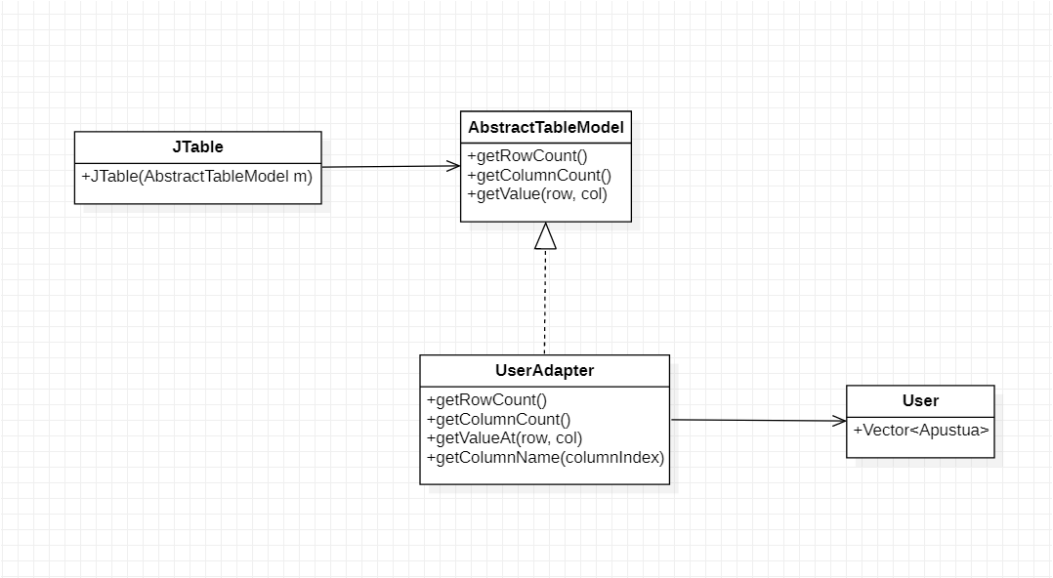
Esteka: https://github.com/aitorcas23/Bets21Azkena

Adapter diseinu partroia bets21Aplikazioan

UML DIAGRAMA



ALDATUTAKO KODEA

JTable egiturak ezin du user klasea erabili honek ez dituelako AbstractTableModel interfazearen metodoak. Zuzenean metodo hauek USer klasean txertatzea ere ez da eraginkorra, horretarako UserAdapter klasea sortuko dugu, klase honek AbstactTableModel interfazea implementatuko du, ondorioz bere metodoak erabili ahalko ditu User klasean aldaketarik egin gabe.

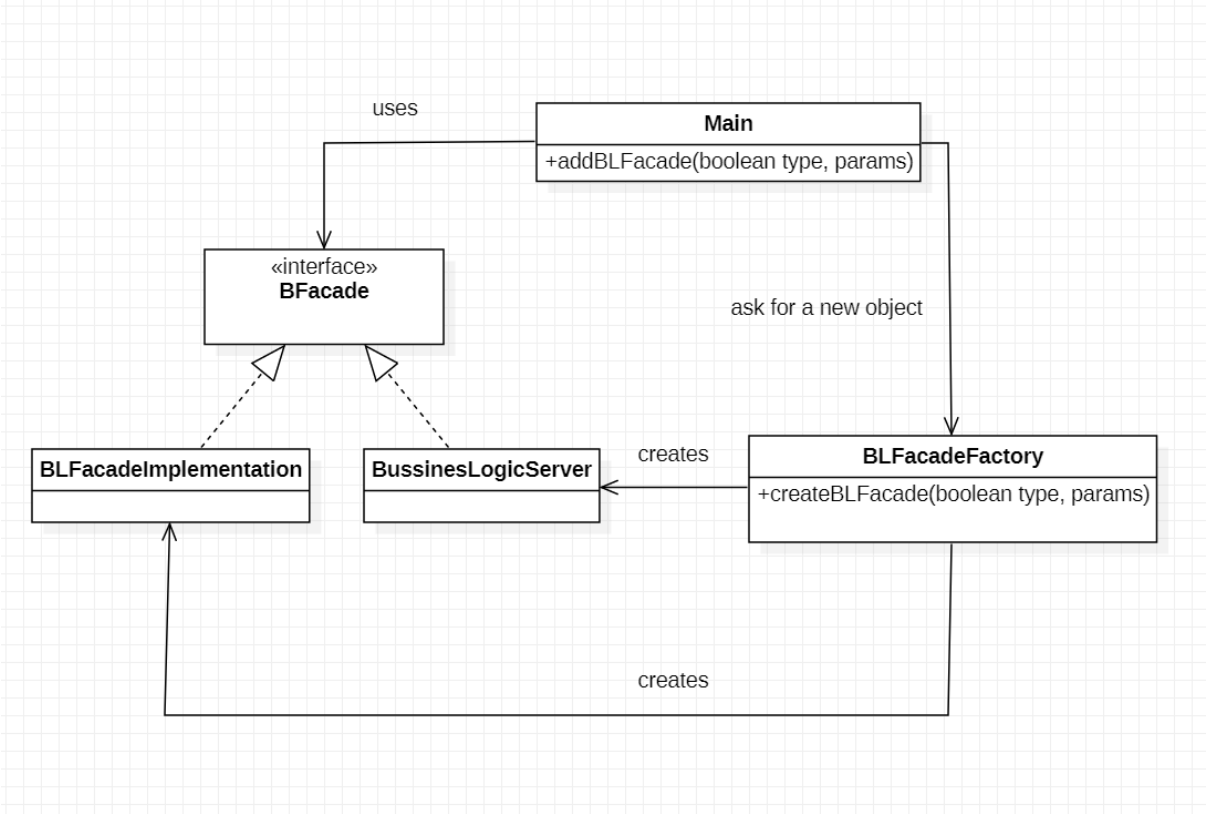
```java
import javax.swing.table.AbstractTableModel;

public class UserAdapter extends AbstractTableModel {
    private Bezeroa bezeroa;

    public UserAdapter(Bezeroa bez) {
        this.bezeroa = bez;
    }
    @Override
    public int getRowCount() {
        return bezeroa.getApustuak().size();
    }

    @Override
    public int getColumnCount() {
        return 4;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        return this.getValueAt(rowIndex, columnIndex);
    }

    @Override
    public String getColumnName(int columnIndex) {
        if(columnIndex==0) {
            return "Event";
        }
        else if(columnIndex==1) {
            return "Question";
        }
        else if(columnIndex==2) {
            return "Event Date";
        }
        else {
            return "Bet(€)";
        }
    }
```

Horretaz aparte AdapterProbaGUI klasearen bidez deitzen diogu UserAdapter klaseari taula erakutsi dezan.

```java
package gui;

import java.awt.BorderLayout;

public class AdapterProbaGUI extends JFrame {

    private JPanel contentPane;
    private JTable table;

    /**
     * Launch the application.
     */

    /**
     * Create the frame.
     */
    public AdapterProbaGUI(Bezeroa bezeroa) {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        AbstractTableModel adapterTable = new UserAdapter(bezeroa);
        System.out.println(adapterTable.getRowCount());
        table = new JTable();
        table.setBounds(10, 11, 416, 241);
        contentPane.add(table);
    }
}
```

FactoryMethod diseinu partroia bets21Aplikazioan

UML DIAGRAMA



ALDATUTAKO KODEA

Kodean aldatu behar dugun zatiak Appliccation launcher eta BLFacadeFactory klase berrian izango dira. Orain arte aplication launcherreko main metodoan sortzen zen BLFacadea (product), bai online edo bai lokalean(Concrete product). Diseinu patroi hau implementatzeko BLFacadeFactory(Creator) klase berri bat sortu behar dugu, honek izango du main metodoan zegoen BLFacade klasea sortzen zuen kodea.

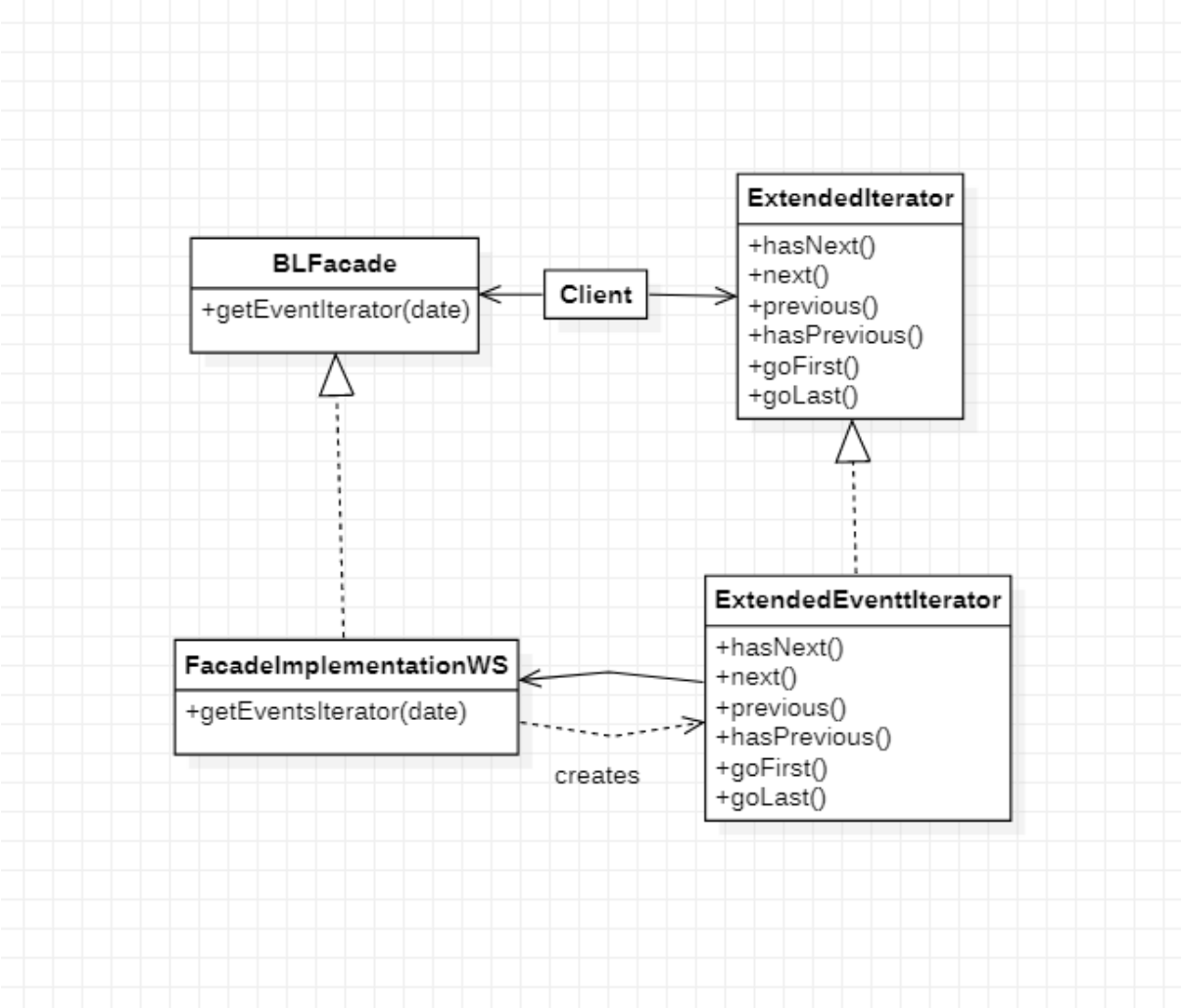
```
public  BLFacade createBLFacade(ConfigXML c) throws MalformedURLException {


        BLFacade  appFacadeInterface;

        if (c.isBusinessLogicLocal()) {
            DataAccess da= new DataAccess(c.getDataBaseOpenMode().equals("initialize"));
            return new BLFacadeImplementation(da);
        }else{
            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/", "FacadeImplementationWSService");
            QName qname = new QName("http://businessLogic/", "BLFacadeImplementationService");

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

             appFacadeInterface = service.getPort(BLFacade.class);
             return appFacadeInterface;
        }
}
```

Application launcherreko main metodoan BLFacadeFactori klaseko instantzia bat sortu behar dugu eta berarkin createBLFacade metodoari deitu ConfigXML ren intantzia (c aldagaia) parametro bezala pasaz.

```
public static void main(String[] args) {

    ConfigXML c=ConfigXML.getInstance();

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

    Locale.setDefault(new Locale(c.getLocale()));

    System.out.println("Locale: "+Locale.getDefault());

    MainGUI a=new MainGUI();
    a.setVisible(true);



    try {
        BLFacadeFactory blFacade = new BLFacadeFactory();

        MainGUI.setBussinessLogic(blFacade.createBLFacade(c));


    }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 diseinu partroia bets21Aplikazioan

UML DIAGRAMA



ALDATUTAKO KODEA

ExtendedIterator interfacia sortu dugu, iterator eta iterable zabaltzen dituena. Ondoren ExtendedEventIterator klasea sortu dugu ExtendedIterator zabaltzen duena. Iterator interfaceko hasNext eta next metodoak gainidazteaz gain, ExtendedIteratorean adierazitako hasPrevious, previous, goFirst eta goLast metodoak inplementatu ditugu. Horrek ongi funtzionatzen duela ziurtatzeko IteratorProba klasea sortu eta bertan dokumentuan emandako probak burutu ditugu.

Interfacea:

```java
package domain;

import java.util.Collection;
import java.util.Iterator;

public interface ExtendedIterator<Event> extends Iterator<Event>, Iterable{
    //uneko elementua itzultzen du eta aurrekora pasatzen da
    public Event previous();
    //true aurreko elementua existitzen bada.
    public boolean hasPrevious();
    //Lehendabiziko elementuan kokatzen da.
    public void goFirst();
    //Azkeneko elementuan kokatzen da.
    public void goLast();

    void add(Event ev);

    public boolean isEmpty();

    public int size();
}
```

Implementazioa:

```java
package domain;

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

public class ExtendedEventIterator implements ExtendedIterator<Event> {
    private int pos;
    private Vector<Event> array;

    public ExtendedEventIterator() {
        this.pos =0;
        this.array = new Vector<Event>();
    }

    public ExtendedEventIterator(Iterable<Event> array) {
        this.pos = 0;
        this.array = new Vector<Event>();
        for(Event e : array) {
            this.array.add(e);
        }
    }

    @Override
    public boolean hasNext() {
        return (pos >= 0 && pos < array.size());

        /*
        if(pos >array.size()-1) {
            return true;
        } else {
            return false;
        }*/
    }
```

```java
@Override
public Event next() {
    Event e = array.get(pos);
    pos++;
    return e;
}

@Override
public boolean hasPrevious() {
    return (pos >= 0 && pos < array.size());
}

@Override
public Event previous() {
    Event e = array.get(pos);
    pos--;
    return e;
}

@Override
public void goFirst() {
    pos = 0;
}

@Override
public void goLast() {
    pos = array.size()-1;
}

@Override
public void add(Event ev) {
    array.add(ev);
}
```

```java
    @Override
    public Iterator iterator() {
        return this;
    }

    @Override
    public boolean isEmpty() {
        return array.isEmpty();
    }

    @Override
    public int size() {
        return array.size();
    }
}
```

PROBA APLIKAZIOA

```java
package domain;

import java.net.MalformedURLException;

public class IteratorProba {
    public static void main(String[] args) {
        ConfigXML c=ConfigXML.getInstance();
        Locale.setDefault(new Locale(c.getLocale()));
        boolean isLocal=true;
        BLFacadeFactory blFacade = new BLFacadeFactory();
        BLFacade facade = null;
        try {
            facade = blFacade.createBLFacade(c);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        try {
            blFacade.createBLFacade(c);
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
        Date date = parseDate("2022-11-14");
        ExtendedIterator<Event> i=facade.getEventIterator(date);
        Event ev;
```

```java
        //Lista osoa errepasatzen du atzetik aurrera
        i.goLast();
        while (i.hasPrevious()){
            ev=i.previous();
            System.out.println(ev.toString());
        }

        //Lista aurretik atzera errepasatu
        while (i.hasNext()){
            ev=i.next();
            System.out.println(ev.toString());
        }

        //Lehen posiziora bueltatu
        i.goFirst();
    }
    public static Date parseDate(String date) {
         try {
             return new SimpleDateFormat("yyyy-MM-dd").parse(date);
         } catch (ParseException e) {
             return null;
         }
    }
}
```

Ez dugu proba aplikazioaren exekuzioa erakusten datu basea hasieratzean errorea ematen duelako. Datu basea ez da ondo ezabatzen.