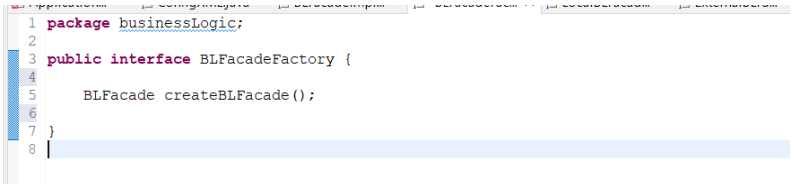
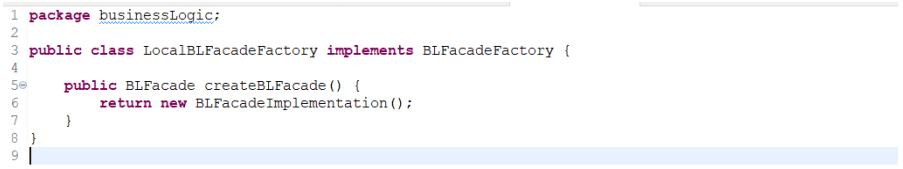
Factory Method Patroia

Interfaz bat eta bi klase sortu ditugu. Sortutako interfazea Creator klasearen rola jokatzen du, BLFacade Product rola du eta BLFacadeImplementation ConcreteProduct klasearen rola hartzen du.



```java
package businessLogic;

public interface BLFacadeFactory {

    BLFacade createBLFacade();

}
```

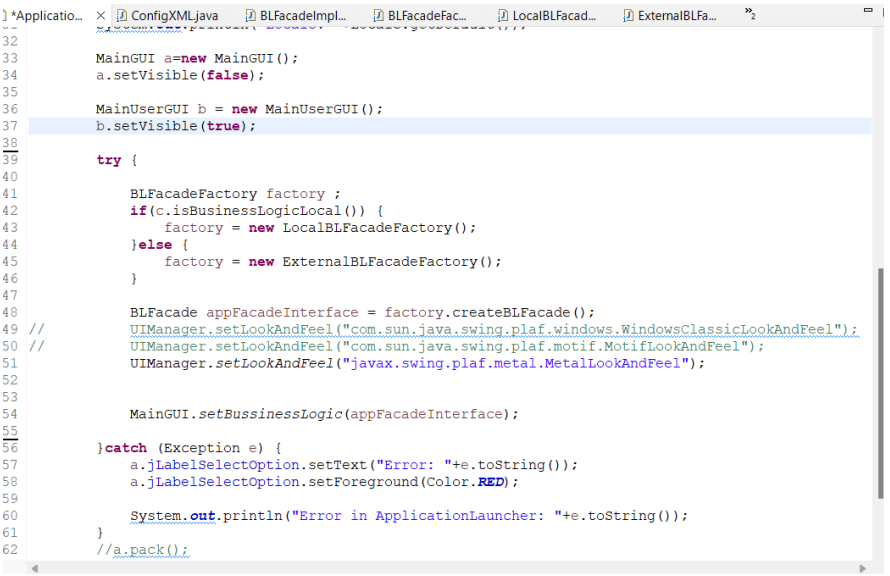
2 klase sortu ditugu, biak BLFacadeFactory interfazea implementatzen dute.
Klase bat BLFacadeFactory lokala sortzen duena eta bestea kanpoko bat sortzen duena.



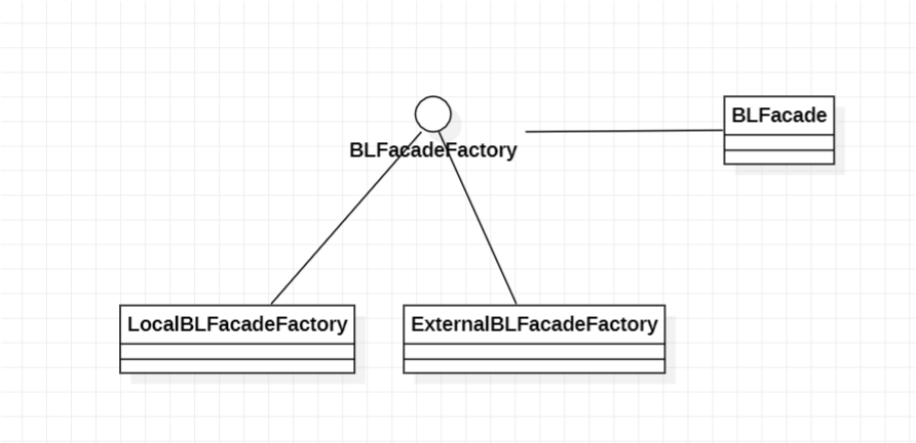
```java
package businessLogic;

public class LocalBLFacadeFactory implements BLFacadeFactory {

    public BLFacade createBLFacade() {
        return new BLFacadeImplementation();
    }
}
```

```java
package businessLogic;

import java.net.URL;

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

import configuration.ConfigXML;

public class ExternalBLFacadeFactory implements BLFacadeFactory {
    public BLFacade createBLFacade() {
        try {
            String serviceName = "http://"+ConfigXML.getInstance().getBusinessLogicNode() +
                                 ":" + ConfigXML.getInstance().getBusinessLogicPort() +
                                 "/ws/" + ConfigXML.getInstance().getBusinessLogicName() + "?wsdl";
            URL url = new URL(serviceName);
            QName qname = new QName("http://businessLogic/", "BLFacadeImplementationService");
            Service service = Service.create(url, qname);
            return service.getPort(BLFacade.class);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Main klasean if bat jarri dugu, factory lokala edo externoa sortzeko.



```java
        MainGUI a=new MainGUI();
        a.setVisible(false);

        MainUserGUI b = new MainUserGUI();
        b.setVisible(true);

        try {

            BLFacadeFactory factory ;
            if(c.isBusinessLogicLocal()) {
                factory = new LocalBLFacadeFactory();
            }else {
                factory = new ExternalBLFacadeFactory();
            }

            BLFacade appFacadeInterface = factory.createBLFacade();
//          UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel");
//          UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
            UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");


            MainGUI.setBussinessLogic(appFacadeInterface);

        }catch (Exception e) {
            a.jLabelSelectOption.setText("Error: "+e.toString());
            a.jLabelSelectOption.setForeground(Color.RED);

            System.out.println("Error in ApplicationLauncher: "+e.toString());
        }
        //a.pack();
```



Iterator Patroia

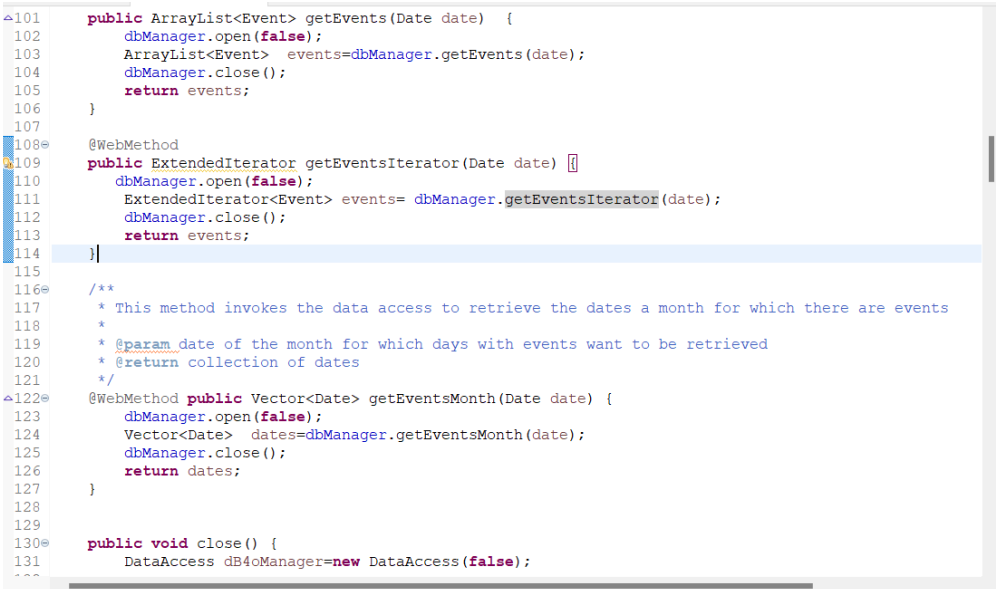
ExtendedIterator interfaze bat sortu dugu, Iterator klasetik heredatzen da eta 4 metodo gehitu diogu.

```
1  package businessLogic;
2
3  import java.util.Date;
4  import java.util.Iterator;
5  import java.util.List;
6
7  import domain.Event;
8
9  public interface ExtendedIterator<Object> extends Iterator<Object> {
10
11     //uneko elementua itzultzen du eta aurrekora pasatzen da
12     public Object previous();
13
14     //true aurreko elementua existitzen bada.
15     public boolean hasPrevious();
16
17     //Lehendabiziko elementuan kokatzen da.
18     public void goFirst();
19
20     //Azkeneko elementuan kokatzen da.
21     public void goLast();
22
23
24
25 }
```

BLFacade klasean getEventsIterator metodoa gehitu dugu.

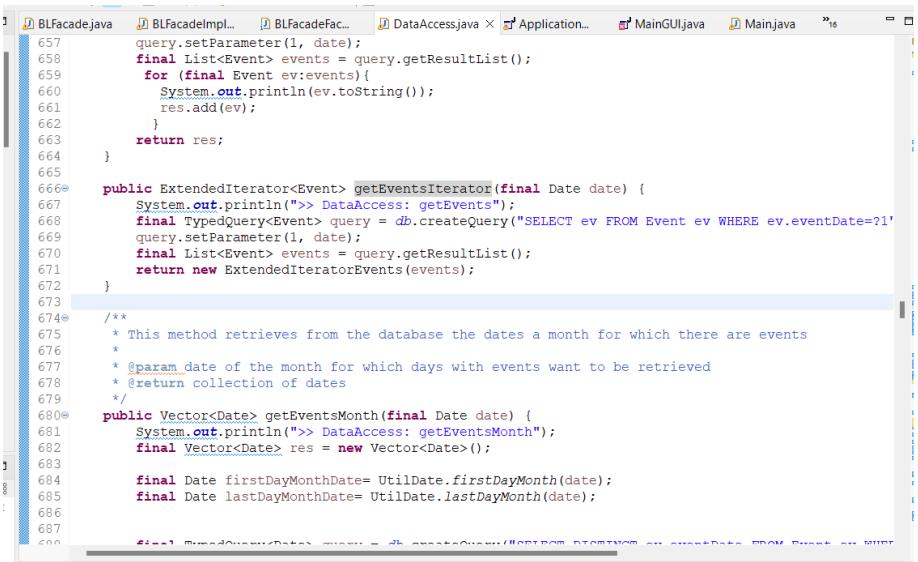
```
106
107     @WebMethod public List<Registered> rankingLortu();
108
109     @WebMethod public List<Event> getEventsAll();
110
111
112     @WebMethod public boolean gertaerakKopiatu(Event e, Date date);
113
114     @WebMethod public boolean jarraitu(Registered jabea, Registered jarraitua, Double limit);
115
116
117     @WebMethod public Sport popularrenaLortu();
118
119     @WebMethod public void ezJarraituTaldea(Registered u);
120
121     @WebMethod public List<Team> getAllTeams();
122
123     @WebMethod public void jarraituTaldea(Registered u, Team t);
124
125     @WebMethod public Registered findUser(Registered user);
126
127     @WebMethod public List<Event> getEventsTeam(Team t);
128
129
130     @WebMethod public Team findTeam(Registered u);
131
132     @WebMethod public Sport findSport(Event q);
133
134     @WebMethod public ExtendedIterator getEventsIterator(Date date);
135 }
136
```

BLFacadeImplementation klasean getEventsIterator() metodoa gehitu dugu, sortutako ExtendedIterator interfazea itzultzen duena.



```
101     public ArrayList<Event> getEvents(Date date)  {
102         dbManager.open(false);
103         ArrayList<Event>  events=dbManager.getEvents(date);
104         dbManager.close();
105         return events;
106     }
107
108     @WebMethod
109     public ExtendedIterator getEventsIterator(Date date) {
110        dbManager.open(false);
111         ExtendedIterator<Event> events= dbManager.getEventsIterator(date);
112         dbManager.close();
113         return events;
114     }
115
116     /**
117      * This method invokes the data access to retrieve the dates a month for which there are events
118      *
119      * @param date of the month for which days with events want to be retrieved
120      * @return collection of dates
121      */
122     @WebMethod public Vector<Date> getEventsMonth(Date date) {
123         dbManager.open(false);
124         Vector<Date>  dates=dbManager.getEventsMonth(date);
125         dbManager.close();
126         return dates;
127     }
128
129
130     public void close() {
131         DataAccess dB4oManager=new DataAccess(false);
```

DataAcces klasean getEventsIterator metodoa gehitu diogu ere bai.



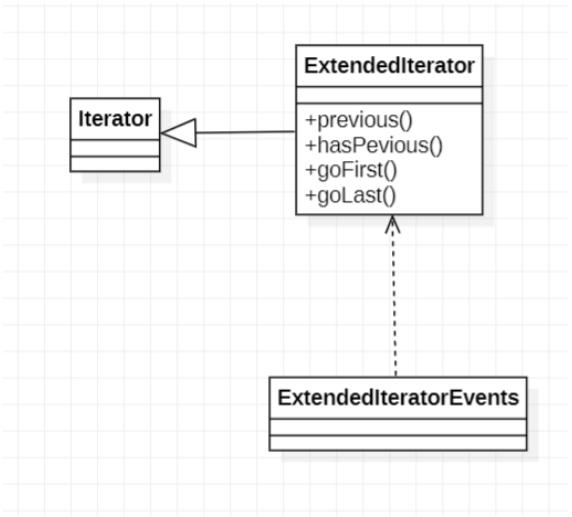
```
BLFacade.java | BLFacadeImpl... | BLFacadeFac... | DataAccess.java | Application... | MainGUI.java | Main.java

657         query.setParameter(1, date);
658         final List<Event> events = query.getResultList();
659          for (final Event ev:events){
660            System.out.println(ev.toString());
661            res.add(ev);
662           }
663         return res;
664     }
665
666     public ExtendedIterator<Event> getEventsIterator(final Date date) {
667         System.out.println(">> DataAccess: getEvents");
668         final TypedQuery<Event> query = db.createQuery("SELECT ev FROM Event ev WHERE ev.eventDate=?1"
669         query.setParameter(1, date);
670         final List<Event> events = query.getResultList();
671         return new ExtendedIteratorEvents(events);
672     }
673
674     /**
675      * This method retrieves from the database the dates a month for which there are events
676      *
677      * @param date of the month for which days with events want to be retrieved
678      * @return collection of dates
679      */
680     public Vector<Date> getEventsMonth(final Date date) {
681         System.out.println(">> DataAccess: getEventsMonth");
682         final Vector<Date> res = new Vector<Date>();
683
684         final Date firstDayMonthDate= UtilDate.firstDayMonth(date);
685         final Date lastDayMonthDate= UtilDate.lastDayMonth(date);
686
687
```

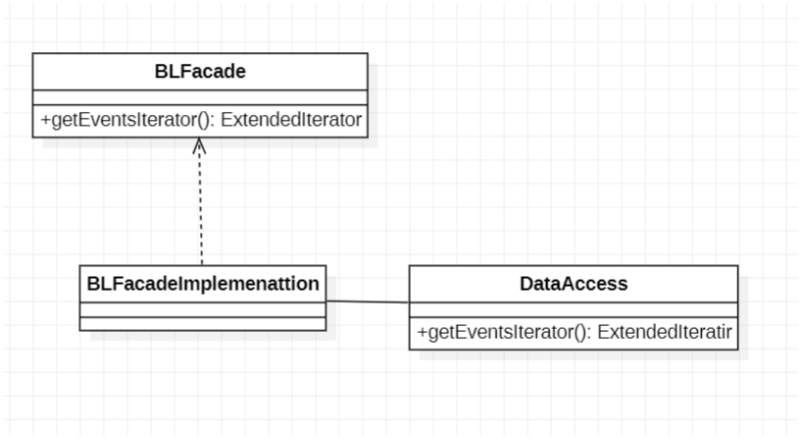
ExtendedIteratorEvents klasea sortu dugu, lehen sortu dugun interfazeak dituen metodoak berridatziz.

```java
package businessLogic;


import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.NoSuchElementException;
import java.util.Vector;

import domain.Event;

public class ExtendedIteratorEvents implements ExtendedIterator<Event> {
    private List<Event> elements;
    private int currentPosition;

    public ExtendedIteratorEvents(List<Event> events) {
        this.elements = new ArrayList<Event>();
        this.currentPosition = 0;
    }


    public boolean hasPrevious() {
        return currentPosition > 0;
    }


   public Event previous() {
        if (!hasPrevious()) {
            throw new NoSuchElementException("Ez daude");
        }
        currentPosition--;
        return elements.get(currentPosition);
   }


   public void goFirst() {
        currentPosition = 0;
   }


   public void goLast() {
        currentPosition = elements.size() - 1;
   }


   public boolean hasNext() {
        return currentPosition < elements.size();
   }


    public Event next() {
         if (!hasNext()) {
                throw new NoSuchElementException("No hay más elementos");
            }
            Event nextElement = elements.get(currentPosition);
            currentPosition++;
            return nextElement;
    }

}
```

Gure main klasea ApplicationLauncher deiturikoan hurrengoa egin dugu:isLocal bada ordenatzeko esan dugu, horretarako booleano bat sortu dugu eta if .isBussinessLogicLocal true bada booleanoa aldatuko da eta ondorioz ordenatuko da.

```
MainGUI a=new MainGUI();
a.setVisible(false);

MainUserGUI b = new MainUserGUI();
b.setVisible(true);
boolean isLocal=false;
try {

    BLFacadeFactory factory ;
    if(c.isBusinessLogicLocal()) {
        factory = new LocalBLFacadeFactory();
        isLocal=true;
    }else {
        factory = new ExternalBLFacadeFactory();
    }

    BLFacade appFacadeInterface = factory.createBLFacade();
    if(isLocal=true) {
        SimpleDateFormat  sdf =  new SimpleDateFormat("dd/MM/yyyy");
        Date    date;
        try {
            date =    sdf.parse("17/12/2023");
            ExtendedIterator<Event> i = appFacadeInterface.getEventsIterator(date);
            Event    e;
            i.goLast();
            while (i.hasPrevious())    {
                e =    i.previous();
                System.out.println(e.toString());
                }
                System.out.println();

                i.goFirst();
                while (i.hasNext())    {
                    e =    i.next();
                    System.out.println(e.toString());
                }
            }    catch (ParseException    e1)    {
                System.out.println("Data arazoak    " +    "17/12/2023");
            }
        }

        //UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel");
        //UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
        UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
        MainGUI.setBussinessLogic(appFacadeInterface);


    }catch (Exception e) {
        a.jLabelSelectOption.setText("Error: "+e.toString());
        a.jLabelSelectOption.setForeground(Color.RED);

        System.out.println("Error in ApplicationLauncher: "+e.toString());
    }
    //a.pack();


}
```





Adapter Patroia

Main2 klase berri bat sortu dugu, Maite erabiltzailearen datuak definituz.

```
public static void main(String[] args) {
    Registered registered = new Registered("Maite", "password", 123456);

    // Añadir 5 apuestas
    for (int i = 0; i < 5; i++) {
        Sport sport = new Sport("Football");
        Team localTeam = new Team("Real Sociedad");
        Team awayTeam = new Team("Real Madrid");
        Event event = new Event("Event " + (i + 1), new Date(), localTeam, awayTeam);
        Question question = new Question("Zeinek irabazi" + (i + 1), 10.0, event);
        Quote quote = new Quote(2.0, "Real Sociedad", question);

        if (quote.getQuote() == null) {
            quote.setQuote(2.0);
        }

        ApustuAnitza apustuAnitza = new ApustuAnitza(registered, 2.0);
        Apustua apustua = new Apustua(apustuAnitza, quote);
        apustua.getApustuAnitza().setBalioa(2.0);
        apustua.getKuota().getQuestion().getEvent().setDescription("Real Sociedad-Real Madrid");
        apustua.getKuota().getQuestion().setQuestion("Zeinek irabazi");
        SimpleDateFormat  sdf =  new SimpleDateFormat("dd/MM/yyyy");
        Date  date;
        try {
            date = sdf.parse("17/12/2023");
            apustua.getKuota().getQuestion().getEvent().setEventDate(date);
        }   catch (ParseException    e1)    {
            System.out.println("Data arazoak     " +    "17/12/2023");
        }

        apustua.getKuota().setQuote(2.0);
```

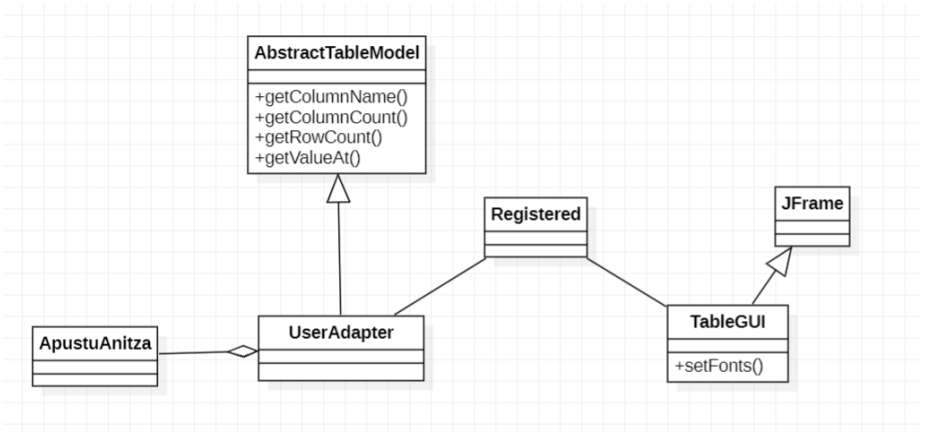
TableGUI klasea sortu dugu JFrame klasetik heredatzen dena. Pantailan agertu behar den taula ikusteko beharrezkoa da.

```java
import domain.ApustuAnitza;
import domain.Apustua;
import domain.Registered;
import domain.UserAdapter;

public class TableGUI extends JFrame {
    JTable table;
    Registered registered;

    public TableGUI(Registered registered) {
        this.setTitle(registered.getUsername());
        this.registered = registered;
        setFonts();
        TableModel tm = new UserAdapter(registered);
        table = new JTable(tm);

        table.setRowHeight(36);
        JScrollPane pane = new JScrollPane(table);
        this.getContentPane().add(pane);

        // Configurar la ventana
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);
        setLocationRelativeTo(null); // Centrar la ventana
    }

    private static void setFonts() {
        Font font = new Font("Dialog", Font.PLAIN, 18);
        UIManager.put("Table.font", font);
```

UserAdapter klasea 4 zutabe dituen tabla sortzen du. Zutabe bakoitzaren izena definitu eta zer balio pantailaratu behar digun bakoitzean esan diogu.

```java
package domain;

import java.util.Iterator;
import java.util.Vector;

import javax.swing.table.AbstractTableModel;

public class UserAdapter extends AbstractTableModel {
    protected Registered registered;
    protected Vector<ApustuAnitza> apustuAnitzak = new Vector<ApustuAnitza>();
    protected String[] columnNames = new String[] { "Event", "Question", "EventDate", "Bet€" };

    public UserAdapter(Registered r) {
        Vector<ApustuAnitza> aa = r.getApustuAnitzak();
        Iterator<ApustuAnitza> i = aa.iterator();
        while (i.hasNext()) {
            apustuAnitzak.add(i.next());
        }
        this.registered = r;
    }

    public int getColumnCount() {
        return 4;
    }

    public String getColumnName(int i) {
        return columnNames[i];
    }

    public int getRowCount() {
        return registered.getApustuAnitzak().size();
```

UML



Ez dugu lortu tabla betetzea.