Patroiak proiektua

Software Ingeniaritza II

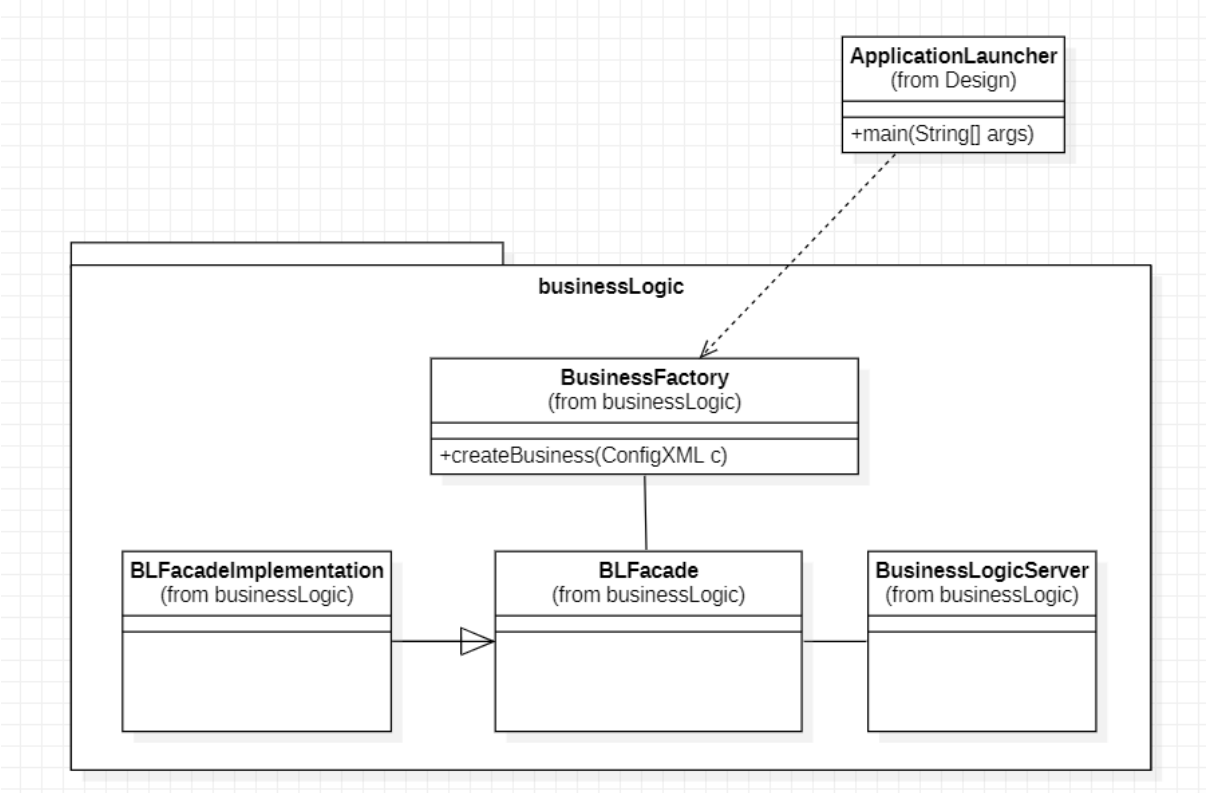
Asier Aizpurua
Juan Alagon
Xabier Artola

Factory Method Patroia

Eskatzen da:

Aplikazioa aldatu negozio logikako objektuaren lorpena faktoria objektu batean zentralizatuta egoteko, eta aurkezpenak zein negozio logikako inplementazio erabili erabaki dezatela. Diseina eta inplementatu ebazpena Creator, Product eta ConcreteProduct jokatzen duten klaseen rolak garbi aurkeztu

UML diagrama hedatua egin dituzun aldaketak aurkeztuz.



BusinessFactory klasea sortu da “Creator” bezala, BLFacade interfazea "Product" izango da eta BLFacadeImplementation/BusinessLogicServer "Concrete Product" izango dira. ApplicationLauncher klaseak main() metodoaren bitartez deituko du Factory-ren barruan dagoen createBusiness(ConfigXML c) metodoari. Hemen “local” edo “remote” motako zerbitzaria erabiltzea kudeatuko da, BLFacade inplementazioa konfiguratuko du emandako parametroen arabera eta honen instantzia bueltatuko du (remote edo local instantzia).

Aldatu duzun kodea, lerro garrantzitsuenak azalduz.

AplicationLauncher.java

```
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);

    BusinessFactory bf = new BusinessFactory();

    // LoginGUI lG = new LoginGUI();
    // lG.setVisible(true);

    try {
        BLFacade appFacadeInterface = null;
        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");
        appFacadeInterface = bf.createBusiness(c);
        /*
         * if (c.getDataBaseOpenMode().equals("initialize"))
         * appFacadeInterface.initializeBD();
         */
        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();

}
```

Lehenik BusinessFactory objektu bat instantziatu dut eta gero, factory-ren **createBusiness** metodoari deituta, BLFacade instantzia sortu dut emandako konfigurazioaren arabera. Hona errefaktorizatu egin da hasiera batean main() metodo honetan zegoen kodea:

BusinessFactory.java

```
public BLFacade createBusiness(ConfigXML c) throws Exception{

    BLFacade appFacadeInterface;

    if (c.isBusinessLogicLocal()) {

        // 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)

        DataAccess da = new DataAccess(c.getDataBaseOpenMode().equals("initialize"));
        appFacadeInterface = new BLFacadeImplementation(da);

    }

    else { // If remote

        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;
}
```

Metodoa bi zatitan banatzen da: “local”-eko zerbitzarian eta “remote”-ko zerbitzarian. Pasatako konfigurazioaren arabera "local" edo "remote" motako facade-a bueltatuko du. Factory-a errorea bueltatu dezake "remote" motako facade-a sortzean errorea agertu daitekelako.

Iterator patroia

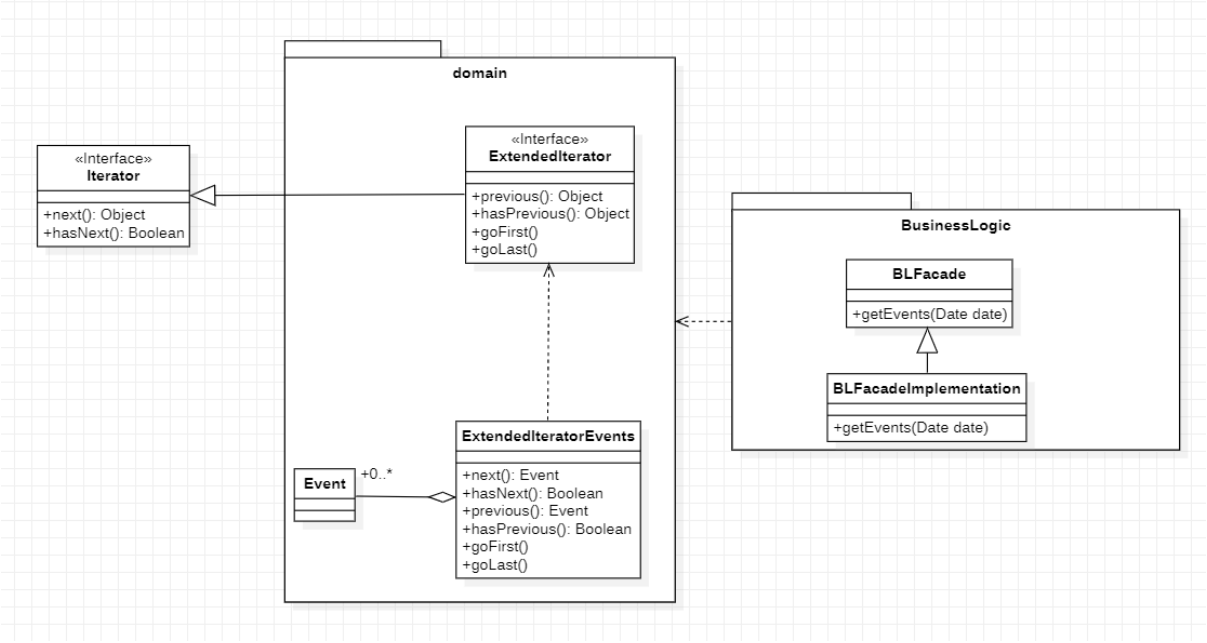
Eskatzen da:

Iteratzaile Hedatua inplementatu, eta adibidezko antzeko programa bat inplementatuz, gertaerak aurkeztutako ordenan inprimatu.

Jarraian zure aplikazioa aldatu, getEvents() modu berrian erabiltzeko.

UML diagrama hedatua egin dituzun aldaketak aurkeztuz.

Iterator klasetik abiatuta domain paketean, ExtendedIterator egongo da eta hau inplementatzen duen ExtendedIteratorEvents klaseako instantzia bat itzuliko du BLFacadeImplementation-eko getEvents metodoak.



Aldatu duzun kodea, lerro garrantzitsuenak azalduz

Enuntziatuak dioenari jarraituz, lehenik eta behin metodoaren signatura aldatu dut, zehatzago itzultzen duen objetu mota Vector<Event> motatik ExtendedIteratorEvents motara. Aldaketa hori hurrengo irudian ikus daiteke (BLFacade.java).

```
 */
@WebMethod public ExtendedIteratorEvents getEvents(Date date);
```

Aldaketa honek eragingo du, BLFacadeImplementation.java-n eguneraketa egitea interfazearen implementazioa bete dadin.

```
     */
    @WebMethod
    public ExtendedIteratorEvents getEvents(Date date) {

        dbManager.open(false);
        Vector<domain.Event> events = dbManager.getEvents(date);
        dbManager.close();

        return new ExtendedIteratorEvents(events);
    }
```

Metodoak aurreko signaturarekin egiten zuen gauza bera egiten du baina “event”-en bektorea itzuli beharrean, iteradore bat itzultzen du bektore hau zeharkatu ahal izateko.

Iteradoreen klaseei dagokienez bi aldaketa nagusi egin ditugu, Alde batetik Iterator interfazea luzatzen duen interfaze berri bat sortuz (ExtendedIterator.java) lau metodo berri dituena irudian ikus daitekeen moduan.

```
3  import java.util.Iterator;
4
5  public interface ExtendedIterator<Object> extends Iterator {
6      //uneko elementua itzultzen du eta aurrekora pasatzen da
7      public Object 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     }
```

Behin interfazea eraikita, inplementatuko duen klasea sortu dugu ExtendedIteratorEvents hain zuzen ere.

```java
package domain;

import java.beans.EventSetDescriptor;

@XmlAccessorType(XmlAccessType.FIELD)
public class ExtendedIteratorEvents implements ExtendedIterator<domain.Event> {

    Vector<domain.Event> events;
    int index = 0;

    public ExtendedIteratorEvents() {
        this.events = new Vector<domain.Event>();
    }

    public ExtendedIteratorEvents(Vector<domain.Event> events) {
        this.events = events;
    }

    @Override
    public boolean hasNext() {
        return index < events.size() - 1;
    }

    @Override
    public domain.Event next() {
        Event ev = events.get(index);
        index++;
        return ev;
    }

    @Override
    public domain.Event previous() {
        domain.Event ev = events.get(index-1);
        index--;
        return ev;
    }

    @Override
    public boolean hasPrevious() {
        return index > 0;

    }

    @Override
    public void goFirst() {
        index = 0;

    }

    @Override
    public void goLast() {
        index = events.size() - 1;
```

Proiektuan aldaketak GUI klaseetan egin behar izan ditugu adibidez azpiko irudian ageri den FindQuestionGUI.java klasean, non komentatutako lerroak ordezkatuak izan diren iteratorren erabilera egiten duten batzuekin.

```
138
139                     try {
140                         tableModelEvents.setDataVector(null, columnNamesEvents);
141                         tableModelEvents.setColumnCount(3); // another column added to allocate ev objects
142
143                         BLFacade facade = MainGUI.getBusinessLogic();
144
145 //                      Vector<domain.Event> events = facade.getEvents(firstDay);
146                         ExtendedIterator<domain.Event> i = facade.getEvents(firstDay);
147
148 //                      if (events.isEmpty())
149                         if (!i.hasNext())
150                             jLabelEvents.setText(ResourceBundle.getBundle("Etiquetas").getString("NoEvents") + ": "
151                                     + dateformat1.format(calendarAct.getTime()));
152                         else
153                             jLabelEvents.setText(ResourceBundle.getBundle("Etiquetas").getString("Events") + ": "
154                                     + dateformat1.format(calendarAct.getTime()));
155 //                      for (domain.Event ev : events) {
156                         while (i.hasNext()) {
157
158                             domain.Event ev = (domain.Event)i.next();
159
160                             Vector<Object> row = new Vector<Object>();
161
162                             System.out.println("Events " + ev);
163
164                             row.add(ev.getEventNumber());
165                             row.add(ev.getDescription());
166                             row.add(ev); // ev object added in order to obtain it with tableModelEvents.getValueAt(i,2)
167                             tableModelEvents.addRow(row);
168                         }
169                         tableEvents.getColumnModel().getColumn(0).setPreferredWidth(25);
170                         tableEvents.getColumnModel().getColumn(1).setPreferredWidth(268);
171                         tableEvents.getColumnModel().removeColumn(tableEvents.getColumnModel().getColumn(2)); // not
```

Exekuzioaren irudi bat

```
Atzetik aurrera:
12;Getafe-Celta
11;Eibar-Barcelona
10;Atlético-Athletic
Aurretik atzera:
10;Atlético-Athletic
11;Eibar-Barcelona
12;Getafe-Celta
```

Adapter Patroia

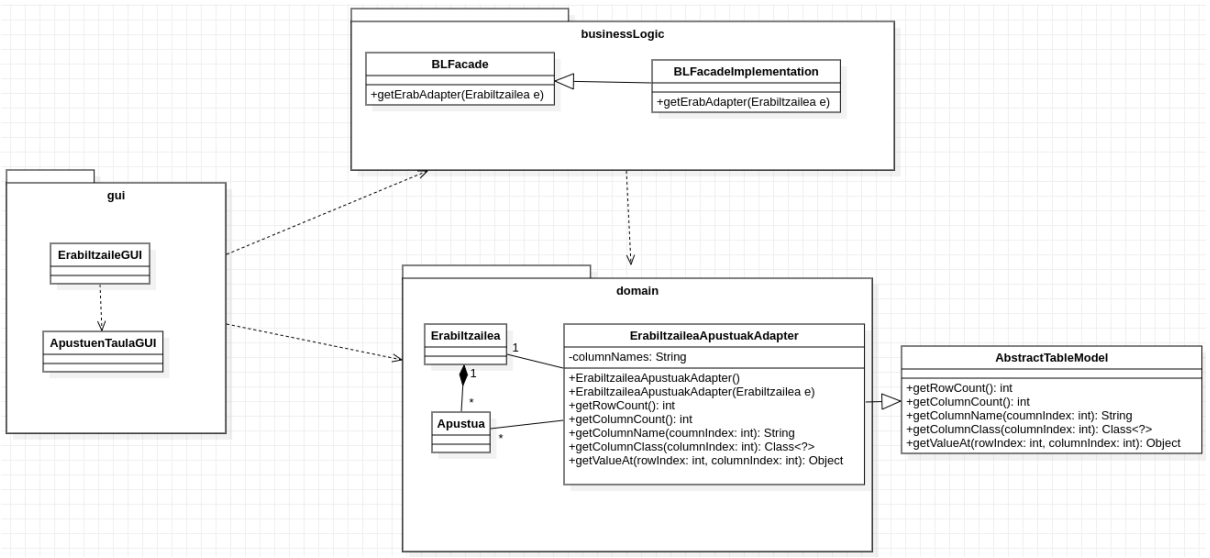
JTable batean Erabiltzaile batek egin dituen Apustu guztien informazioa aurkezten duen leiho berri bat sortu. Ohar: Ezin da Erabiltzaile klasea aldatu. Diseina eta inplementatu ebazpena.

UML diagrama hedatua egin dituzun aldaketak aurkeztuz.

Lehendabizi, egindako apustuen taula bistaratzeko GUI-a sortu dut, *ApustuenTaulaGUI* izenekoa, *ErabiltzaileGUI*-tik irekiko dena.

ApustuenTaulaGUI facadeko **getErabAdapter** metodoari deituko dio, honek *ErabiltzaileaApustuakAdapter* motako klasearen instantzia bat bueltatuko dio. Instantzia hau *JTable* baten “model” bezala erabiliko du.

ErabiltzaileaApustuakAdapter klasea *AbstractTableModel* klase abstraktua luzatzen du, horrela, *JTable* baten “model” bezala erabili ahalko dugu. Klase honi erabiltzaile bat pasata, honek egindako apustuak eskuragarri jarriko ditu *AbstractTableModel* interfazetik (**getValueAt** metodoa erabilita atzitu ahalko ditugu).



(*) Oharra: Klase diagraman ez dira agertzen “container” motako klaseak (ApustuaContainer, KuotaContainer, ErabiltzaileaContainer, …) inplementazioaren ondorioz agertu diren klaseak direlako, ez dira diseinuaren parte. Ondorioz, *ErabiltzaileaApustuakAdapter* eraikitzaileari UML-n *Erabiltzaile* klase bat pasatzen zaion arren, inplementazioan *ErabiltzaileContainer* pasatzen zaio, serializazioaren ondorioz agertzen diren arazoak konpontzen dituena.

Aldatu duzun kodea, lerro garrantzitsuenak azalduz.

ErabiltzaileaApustuakAdapter klasea sortu dut, *AbstractTableModel* klase abstraktua luzatzen duena. Klase honek erabiltzaile baten apustuak eskuragarri jartzen ditu *JTable* batek atzitu ahal izateko.

Oharra: Azkeneko bertsioan apustu anizkoitzak inplementatu ziren baino adibidean apustu soilak zirenez, apustu anizkoitzak soilak bezala tratatu ditugu, soilik lehenengo kuota kontuan hartuz gertaera eta galdera atzitzeko.

```java
@XmlAccessorType(XmlAccessType.FIELD)
public class ErabiltzaileaApustuakAdapter extends AbstractTableModel {
    private Erabiltzailea erabiltzailea;
    private List<ApustuaContainerLuzatuta> apustuak;
    private String[] columnNames = {"Event","Question","EventDate","Bet (€)"};

    public ErabiltzaileaApustuakAdapter() {
        this.erabiltzailea = null;
        this.apustuak = new ArrayList<>();
    }
    public ErabiltzaileaApustuakAdapter(ErabiltzaileaContainer erab) {
        this.erabiltzailea = erab.getE();
        this.apustuak = erab.getApustuak();
    }
    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        ApustuaContainerLuzatuta apCL = this.apustuak.get(rowIndex);
        Apustua ap = apCL.getApustua();

        QuestionContainer qC = apCL.getKuotak().get(0).getQuestion();
        Question q = qC.getQuestion();
        Event ev = qC.getEvent();

        switch(columnIndex) {
        case 0: // Event
            return ev.getDescription();
        case 1: // Question
            return q.getQuestion();
        case 2: // EventDate
            return ev.getEventDate();
        case 3: // Bet (money)
            return Double.toString(ap.getDiruKop());
        default:
            throw new IndexOutOfBoundsException(columnIndex);
        }
    }
    @Override
    public int getRowCount() { return apustuak.size(); }
    @Override
    public int getColumnCount() { return columnNames.length; }
    @Override
    public String getColumnName(int columnIndex) { return columnNames[columnIndex]; }
    @Override
    public Class<?> getColumnClass(int columnIndex) { return String.class; }
}
```

BLFacade interfazean **getErabAdapter** metodoa gehitu dut eta *BLFacadeImplementation* klasean inplementazioa egin det. Metodo honeri erabiltzaile bat pasata *ErabiltzaileaApustuakAdapter* klasearen instantzia bat bueltatuko du erabiltzaile honentzako. Metodo honek ere bermatzen du adapter-a eskuratzean, erabiltzailearen informazioa eguneratuta egongo dela DB-tik erabiltzailea atzituz.

Oharra: Serializazio arazoak konpontzeko “facade” adapter sortzea behar izan dut honen “container” sortzeko, gero bezeroari datu osoak iristeko (hau da, bidean serializazio arazoak direla eta, datuak ez direla galtzen bermatzeko)

```
@WebMethod ErabiltzaileaApustuakAdapter getErabAdapter(Erabiltzailea e);
```

```
@Override
public ErabiltzaileaApustuakAdapter getErabAdapter(Erabiltzailea e) {
    dbManager.open(false);
    Erabiltzailea eDB = dbManager.getErabiltzaileaIzenarekin(e.getIzena());
    dbManager.close();
    ErabiltzaileaContainer eC = new ErabiltzaileaContainer(eDB);
    return new ErabiltzaileaApustuakAdapter(eC);
}
```

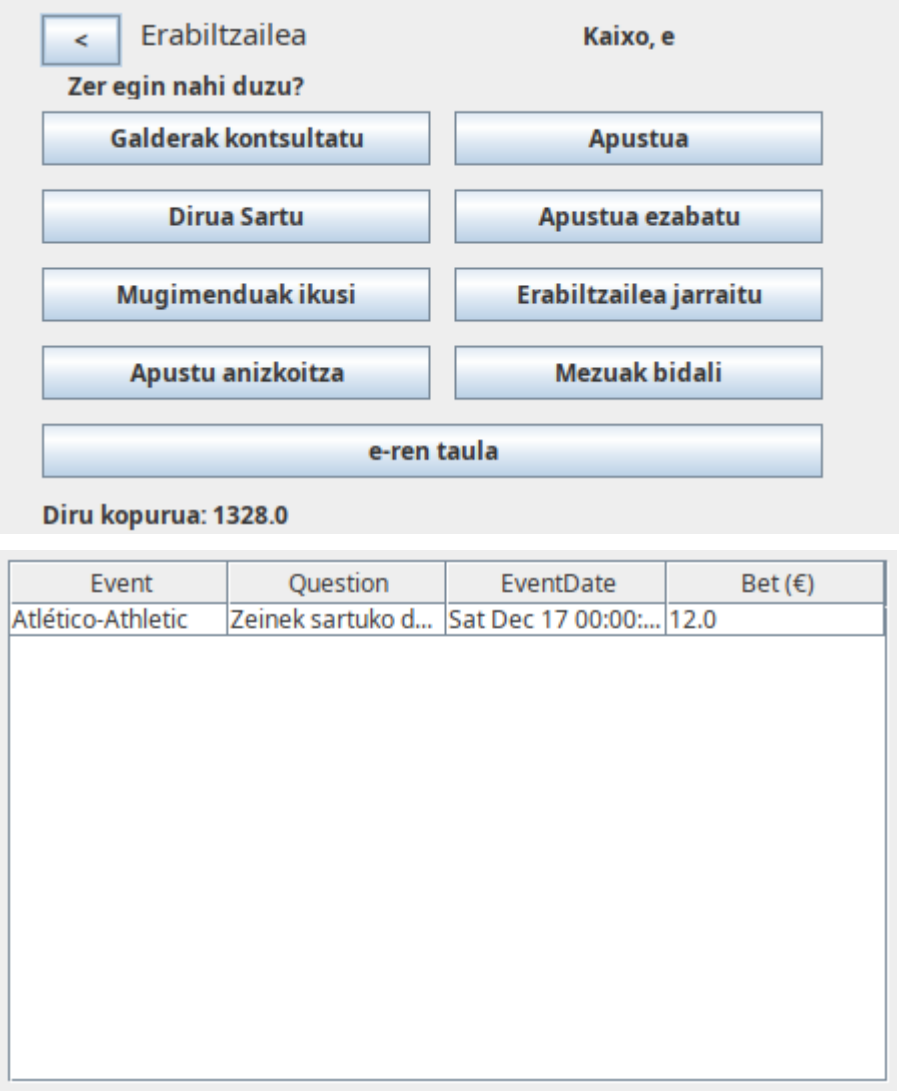
ErabiltzaileGUI klasean “Table of <user>” butoia gehitu det, uneko erabiltzaileak egindako apustuen taula ikusteko.

```
String tableOfUser = ResourceBundle.getBundle("Etiquetas").getString("ErabiltzaileGUI.btnTableOfUser.text");
tableOfUser = tableOfUser.replace("<user>", this.erabiltzailea.getIzena());
JButton btnTableOfUser = new JButton(tableOfUser); //$NON-NLS-1$ //$NON-NLS-2$
btnTableOfUser.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        ErabiltzaileaApustuakAdapter eAdapter = facade.getErabAdapter(erabiltzailea);
        JFrame apustuTaula = new ApustuenTaulaGUI(eAdapter);
        apustuTaula.setVisible(true);
    }
});
btnTableOfUser.setBounds(22, 215, 390, 27);
contentPane.add(btnTableOfUser);
```

ApustuenTaulaGUI klasea sortu dut, erabiltzaileak egindako apustuak bistaratzeko *JTable* elementu bat duena. GUI honek *ErabiltzaileaApustuakAdapter* klasearen instantzia bat jasoko du eraikitzailearen bidez eta *JTable* elementuari "model" bezala pasatuko dio.

```
public class ApustuenTaulaGUI extends JFrame {

    private JPanel contentPane;
    private JTable apustuakTaula;

    /**
     * Create the frame.
     * @param erabiltzaileaApustuakAdapter
     */
    public ApustuenTaulaGUI(ErabiltzaileaApustuakAdapter erabiltzaileaApustuakAdapter) {
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        contentPane.setLayout(new GridLayout(0, 1, 0, 0));

        JScrollPane apustuakPane = new JScrollPane();
        contentPane.add(apustuakPane);

        apustuakTaula = new JTable();
        apustuakTaula.setModel(erabiltzaileaApustuakAdapter);
        apustuakTaula.setFillsViewportHeight(true);
        apustuakPane.setViewportView(apustuakTaula);
    }
}
```

Exekuzioaren irudi bat



GitHub-eko proiekturako esteka

https://github.com/guskikalola/Bets21-SI2