Parte 3 - Patrones

Proyecto BETS - Ingeniería del Software II - UPV/EHU

14 de Noviembre del 2023

Nicolás Aguado

Xiomara Cáceres

Eider Fernández

Dirección del Proyecto

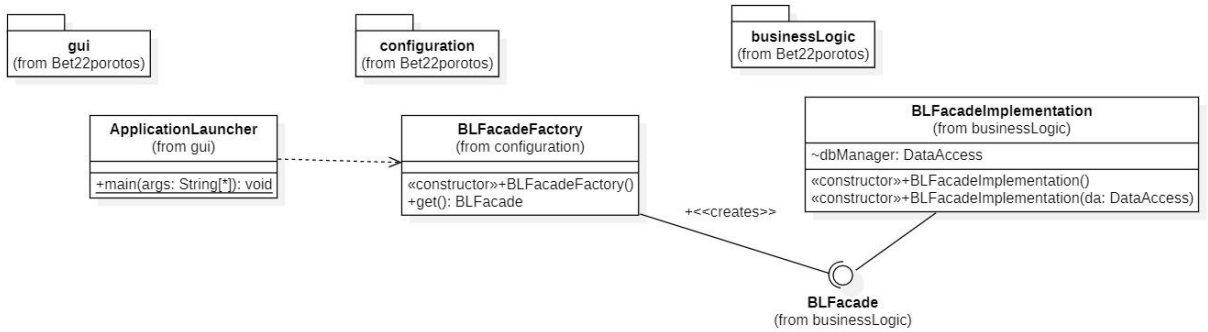
El proyecto de github del grupo se puede encontrar en la siguiente dirección:

https://github.com/por0tos/Bet22porotos

Horas de dedicación grupales: 8h

Patrón Factory

[Enlace al commit]



[Enlace al archivo .mdj]

Se ha implementado el patrón Factory de manera que el método main ha quedado así:

```
BLFacade appFacadeInterface = new BLFacadeFactory().get();

UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

MainGUI.setBussinessLogic(appFacadeInterface);
```

La creación de una lógica de negocio estará delegada a la clase BLFacadeFactory, y será ésta la que haga las comprobaciones pertinentes (comprobar el valor xml). Además, creará el objeto de tipo BLFacade de acuerdo a las descripciones.

En lo correspondiente al patrón Factory, la parte “Creadora” (Creator) se corresponde a la clase BLFacadeFactory, la parte “Producto” (Product) se corresponde a la clase BLFacade y la parte “Producto Concreto” (Concrete Product) se corresponde a la clase BLFacadeImplementation

```java
public class BLFacadeFactory {

    ConfigXML c;

    public BLFacadeFactory() {
        c=ConfigXML.getInstance();
        System.out.println(c.getLocale());
        Locale.setDefault(new Locale(c.getLocale()));
        System.out.println("Locale: "+Locale.getDefault());
    }

    public BLFacade get() throws MalformedURLException {
        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"));
             return new BLFacadeImplementation(da);
        }
        else {
            //If remote
             String serviceName= "http://"+c.getBusinessLogicNode() +":"+ c.getBusinessLogicPort()+"/ws/"+c.getBusinessLogicName()+"?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/", "BLFacadeImplementationService");

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

            return service.getPort(BLFacade.class);
        }
    }

}
```

Patrón Iterator

[Enlace al commit]

Descripción

Para facilitar que se recorra una colección de eventos, se ha creado la clase "ExtendedIterator". Esta clase permitirá recorrer una lista de eventos tanto hacia atrás como hacia delante. Para crearla, se recogerá un elemento de tipo "List" y se proveerán todos los métodos típicos de un iterador. [Código]

Entonces, cabe destacar que en el método getEvents(...) se estaba devolviendo un Vector de eventos. Pero claro, un vector no es una estructura que se pueda recorrer con un iterador. Por lo tanto, se ha creado el método getEventsIterator(...) que dada una fecha devolverá el iterador de los eventos en esa fecha, usando la nueva clase "ExtendedIterator" para ello.

```
public ExtendedIterator<Event> getEventsIterator(Date date)  {
    dbManager.open(false);
    Vector<Event> events=dbManager.getEvents(date);
    dbManager.close();
    ArrayList<Event> arr = new ArrayList<Event>();
    for (Event e : events) {
        arr.add(e);
    }
    return new ExtendedIterator<Event>(arr);
}
```

Ejecución

Para probar la ejecución de esta nueva clase se ha añadido un programa de pruebas dentro de /src/test/java/test.businesslogic/TestEventIterator.java

```
public class TestEventIterator {
    public static void main(String[] args) {
        // Queremos objeto date del día 17 del proximo mes
        Calendar calendar = Calendar.getInstance();
        calendar.set(Calendar.MONTH, calendar.get(Calendar.MONTH) + 1);
        calendar.set(Calendar.DAY_OF_MONTH, 17);
        calendar.set(Calendar.HOUR_OF_DAY, 0);
        calendar.set(Calendar.MINUTE, 0);
        calendar.set(Calendar.SECOND, 0);
        calendar.set(Calendar.MILLISECOND, 0);
        Date nextMonth17thDate = calendar.getTime();


        try {
            BLFacade bl = new BLFacadeFactory().get();
            ExtendedIterator<Event> i = bl.getEventsIterator(nextMonth17thDate);
            Event e;
            System.out.println("---------------");
            System.out.println("hacia atras");
            System.out.println("---------------");
            i.goLast();
            while (i.hasPrevious()) {
                e = i.previous();
                System.out.println(e.toString());
            }
            System.out.println("---------------");
            System.out.println("hacia delante");
            System.out.println("---------------");
            i.goFirst();
            while (i.hasNext()) {
                e = i.next();
                System.out.println(e.toString());
            }

        } catch(Exception e) {
            System.out.println(e.getMessage());
        }

    }
}
```

Y después de ejecutar este programa conseguimos los siguientes resultados por consola:

```
---------------
hacia atras
---------------
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
---------------
hacia delante
---------------
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
```

Patrón Adapter

[Enlace al commit]

Este ejercicio se plantea desde un enfoque práctico. El usuario desea consultar una lista de apuestas realizadas, pero el desarrollador tiene estos datos esparcidos por la base de datos en distintos lugares. Entonces, para poder acceder a la información correcta y sin modificar ninguna de las clases originales, se ha creado un “Adaptador” (y lo hemos llamado UserTableAdapter.java) que obtenía desde un objeto “Registered” (un usuario) toda la información necesaria para visualizar los datos relativos a las apuestas en una tabla de tipo “AbstractTableModel” (proveniente de la librería JSwing).

```
public Object getValueAt(int rowIndex, int columnIndex) {
    String res;
    switch(columnIndex) {
        case 0:
            res = apuestas.get(rowIndex).getKuota().getQuestion().getEvent().getDescription();
            break;
        case 1:
            res = apuestas.get(rowIndex).getKuota().getQuestion().getQuestion();
            break;
        case 2:
            res = apuestas.get(rowIndex).getKuota().getQuestion().getEvent().getEventDate().toString();
            break;
        case 3:
            res = dineros.get(rowIndex).toString();
            break;
        default:
            res = "ERR";
    }
    return res;
}

public UserTableAdapter(Registered u) {
    this.user = u;
    this.apuestas = new ArrayList<>();
    this.dineros = new ArrayList<>();
    for (ApustuAnitza a : u.getApustuAnitzak()) {
        for (Apustua ap : a.getApustuak()) {
            this.apuestas.add(ap);
            this.dineros.add(a.getBalioa());
        }
    }
}
```

Además, se ha creado una nueva interfaz (UserBetsGUI.java) para poder representar estos datos en una tabla

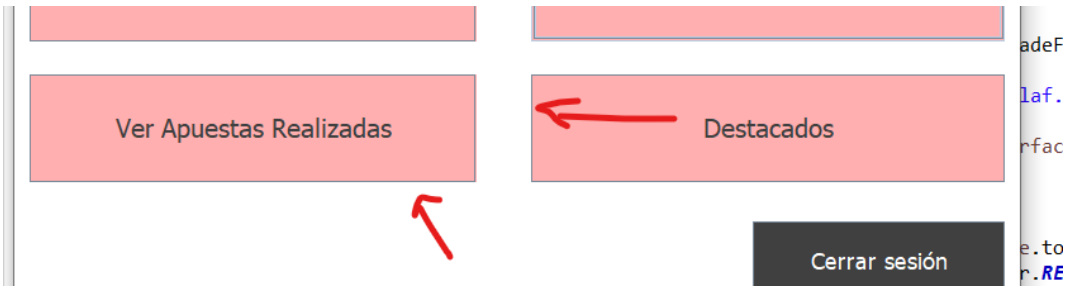
```
public class UserBetsGUI extends JFrame {

    private static final long serialVersionUID = 1L;
    private Registered user;
    private JTable tabla;

    public UserBetsGUI(Registered usuario) {
        super("Apuestas realizadas por "+usuario.getUsername());
        this.user = usuario;
        this.setBounds(100, 100, 700, 200);
        // Adapter (AbstractTableModel -> new UserTableAdapter(usuario))
        tabla = new JTable(new UserTableAdapter(usuario));
        tabla.setPreferredScrollableViewportSize(new Dimension(500, 70));
        JScrollPane sc = new JScrollPane(tabla);
        getContentPane().add(sc, BorderLayout.CENTER);
    }

}
```

Y se ha añadido un botón al menú de usuario (RegisteredGUI.java)



De tal manera que al hacer clic en la interfaz se abriese la tabla que obtenía los datos de nuestra clase adaptador.