitza:



```

<terminated> getEventsDemo [Java Application] C:\U:
1;Event Description 1
2;Event Description 2
3;Event Description 3
4;Event Description 4
5;Event Description 5
6;Event Description 6
7;Event Description 7
8;Event Description 8
9;Event Description 9
10;Event Description 10

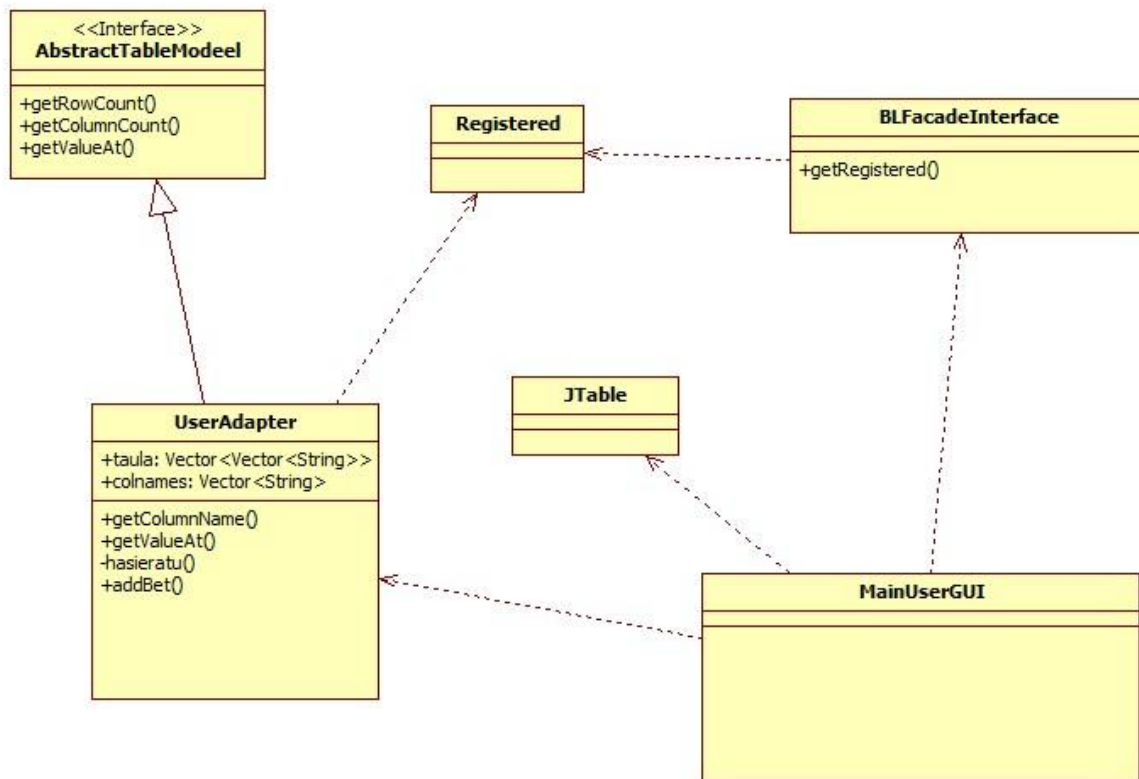
BREAK

10;Event Description 10
9;Event Description 9
8;Event Description 8
7;Event Description 7
6;Event Description 6
5;Event Description 5
4;Event Description 4
3;Event Description 3
2;Event Description 2

```

Adapter Patroia:

UML diagrama:



Kode aldaketak:

Adapter patroia implementatu ahal izateko aldaketak behean idatzi ditut, hauetatik garrantzitsuenak hemen azalduko ditut:

Lehenik UserAdapter klasea sortu dut, user batetik apustuen informazioa ateratzeko eta JTablek erabiltzeko prestatzen duena, informazioa asotzeko matrize antzeko egitura bat sortu dut vectore batean zutabe bakoitzarentzat beste bektore bat sartuaz.

Ondoren UserAdapterrek behar duen erabiltzailea jaso ahal izateko GetRegistered metodoa implementatu dut DataAccess-en erabiltzailearen izena erabiliz datu basean bilatzen duena. Metodo hau BLFacaden eta BLFacadeImplementation barruan ere sartu dut.

Hau amaitzean konturatu naiz ApplicationLauncher klaseak ez duela MainUserGUI abiaratzen, zein gure aplikazioaren hasiera puntua da eta gehitu egin dut.

Azkenik MainUserGUI klasean botoi berri bat sortu dut. Hau sakatzean aplikazioak getRegistered-rekin Maite Urrutia-ren erabiltzaile lortzen du, ondoren UserAdaptterri bidaltzen dio transformatzeko, ondoren JTable bat sortzen du informazio honekin eta bistaratu egiten du.

DataAccess klasean:

```
public Registered getRegistered(String name) {
```



```

        Registered user;
        user= (Registered) db.find(Registered.class, name);
        return user;
    }

    public void initializeDB(){
....
        Registered reg1 =new Registered("MariaUrrieta", "123", 1234);
....}

```

BLFacadeImplementation klasean:

```

@WebMethod
public Registered getRegistered(String name) {
    dbManager.open(false);
    Registered user= dbManager.getRegistered(name);
    dbManager.close();
    return user;
}

```

BLFacade klasean:

```

@WebMethod public Registered getRegistered(String name);

```

UserAdapter klasean:

```

public class UserAdapter extends AbstractTableModel{
    private Vector<String> colnames;
    private Vector<Vector<String>> taula;

    private void hasieratu() {
        colnames = new Vector<String>();
        colnames.add("Event");
        colnames.add("Question");
        colnames.add("Event Date");
        colnames.add("Bet (€)");
        taula= new Vector<Vector<String>>();
        for (int i=0;i<4;i++) {
            taula.add(new Vector<String>());
        }
    }

    public int getRowCount() {
        return taula.size();
    }

    public int getColumnCount() {
        return taula.get(0).size();
    }

    public Object getValueAt(int rowIndex, int columnIndex) {
        return taula.get(rowIndex).get(columnIndex);
    }

    public void addBet(String ev,String que, String date, String bal) {

```

```

        taula.get(0).add(ev);
        taula.get(1).add(que);
        taula.get(2).add(date);
        taula.get(3).add(bal);
    }
    public String getColumnName(int columnIndex) {
        return colnames.get(columnIndex);
    }

    public UserAdapter(Registered user) {
        this.hasieratu();
        for(ApustuAnitza apuan: user.getApustuAnitzak()) {
            Vector<Apustua> apu=apun.getApustuak();
            for (Apustua ap: apu) {
                Question que=ap.getKuota().getQuestion();
                Event ev= que.getEvent();

                this.addBet(ev.getDescription(),que.getQuestion(),apun.getData().toString(),apun.getBalioa().toString());
            }
        }
    }
}

```

ApplicationLauncher klasean:

```

MainGUI a=new MainGUI();
a.setVisible(false);
MainUserGUI b= new MainUserGUI();
b.setVisible(true);
MainUserGUI.setBussinessLogic(BLFacadeInterface);

```

MainUserGUI klasean:

```

    private JButton jButtonMarieUrrieta = null;
    ...
    private JPanel getJContentPane() {
        if (jContentPane == null) {
            jContentPane = new JPanel();
            jContentPane.setLayout(null);
            jContentPane.add(getLblNewLabel());
            jContentPane.add(getBoton3());
            jContentPane.add(getBoton4());
            jContentPane.add(getPanel());
        }
        ...
        private JButton getBoton4() {
            if (jButtonMarieUrrieta == null) {
                jButtonMarieUrrieta = new JButton();
            }
        }
    }
}

```

```

jButtonMarieUrrieta.setBounds(0, 25, 481, 38);
jButtonMarieUrrieta.setText("Table of MariaUrrieta");
jButtonMarieUrrieta.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ev) {
        BLFacade bl= getBusinessLogic();
        try {
            Registered user= bl.getRegistered("MariaUrrieta");
            JTable taula= new JTable(new UserAdapter(user));
            //System.out.println(taula.getColumnCount());
            //System.out.println(taula.getRowCount());
            //System.out.println(taula.toString());
            //System.out.println(taula.getValueAt(1,1));
            taula.validate();
            //JTable taulaproba1= new JTable();
            //taulaproba1.setVisible(true);
            //JTable taulaproba= new JTable(2,2);
            //taulaproba.setVisible(true);
            taula.setVisible(true);
            //System.out.println(taula.isVisible());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
}
return jButtonMarieUrrieta;
}

```

Exekuzio irudia:

Ezin izan dut exekzuioa gauzatu arazo ezezagun baten gatik JTable pantailan ez delako agertzen, botoaren inplementazioan dokumentatu ditut arazoa bilatzeko egin ditudan proba batzuk, horien esker dakit taula sortua dagoela, taularen balioak ondo jasotzen direla eta visibilitatea "true" dela.

Bazpaere saiatu naiz taula batzuk sortzen defektuzko kodeketa erabiliz baino horiek ere ez dira pantailaratzen.

