

Patrones de diseño

Patrón Factory Method:

```
public class ApplicationLauncher {  
  
    public static void main(String[] args) {  
        int isLocal = 1;  
        BLFacade blFacade = new BLFactory().getBusinessLogicFactory(isLocal);  
    }  
}
```

ApplicationLauncher solo llama a la clase de tipo factoría *BLFactory*, que crea un elemento de tipo *BLFacade* local o remoto dependiendo del valor que le demos a *isLocal*.

```
public class BLFactory {  
  
    public BLFacade getBusinessLogicFactory(int isLocal) {  
        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(false);  
  
        MainUserGUI b = new MainUserGUI();  
        b.setVisible(true);  
        BLFacade appFacadeInterface= null;  
        try {  
            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");  
  
            if (isLocal==1) {  
                //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);  
            }  
        }  
    }  
}
```

```

    appFacadeInterface=new BLFacadeImplementation(ua);
}
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);
}
/*if (c.getDataBaseOpenMode().equals("initialize"))
    appFacadeInterface.initializeBD();
*/
MainGUI.setBusinessLogic(appFacadeInterface);
} catch (Exception e) {
    a.jLabelSelectOption.setText("Error: "+e.toString());
    a.jLabelSelectOption.setForeground(Color.RED);

    System.out.println("Error in ApplicationLauncher: "+e.toString());
}
return appFacadeInterface;

```

BLFactory se encarga de ejecutar el código que antes teníamos en *ApplicationLauncher*, con ello crea un elemento *BLFacade* de tipo local o remoto dependiendo del valor *isLocal* que le demos, y nos devuelve dicho elemento.

Patrón Iterator:

Para realizar el patron iterator primero hemos creado una interface llamada *ExtendedIterator<Event>* que es hija de la clase *Iterator<Object>* , aqui se definen los métodos *next*, *previous*, *hasPrevious*, *goFirst* y *GoLast*

```

package businessLogic;

import java.util.*;

public interface ExtendedIterator<Event> extends Iterator<Object>{

    public Event next();

    //return the actual element and go to the previous
    public Object previous();

    //true if ther is a previous element
    public boolean hasPrevious();

    //It is placed in the first element
    public void goFirst();

    // It is placed in the last element
    public void goLast();

}

```

Después hemos creado la clase que implementa la interface que hemos creado arriba, esta clase tiene una lista y un integer que nos indica la posición en la que se encuentra la lista, los métodos *hasnext* comprueba si el índice está en la última posición del array, si este ha llegado al final devolverá false y si no devolverá true *next* simplemente devuelve el elemento actual y suma 1 al indice lo mismo con *previous* solo que este resta 1 al indice, *hasprevious* comprueba que el indice es

mayor o igual a 0. Gofirst da el valor 0 al índice y golast guarda la última posición del array en el índice

```
package businessLogic;

import java.util.*;

public class ExtendedIteratorEvents implements ExtendedIterator<Event>{

    List<Event> eventos;
    int posicion = 0;

    public ExtendedIteratorEvents (List<Event> eventos)
    {
        this.eventos = eventos;
    }

    public boolean hasNext() {

        return posicion < eventos.size();
    }

    public Event next() {
        Event evento =eventos.get(posicion);
        posicion = posicion + 1;

        return evento;
    }

    public Event previous() {

        Event terminado = eventos.get(posicion);
        posicion = posicion-1;

        return terminado;
    }

    public boolean hasPrevious() {
        if(posicion>=0) {
            return true;
        }
        return false;
    }

    public void goFirst() {

        if(eventos.size()>0)
            posicion=0;
    }

    public void goLast() {
        if(eventos.size()>0)
            posicion=eventos.size()-1;
    }

}
```

Para ejecutar la prueba hemos creado una nueva clase que hemos llamado prueba y este ha sido el resultado:

| | RECORRIDO | HACIA | ATRÁS |
|-----|-------------------------------|-------|-------|
| 27; | Djokovic-Federer | | |
| 24; | Miami Heat-Chicago Bulls | | |
| 23; | Atlanta Hawks-Houston Rockets | | |
| 22; | LA Lakers-Phoenix Suns | | |
| 10; | Betis-Real Madrid | | |
| 9; | Real Sociedad-Levante | | |
| 8; | Girona-Leganes | | |
| 7; | Malaga-Valencia | | |
| 6; | Las Palmas-Sevilla | | |
| 5; | Espanol-Villareal | | |
| 4; | Alaves-Deportivo | | |
| 3; | Getafe-Celta | | |
| 2; | Eibar-Barcelona | | |
| 1; | Atletico-Athletic | | |

| | RECORRIDO | HACIA | ADELANTE |
|-----|-------------------------------|-------|----------|
| 1; | Atletico-Athletic | | |
| 2; | Eibar-Barcelona | | |
| 3; | Getafe-Celta | | |
| 4; | Alaves-Deportivo | | |
| 5; | Espanol-Villareal | | |
| 6; | Las Palmas-Sevilla | | |
| 7; | Malaga-Valencia | | |
| 8; | Girona-Leganes | | |
| 9; | Real Sociedad-Levante | | |
| 10; | Betis-Real Madrid | | |
| 22; | LA Lakers-Phoenix Suns | | |
| 23; | Atlanta Hawks-Houston Rockets | | |
| 24; | Miami Heat-Chicago Bulls | | |
| 27; | Djokovic-Federer | | |


```
package businessLogic;

import domain.*;

public class Prueba {

    public static void main(String[] args) {
        // obtener el objeto Facade local
        int isLocal = 1;
        BLFacade blFacade = new BLFactory().getBusinessLogicFactory(isLocal);
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        Date date;
        try {
            date = sdf.parse("17/12/2023"); // 17 del mes que viene
            ExtendedIterator<Event> i = blFacade.getEventsIterator(date);
            Event e;
            System.out.println(" ");
            System.out.println("RECORRIDO HACIA ATRÁS");
            i.goLast(); // Hacia atrás
            while (i.hasPrevious()) {
                e = (Event) i.previous();
                System.out.println(e.toString());
            }
            System.out.println();
            System.out.println(" ");
            System.out.println("RECORRIDO HACIA ADELANTE");
            i.goFirst(); // Hacia adelante
            while (i.hasNext()) {
                e = i.next();
                System.out.println(e.toString());
            }
        } catch (ParseException e1) {
            System.out.println("Problems with date?? " + "17/12/2020");
        }
    }
}
```

Patrón Adapter:

 Registered

—

□

×

Select Option

Query Questions

Enter money

Bet

Remove bet


See movements

Ranking

Apustuak ikusi

Featured

Close session

 Apuestasrealizadaspormikel:

—

□

×

| Event | Question | Date | Bet(€) |
|-----------------------|-------------------------|--------------------------|--------|
| Real Madrid-Barcelona | Emaitza? | Nov 10, 2023, 4:23:20 PM | 2.5 |
| Atletico-Athletic | Who will win the match? | Nov 10, 2023, 4:23:20 PM | 1.3 |
| | | | |

Clase *RegisteredAdapter*:

```
//UserAdapter
public class RegisteredAdapter extends AbstractTableModel{

    private static final long serialVersionUID = 1L;
    protected Registered user;
    protected String[] columnNames = new String [] {"Event","Question","Date","Bet(€)"};
    public RegisteredAdapter(Registered user) {
        this.user=user;
    }

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

    public int getRowCount() {
        System.out.println(user.getUsername());
        return user.getApustuAnitzak().size();
    }

    public int getColumnCount() {
        return 4;
    }

    public String getValueAt(int rowIndex, int columnIndex) {

        Apustua bet = getBetInTable(rowIndex);
        if(columnIndex==0) {
            return bet.getKuota().getQuestion().getEvent().getDescription();
        }else if(columnIndex==1) {
            return bet.getKuota().getQuestion().getQuestion();
        }else if(columnIndex==2) {
            return bet.getApustuAnitza().getData().toLocaleString();
        }else if(columnIndex==3) {
            return bet.getKuota().getQuote().toString();
        }else {
            return null;
        }
    }

    public Apustua getBetInTable(int rowIndex) {
        int totalBets = 0;

        for (ApustuAnitza apustuAnitza : user.getApustuAnitzak()) {
            for (Apustua apustua : apustuAnitza.getApustuak()) {
                if (totalBets == rowIndex) {
                    return apustua;
                }
                totalBets++;
            }
        }

        return null;
    }
}
```

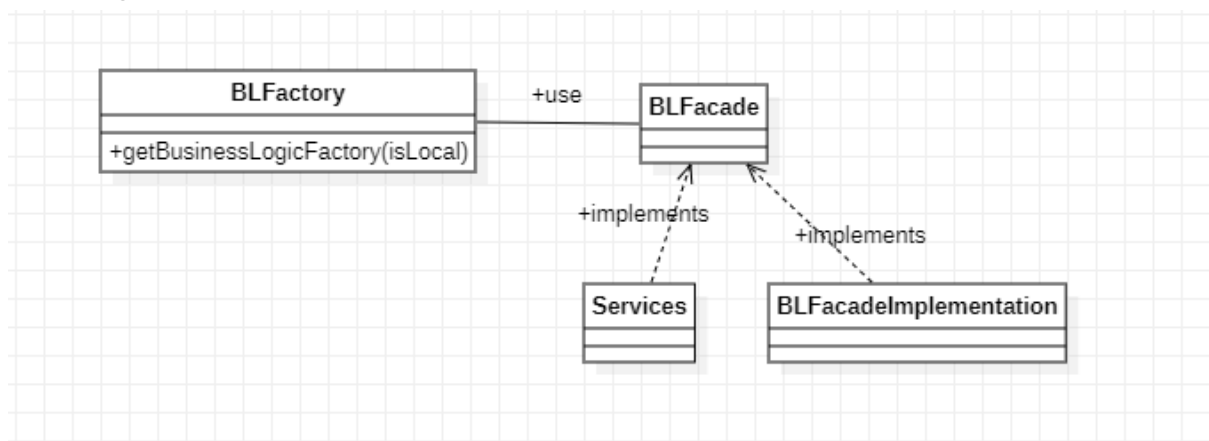
La clase *RegisteredAdapter* adapta la información de un usuario para que se pueda mostrar en un *JTable*. La clase *Registered* que representa los usuarios tiene una lista de apuestas múltiples, en vez de apuestas normales, así que para acceder a una apuesta individual

hemos creado una función auxiliar llamada *getBetInTable* que nos devuelve la apuesta que corresponde a la posición de fila *rowIndex*.

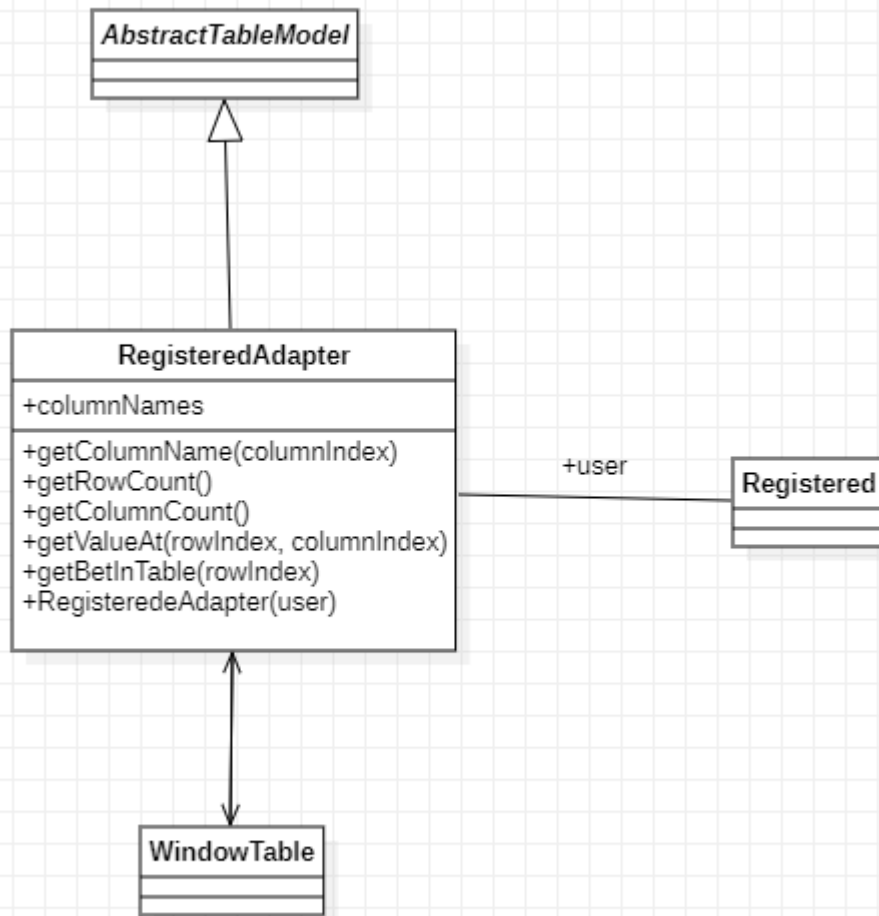
Clase *WindowTable*:

```
public class WindowTable extends JFrame {  
  
    private Registered user;  
    private JTable tabla;  
  
    public WindowTable(Registered user){  
        super("Apuestas realizadas por " + user.getUsername()+":");  
        this.setBounds(100, 100, 700, 200);  
        this.user = user;  
  
        RegisteredAdapter adapt = new RegisteredAdapter(user);  
        tabla = new JTable(adapt);  
  
        tabla.setPreferredSize(new Dimension(500, 70));  
        //Creamos un JScrollPane y le agregamos la JTable  
        JScrollPane scrollPane = new JScrollPane(tabla);  
        //Agregamos el JScrollPane al contenedor  
        getContentPane().add(scrollPane, BorderLayout.CENTER);  
    }  
}
```

UML Factory



Adapter



Iterator

