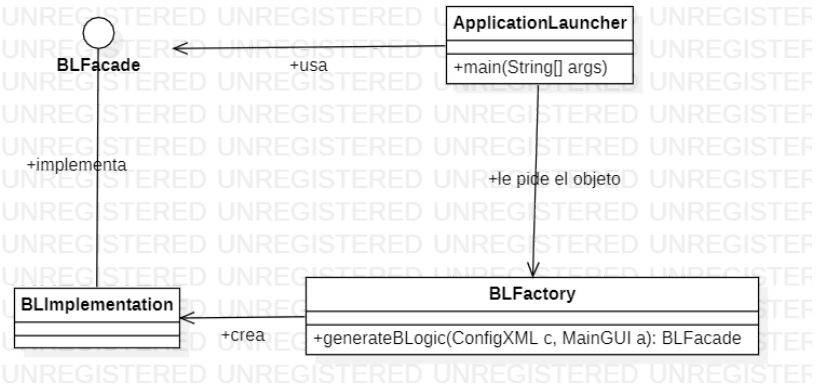
Patrones de diseño

Juan Navarlaz, Joseba Morate, Naiara Ortega

Patrón Factory

Diagrama UML



Cambios realizados

Clase añadida:

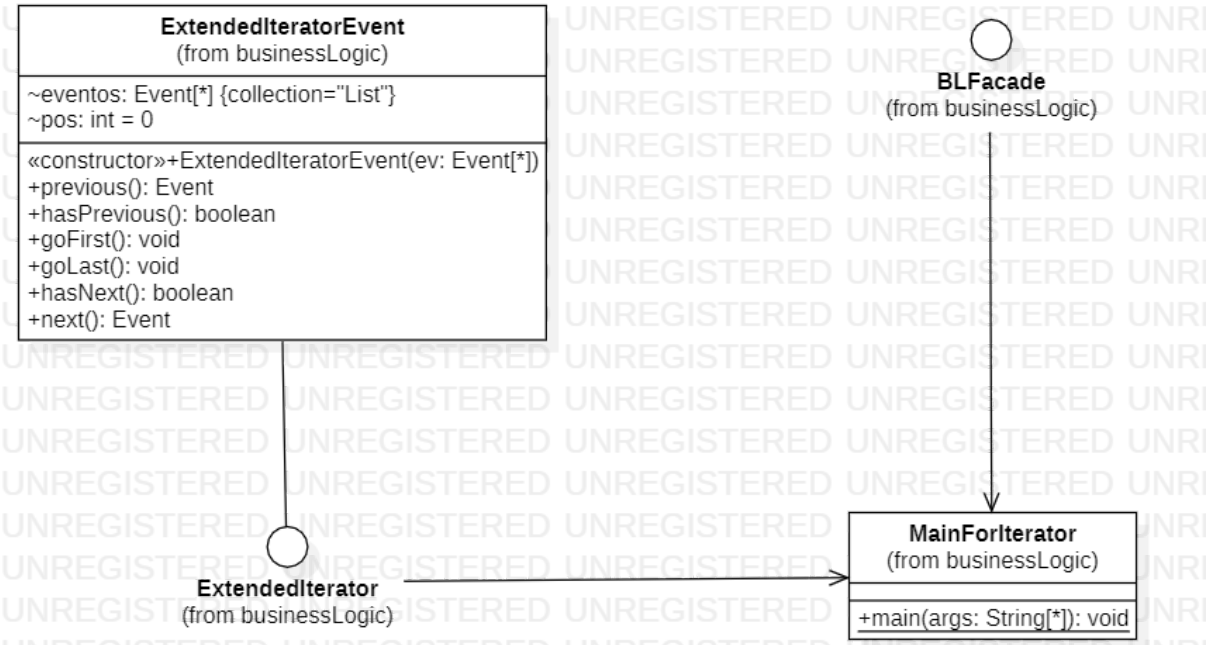
```
public class BLFactory {

    private BLFacade appFacadeInterface;

    public BLFactory() {

    }

    public BLFacade generateBLogic(ConfigXML c) throws Exception {
        BLFacade appFacadeInterface;
        if (c.isBusinessLogicLocal()) {
            DataAccess da = new DataAccess(c.getDataBaseOpenMode().equals("initialize"));
            appFacadeInterface = new BLFacadeImplementation(da);
        } else { // If remote
            String serviceName = "http://" + c.getBusinessLogicNode() + ":" + c.getBusinessLogicPo
                    + c.getBusinessLogicName() + "?wsdl";
            URL url = new URL(serviceName);
            QName qname = new QName("http://businessLogic/", "BLFacadeImplementationService");
            Service service = Service.create(url, qname);
            appFacadeInterface = service.getPort(BLFacade.class);
        }
        return appFacadeInterface;
    }

}
```

Cambios implementados:

```
public class ApplicationLauncher {


    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 {
        BLFacade appFacadeInterface; //product
        UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

        BLFactory fc= new BLFactory(); //Creator
        appFacadeInterface=fc.generateBLogic(c); //concrete Product (BLFacadeImplementation)

        MainGUI.setBussinessLogic(appFacadeInterface);

        }
```

Patrón Iterator

Diagrama UML



Cambios realizados

Clase main añadida:

```
public class MainForIterator {
    public static void main(String[] args)  {
        // obtener el objeto Facade local
        ConfigXML c=ConfigXML.getInstance();
        BLFacade blFacade;
        DataAccess da = new DataAccess(c.getDataBaseOpenMode().equals("initialize"));
        blFacade = new BLFacadeImplementation(da);
        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 = i.previous();
                System.out.println(e.toString());
            }
            System.out.println("fin recorrido hacia atras");
            System.out.println("_____________________");
            System.out.println("RECORRIDO HACIA ADELANTE");
            i.goFirst(); // Hacia adelante
            while (i.hasNext()) {
                e = i.next();
```

Clase Iterator e Interfaz añadida:

```java
public interface ExtendedIterator<Object> extends Iterator<Object> {

    //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();
}
```

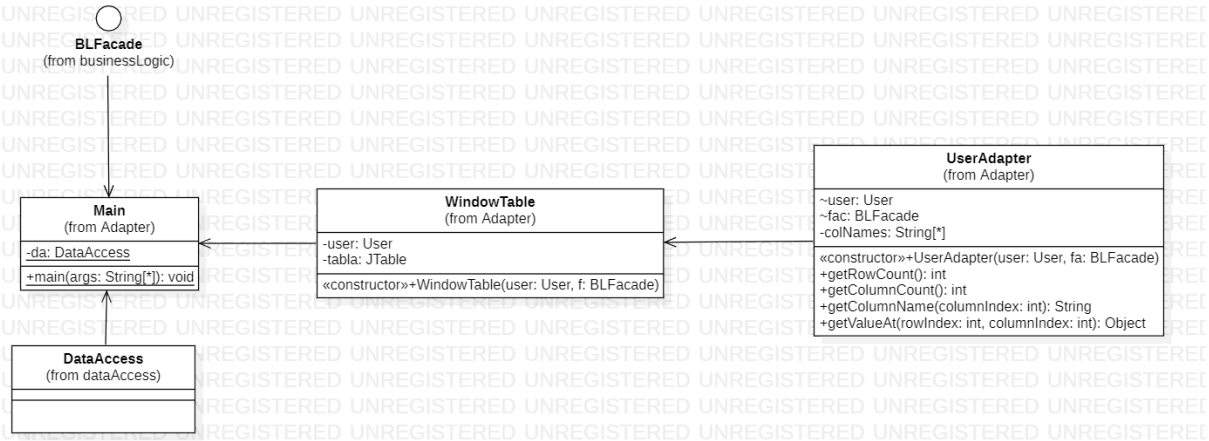
```java
public class ExtendedIteratorEvent implements ExtendedIterator{

    List<Event> eventos;
    int pos=0;

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

    //return    the    actual    element    and    go    to    the    previous
    public Event previous(){
        Event evento=eventos.get(pos);
        pos--;
        return evento;
    }

    //true    if ther    is    a    previous    element
    public boolean hasPrevious() {
        return pos>-1;
    }

    //It    is  placed  in  the first   element
    public void goFirst() {
        pos=0;
    }

    // It   is  placed  in  the last    element
    public void goLast() {
       pos=eventos.size()-1;
    }

    @Override
    public boolean hasNext() {
        return pos< eventos.size();
    }

    @Override
    public Event next() {
        Event evento =eventos.get(pos);
        pos ++;
        return evento;
    }

}
```

Foto de la ejecución

```
Db initialized
DataBase closed
Opening DataAccess instance => isDatabaseLocal: true getDatabBaseOpenMode: initialize
>> DataAccess: getEvents
1;Atl□tico-Athletic
2;Eibar-Barcelona
3;Getafe-Celta
4;Alav□s-Deportivo
5;Espanyol-Villareal
6;Las Palmas-Sevilla
7;Malaga-Valencia
8;Girona-Legan□s
9;Real Sociedad-Levante
10;Betis-Real Madrid
DataBase closed
_____________________
RECORRIDO HACIA ATRÁS
10;Betis-Real Madrid
9;Real Sociedad-Levante
8;Girona-Legan□s
7;Malaga-Valencia
6;Las Palmas-Sevilla
5;Espanyol-Villareal
4;Alav□s-Deportivo
3;Getafe-Celta
2;Eibar-Barcelona
1;Atl□tico-Athletic
fin recorrido hacia atras

_____________________
RECORRIDO HACIA ADELANTE
1;Atl□tico-Athletic
2;Eibar-Barcelona
3;Getafe-Celta
4;Alav□s-Deportivo
5;Espanyol-Villareal
6;Las Palmas-Sevilla
7;Malaga-Valencia
8;Girona-Legan□s
9;Real Sociedad-Levante
10;Betis-Real Madrid
fin recorrido hacia delante
```

Patrón Adapter

Diagrama UML



Cambios realizados

Clases añadidas:

```
public class WindowTable extends JFrame {
        private User user;
        private JTable tabla;

        public WindowTable(User user, BLFacade f) {
                super("Apuestas realizadas      por     " + user.getUserName() + ":");
                this.setBounds(100, 100, 700, 200);
                this.user = user;
                UserAdapter adapt = new UserAdapter(user, f);
                tabla = new JTable(adapt);
                tabla.setPreferredScrollableViewportSize(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);
        }
}
```

```java
public class UserAdapter extends AbstractTableModel{
	User user;
	BLFacade fac;
	private String[] colNames = new String[] {"Event", "Question","Event Date","Bet"};
	public UserAdapter(User user, BLFacade fa) {
		this.user=user;
		this.fac=fa;
	}
	@Override
	public int getRowCount() {
		return user.getVecApuestas().size();
	}

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

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

```java
	@Override
	public Object getValueAt(int rowIndex, int columnIndex) {
		// TODO Auto-generated method stub
		switch (columnIndex) {
		case 0:
			return "Eibar-Barcelona";

		case 1:
			int qNum1= user.getVecApuestas().get(rowIndex).getQNum();
			String question =fac.getQuestionByNum(qNum1+1).getQuestion();
			return question;
		case 2:
			return "17/12/2023";
		case 3:
			return user.getVecApuestas().get(rowIndex).getCantidad();
		}
		return null;
	}

}
```

```java
public class Main {
	private static DataAccess da;
	public static void main(String[] args) {
		try {
			ConfigXML c = ConfigXML.getInstance();
			BLFacade blFacade;
			da = new DataAccess(c.getDataBaseOpenMode().equals("initialize"));
			blFacade = new BLFacadeImplementation(da);
			User user = blFacade.getUserByName("Juan");
			WindowTable vt = new WindowTable(user, blFacade);
			vt.setVisible(true);
		} catch (Exception e) {
			e.printStackTrace();
		}finally {
			da.close();
		}
	}

}
```

Foto de la ejecución