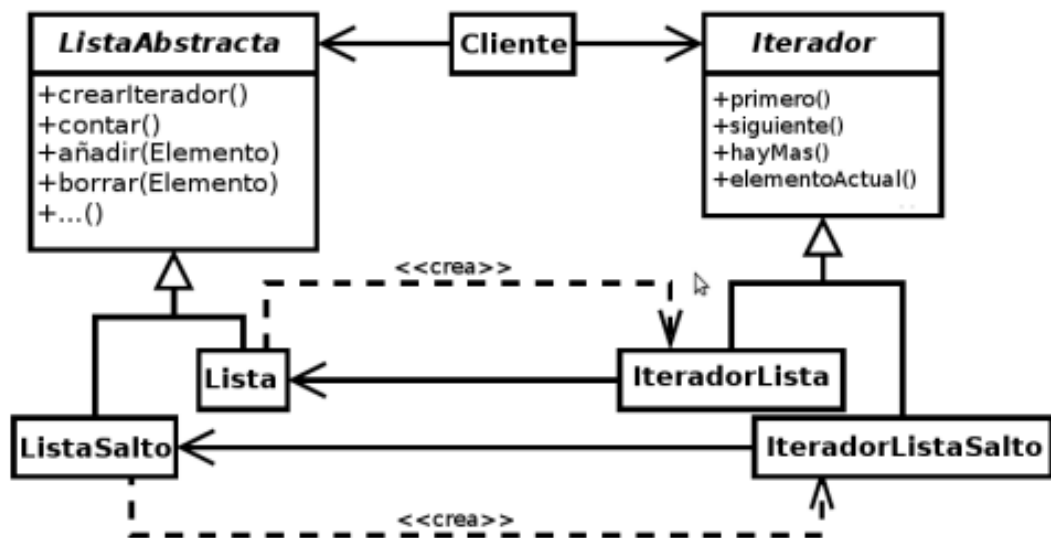


DISEINU PATROIAK



Egileak:

- Intza Larburu
- Amets Carrera
- Ioritz Aramendi

AURKIBIDEA

1. SARRERA	3
2. FACTORY PATROIA	3
3. ITERATOR PATROIA	7
4. ADAPTER PATROIA	9

1. SARRERA

Lan honetan, diseinu patroiak aplikatuko dizkiogu gure Bets proiektuari, aldaketak eginez lehendik sortua genuen proiektuari. Modu honetara, Factory, Iterator eta Adapter patroiak ezarriko ditugu. Modu honetara:

2. FACTORY PATROIA

Factory patroi honetan honakoa eskatzen da: orain arte, negozio logikako objektua ApplicationLauncherrean egon beharrean, Factory izeneko klasean zentralizatu dugu eta bertatik jasoko du applicationLauncherrek objektua. Horretarako businessLogic barnean Factory klase bat ezarri dugu, non negozio logika lokala den edo ez kontuan hartuz, uneko BLFacade implementazioa edota service berri baten portua itzuliko duen.

Hori horrela, azter ditzagun applicationLauncher klasean egin behar izan ditugun aldaketak eta faktory klasearen kodea.

ApplicationLauncherren hasierako egoera:

```

1 package gui;
2
3 import java.awt.Color;
17
18 public class ApplicationLauncher {
19
20
21
22     public static void main(String[] args) {
23
24         ConfigXML c=ConfigXML.getInstance();
25
26         System.out.println(c.getLocale());
27
28         Locale.setDefault(new Locale(c.getLocale()));
29
30         System.out.println("Locale: "+Locale.getDefault());
31
32         Hasierako_pantailaGUI a=new Hasierako_pantailaGUI();
33         GroupLayout groupLayout = new GroupLayout(a.getContentPane());
34         groupLayout.setHorizontalGroup(
35             groupLayout.createParallelGroup(Alignment.LEADING)
36                 .addGap(0, 426, Short.MAX_VALUE)
37         );
38         groupLayout.setVerticalGroup(
39             groupLayout.createParallelGroup(Alignment.LEADING)
40                 .addGap(0, 253, Short.MAX_VALUE)
41         );
42         a.getContentPane().setLayout(groupLayout);
43         a.setVisible(true);
44
45
46         try {
47
48             BLFacade appFacadeInterface;
49             UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
50             if (c.isBusinessLogicLocal()) {
51
52                 //In this option the DataAccess is created by FacadeImplementationWS
53                 //appFacadeInterface=new BLFacadeImplementation();
54
55                 //In this option, you can parameterize the DataAccess (e.g. a Mock DataAccess object)
56
57                 DataAccess da= new DataAccess(c.getDataBaseOpenMode().equals("initialize"));
58                 appFacadeInterface=new BLFacadeImplementation(da);
59
60             }
61
62             else { //If remote
63
64                 String serviceName= "http://"+c.getBusinessLogicNode() +":"+ c.getBusinessLogicPort()+"/ws/";
65
66                 //URL url = new URL("http://localhost:9999/ws/ruralHouses?wsdl");
67                 URL url = new URL(serviceName);
68
69
70                 //1st argument refers to wsdl document above
71                 //2nd argument is service name, refer to wsdl document above
72                 QName qname = new QName("http://businessLogic/", "FacadeImplementationWSService");
73                 QName qname = new QName("http://businessLogic/", "BLFacadeImplementationService");
74
75                 Service service = Service.create(url, qname);
76
77                 appFacadeInterface = service.getPort(BLFacade.class);
78
79                 /*if (c.getDataBaseOpenMode().equals("initialize"))
80                     appFacadeInterface.initializeBD();
81                 */
82                 MainGUI.setBussinessLogic(appFacadeInterface);
83
84             } catch (Exception e) {
85                 a.jLabelSelectOption.setText("Error: "+e.toString());
86                 a.jLabelSelectOption.setForeground(Color.RED);
87
88                 System.out.println("Error in ApplicationLauncher: "+e.toString());
89             }
90             //a.pack();
91
92         }
93
94     }
95 }

```

Application Launcherren oraingo egoera:

```
1 package gui;
2
3 import java.awt.Color;
17
18 public class ApplicationLauncher {
19
20
21
22 public static void main(String[] args) {
23
24     ConfigXML c=ConfigXML.getInstance();
25
26     System.out.println(c.getLocale());
27
28     Locale.setDefault(new Locale(c.getLocale()));
29
30     System.out.println("Locale: "+Locale.getDefault());
31
32     Hasierako_pantailaGUI a=new Hasierako_pantailaGUI();
33     GroupLayout groupLayout = new GroupLayout(a.getContentPane());
34     groupLayout.setHorizontalGroup(
35         groupLayout.createParallelGroup(Alignment.LEADING)
36             .addGap(0, 426, Short.MAX_VALUE)
37     );
38     groupLayout.setVerticalGroup(
39         groupLayout.createParallelGroup(Alignment.LEADING)
40             .addGap(0, 253, Short.MAX_VALUE)
41     );
42     a.getContentPane().setLayout(groupLayout);
43     a.setVisible(true);
44
45
46     try {
47
48         BLFacade appFacadeInterface;
49
50         UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
51         if (c.isBusinessLogicLocal()) {
52             appFacadeInterface=Factory.createFactory(true);
53
54
55
56
57
58         else { //If remote
59             appFacadeInterface = Factory.createFactory(false);
60
61             MainGUI.setBussinessLogic(appFacadeInterface);
62
63
64
65
66         } catch (Exception e) {
67             a.jLabelSelectOption.setText("Error: "+e.toString());
68             a.jLabelSelectOption.setForeground(Color.RED);
69
70             System.out.println("Error in ApplicationLauncher: "+e.toString());
71         }
72         //a.pack();
73
74
75     }
76
77 }
```

Ikusi dugunez, orain ez da applicationLauncherrean sortzen negozio logikako objektua, baizik eta factory klaseko createFactory metodoari dei egiten zaio eta bertatik jasotzen da instantzia.

Factory klasea honako hau dugu:

```
import javax.xml.namespace.*;
import javax.xml.ws.Service;

import configuration.ConfigXML;

public class Factory {

    public static BLFacade createFactory(boolean local){
        if(local) {
            return new BLFacadeImplementation();
        }
        if(!local) {
            try {
                ConfigXML c = ConfigXML.getInstance();
                String serviceName= "http://"+c.getBusinessLogicNode() +":"+ c.getBusinessLogicPort()+"/ws/

                //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/", "FacadeImplementationWSService");
                QName qname = new QName("http://businessLogic/", "BLFacadeImplementationService");

                Service service = Service.create(url, qname);

                return service.getPort(BLFacade.class);
            } catch (Exception e) {
                System.out.println("Error connecting: " + e.toString());
            }
        }
        return null;
    }
}
```

Bertan, bolear bat jasotzen da lokala den edo ez jakiteko eta horren arabera itzultzen zaio BLFacadeImplementation edota sorturiko service berria.

Modu honetan, esan dezakegu Factory dela *Creator* rola jokatzeko duena, beronek sortzen baitu instantzia. BLFacade *Product* rola jokatzeko du eta BLFacadeImplementationek, Factory objektuari inplementazioa ematen baitio.

3. ITERATOR PATROIA

Iterator patroiarri dagokionez, gertaeren iteratzaile bat gauzatzeko eskatzen zaigu. Hain zuzen, gertaerak bi noranzkoetan iteratuko dituen iterator bat sortzeko. Horretarako, honako iteratorTest klasea sortu dugu:

```
package domain;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

import businessLogic.BLFacade;
import businessLogic.Factory;

public class IteratorTest {

    public static void main(String [] args) {
        boolean isLocal = true;
        //Facade objektua lortu
        BLFacade blFacade = (new Factory()).createFactory(isLocal);
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        Date date;
        try {
            date = sdf.parse("17/12/2023");
            ExtendsIterator<Event> iterator = blFacade.getEventsIterator(date);
            Event e;
            System.out.println("_____");
            System.out.println( AZKENEKO AURREKARRAKA );
            iterator.goLast(); //Azkeneko elementuan kokatu
            while (iterator.hasPrevious()) {
                e = iterator.previous();
                System.out.println(e.toString());
            }

            System.out.println();
            System.out.println("_____");
            System.out.println("AURRETIK ATZERAKA");
            iterator.goFirst(); // Lehen elem. kokatu
            while (iterator.hasNext()) {
                e = iterator.next();
                System.out.println(e.toString());
            }

        } catch (ParseException e) {
            e.getMessage();
        }
    }
}
```

Ikus dezakegunez, ExtendsIterator<Event> objetua sortu dugu eta .goFirst() / .goLast() metodoen bidez, gertaeren lista iteratuko dugu bi noranzkotan. Horretarako, honako bi klase hauek sortu ditugu, ExtendsIterator eta ExtendsIteratoEvent:

```

1  package domain;
2
3  import java.util.Iterator;
4
5  public interface ExtendedIterator<Event> extends Iterator<Event> {
6      //uneko elementua itzultzen du eta aurrekora pasatzen da
7      public Event previous();
8      //true aurreko elementua existitzen bada.
9      public boolean hasPrevious();
10     //Lehendabiziko elementuan kokatzen da.
11     public void goFirst();
12     //Azkeneko elementuan kokatzen da.
13     public void goLast();
14 }

```

```

1  package domain;
2
3  import java.util.Vector;
4
5  public class ExtendedIteratorEvents implements ExtendedIterator<Event> {
6
7      private int pos;
8      private Vector<Event> lista;
9
10
11     public ExtendedIteratorEvents (Vector<Event> list) {
12
13         this.lista=list;
14         this.pos = 0;
15     }
16
17
18     @Override
19     public boolean hasNext() {
20         // TODO Auto-generated method stub
21         if (this.pos <= this.lista.size()-1) {
22             return true;
23         }

```

Azkenik, BLFacade eta BLFacadeImplementationen honako metodoa gehitu dugu gertaeraz osaturiko iteratzailearen objektua itzuli dezan:

Modu honetara burutu dugu iterator patroia.

```

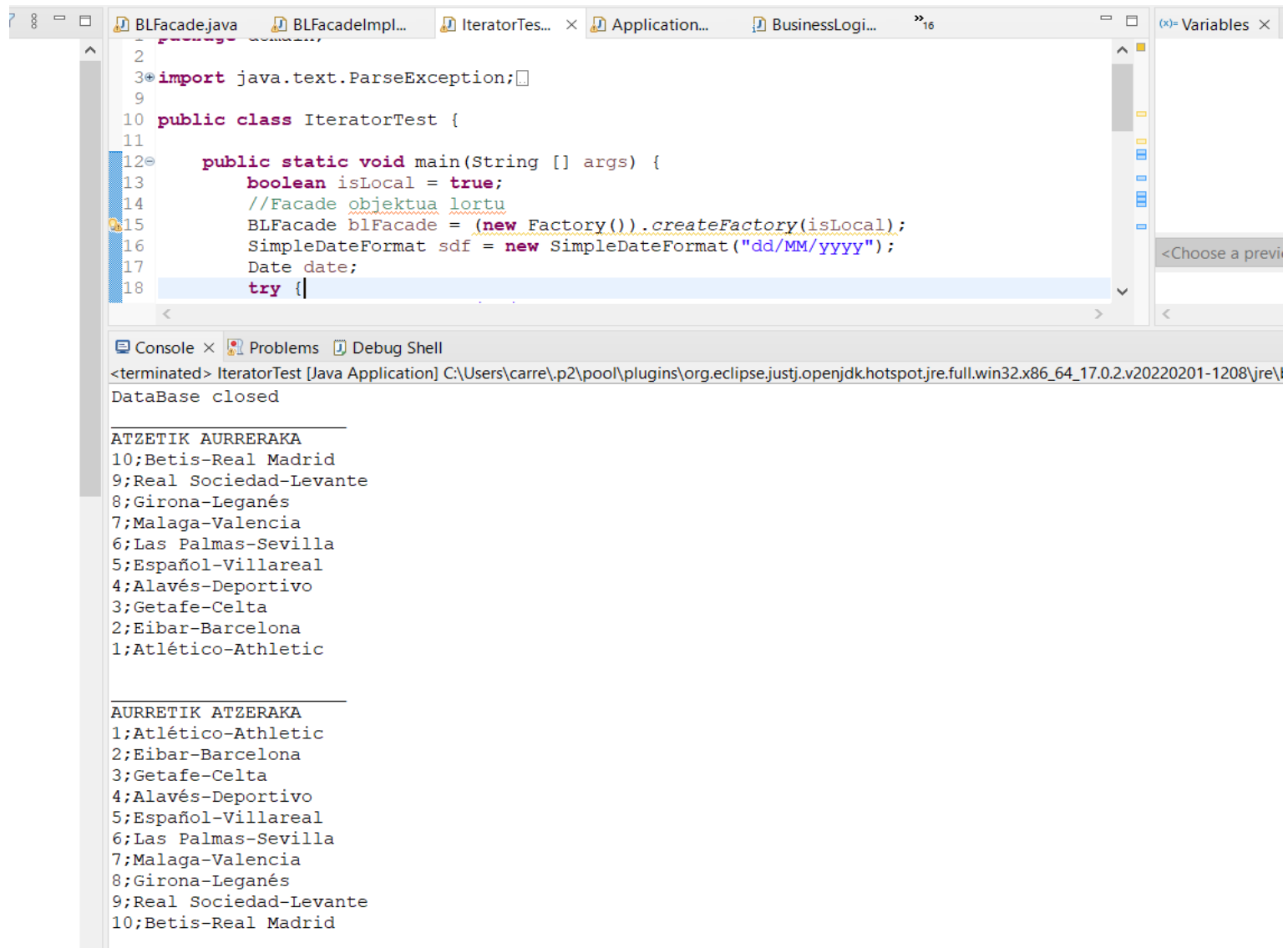
@WebMethod
public ExtendedIterator getEventsIterator(Date date) {
    return new ExtendedIteratorEvents(getEvents(date));
}

```


Hain zuzen, metodo horrek datu baseko beste metodo honi egingo dio deia eta modu honetara eskuraturko ditu gertaera guztiak:

```
250 public Vector<Event> getEvents(Date date) {
251     System.out.println(">>> DataAccess: getEvents");
252     Vector<Event> res = new Vector<Event>();
253     TypedQuery<Event> query = db.createQuery("SELECT ev FROM Event ev WHERE ev.eventDate=?1", Event.class);
254     query.setParameter(1, date);
255     List<Event> events = query.getResultList();
256     for (Event ev:events){
257         system.out.println(ev.toString());
258         res.add(ev);
259     }
260     return res;
261 }
```

ITERATOR PATROIAREN EXEKUAZIOAREN IRUDIA:



The screenshot shows the Eclipse IDE with the following components:

- Editor:** Displays the source code of `IteratorTest.java`. The code includes imports for `java.text.ParseException` and `java.util.Iterator`, and defines a `main` method that creates a `BLFacade` object and calls `getEvents` with a date.
- Console:** Shows the output of the program. It starts with a message indicating the database is closed, followed by two lists of football teams: `ATZETIK AURRERAKA` and `AURRETIK ATZERAKA`. Each list contains 10 teams, numbered 1 to 10.

Console Output:

```
<terminated> IteratorTest [Java Application] C:\Users\carre\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\java.exe
DataBase closed

ATZETIK AURRERAKA
10;Betis-Real Madrid
9;Real Sociedad-Levante
8;Girona-Leganés
7;Malaga-Valencia
6;Las Palmas-Sevilla
5;Español-Villareal
4;Alavés-Deportivo
3;Getafe-Celta
2;Eibar-Barcelona
1;Atlético-Athletic

AURRETIK ATZERAKA
1;Atlético-Athletic
2;Eibar-Barcelona
3;Getafe-Celta
4;Alavés-Deportivo
5;Español-Villareal
6;Las Palmas-Sevilla
7;Malaga-Valencia
8;Girona-Leganés
9;Real Sociedad-Levante
10;Betis-Real Madrid
```

4. ADAPTER PATROIA

Adapter patroiarri dagokionez, proiektuan aukera bat gehitzeko eskatzen da, bertan, erabiltzaile batek eginiko apustu guztiak JTable batean erakusteko. Hau, Erabiltzaile kalsea aldatu gabe egin behar da, beraz, esan bezala, adapter patroia erabiliko dugu hau egiteko. Horretarako, hemen ere hainbat klase aldatu behar izan ditugu. Hasteko, ApustuakTableModel sortu dugu, eginiko apustu guztien datuak bilduko dituen:

```
package domain;
import java.util.Vector;
import javax.swing.table.AbstractTableModel;

public class ApustuakTableModel extends AbstractTableModel {
    private String[] zutabeIzenak= {"Event", "Question", "Bet(€)"};
    private Vector<Apustua> datuak;

    public ApustuakTableModel(Vector<Apustua> datuak) {
        this.datuak=datuak;
    }

    public int getColumnCount() {
        return zutabeIzenak.length;
    }

    public int getRowCount() {
        int totalOptions=0;
        for(Apustua dat:datuak) {
            totalOptions=totalOptions+dat.getErantzunKop();
        }
        return totalOptions;
    }

    public Object getValueAt(int row, int col) {
        int currentRow = 0;

        for (Apustua apustua : datuak) {
            for (Aukera auk : apustua.getErantzunak()) {
                if (currentRow == row) {
                    switch (col) {
                        case 0:
                            return auk.getGaldera().getEvent();
                        case 1:
                            return auk.getGaldera();
                        case 2:
                            return apustua.getZenbatekoa();
                        default:
                            return null;
                    }
                }
                currentRow++;
                System.out.println("Event: " + auk.getGaldera().getEvent());
            }
        }
    }
}
```

```

        }
        currentRow++;
        System.out.println("Event: " + auk.getGaldera().getEvent());
        System.out.println("Question: " + auk.getGaldera());
        System.out.println("Bet(€): " + apustua.getZenbatekoa());

    }

}

return null;
}
@Override
public String getColumnName(int column) {
    return zutabeIzenak[column];
}
}

```

Eta bestalde eginiko apustuen interfaze grafikoa sortu dugu, non, apustuen taula agertuko den, ondorengo eraikitzailearekin:

```

public EgindakoApustuakGUI(String log) {
    super();
    nirePantaila=this;
    nirePantaila.setTitle(ResourceBundle.getBundle("Etiquetas").getString("Ban"));
    table = new JTable();
    this.negozioLogika=MainGUI.getBusinessLogic();
    Erregistratua e=negozioLogika.erregistratuaItzuli(log);
    Vector<Apustua> apustuak=e.getApustuak();
    modeloTabla = new ApustuakTableModel(apustuak);
    table.setModel(modeloTabla);
    // Agrega la tabla al GUI
    JScrollPane scrollPane = new JScrollPane(table);
    getContentPane().add(scrollPane, BorderLayout.CENTER);

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    pack();
    setLocationRelativeTo(null);
}

```

Interfaze honetara iristeko, ErregistratuaGUI interfazeaz botoi berri bat gehitu dugu erregistratu batek egin dituen apustu guztiak ikusi ahal izateko.

```
private JButton getApostuakIkusi() {
    if (btnApostuakIkusi == null) {
        btnApostuakIkusi = new JButton();
        btnApostuakIkusi.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JFrame a= new EgindakoApustuakGUI(log);
                nirePantaila.setVisible(false);
                a.setVisible(true);
            }
        });
        btnApostuakIkusi.setText(this.log+"-en apustuak ikusi");
    }
    return btnApostuakIkusi;
}
```

Aipatu beharra dago, gure hainbat akats topatu ditugula aurretik genuen kodean, eta beraz, zailtasunak izan ditugula ariketa hau ongi dagoen egiaztatzeko. Hala ere, kodea lehen bi patroiek ongi funtzionatzen dute eta adapter patroia ere aipatu dugun moduan geldituko litzatekeela.