eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Patterns

Software ingeniaritza

Joritz Arocena eta Mikel Carrasco

25-26 ikasturtea

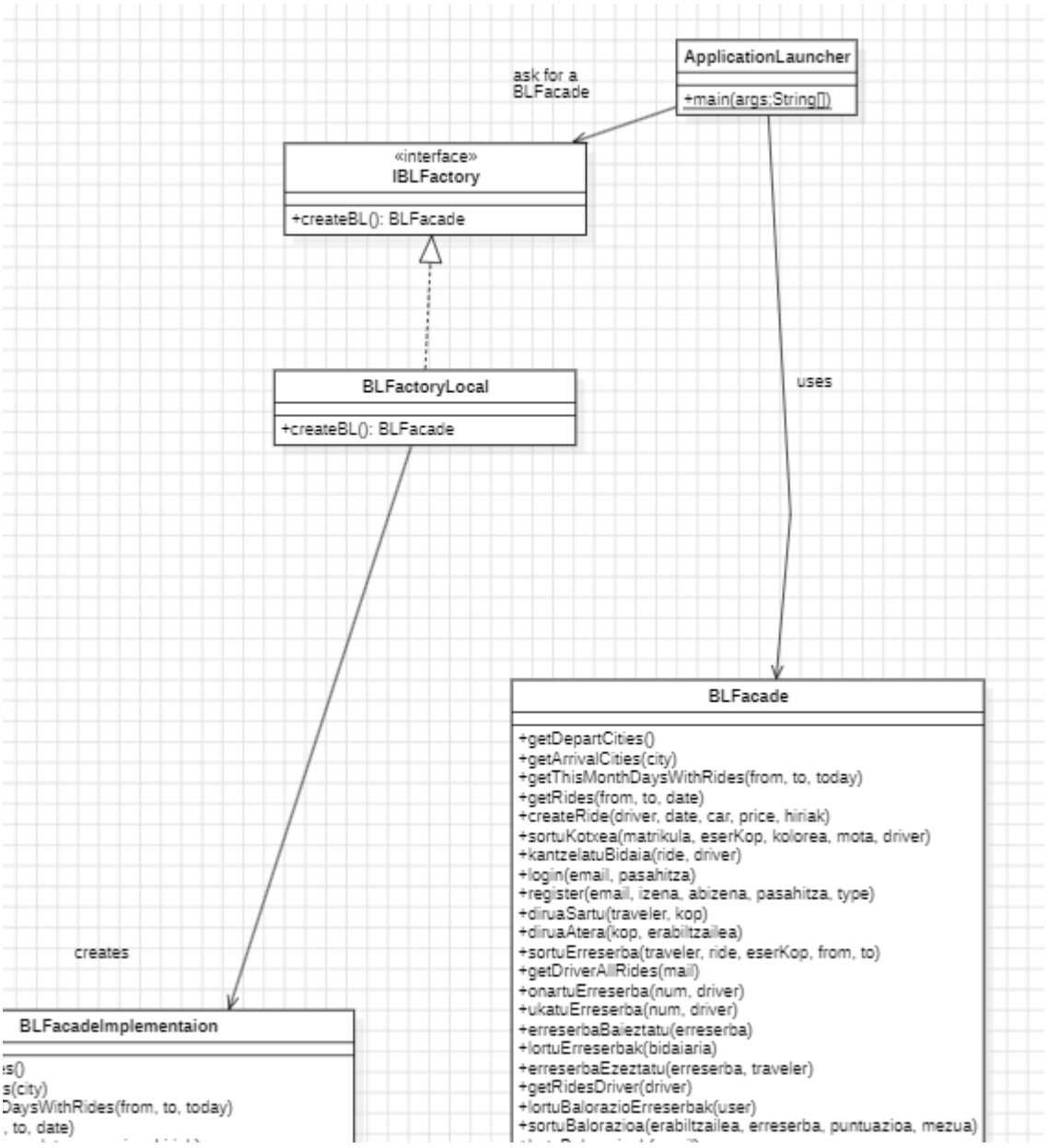
AURKIDEA

1. Sarrera

Dokumentu honetan proiektuan diseinu patroiak nola implementatu diren adierazten da. Kodeari buruzko informazio gehiago honako esteka honetan: https://github.com/Carrascomikel/Rides25

2. Factory method patroia

a)UML diagrama hedatua



b)Kodeko aldaketa nagusiak

Factory method patroia jarraitu ahal izateko lehendabizi IBLFactory interfaze bat sortu dugu eta bertan createBL metodoa sortu dugu ondoren negozio logika bat sortzerakoan metodo hau erabili ahal izateko. Hona hemen interfazearen kode zatia:

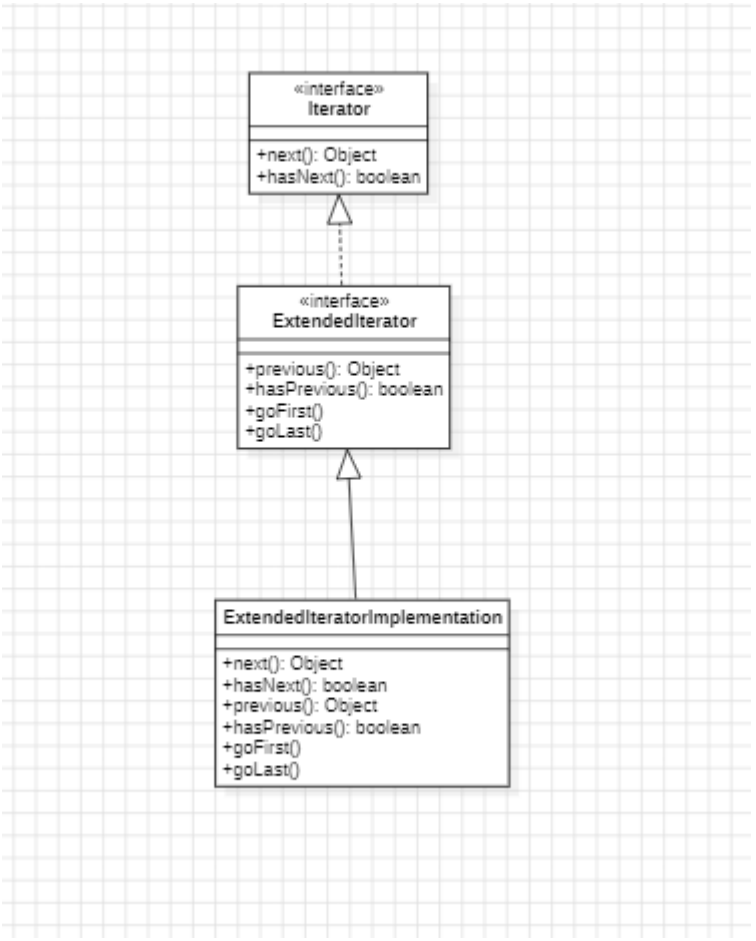
Ondoren interfazea hori inplementatzen duen beste klase bat sortu dugu eta bertan createBL metodoa inplementatu dugu, negozio logikaren sorrera lortzeko. Hona hemen klase honen kodea:

Eta azkenik AplicationLauncher klasean appFacadeInterface atributuari negozio logikaren balioa ematerakoan sortu berri ditugun klaseak eta metodoak erabiliko ditugu.Hona hemen kode zatia:

```
try {

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

    IBLFactory factory = new BLFactoryImplementation();

    appFacadeInterface = factory.createBL(c.isBusinessLogicLocal());
    appFacadeInterface=factory.createBL(c.isBusinessLogicLocal());
    MainGUI.setBussinessLogic(appFacadeInterface);
    MainGUI a = new MainGUI();
    a.setVisible(true);
```

3. Iterator patroia

a) UML diagrama hedatua



b) Kodeko aldaketa nagusiak

Patroi hau aplikatzeko daukagun getdepartingCities metodoa aldatu nahi dugu iteradore bat erabiliz inplementatzeko. Horretarako ExtendedIterator interfaze bat eta klase bat sortuko ditugu behar dituen metodak erazagutzeko. Hona hemen kode zatia:

```java
package domain;

import java.util.Iterator;

public interface ExtendedIterator<E> extends Iterator<E>{
    //return the actual element and go to the previous
    public E 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();
}
```

Ondoren BLFacade interfazean honako kodea zati hau jarriko dugu:

```java
@WebMethod
public ExtendedIterator<String> getDepartingCitiesIterator();
```

Ondoren metodo honen inplementazioa egingo dugu BLFacadeImplementation klasean eta honako kode hau idatziko dugu:

```java
@Override
public ExtendedIterator<String> getDepartingCitiesIterator() {
    return new ExtendedIteratorImplementation<String>(this.getDepartCities());
}
```

Azkenik Iterator main klasea inplementatuko dugu:

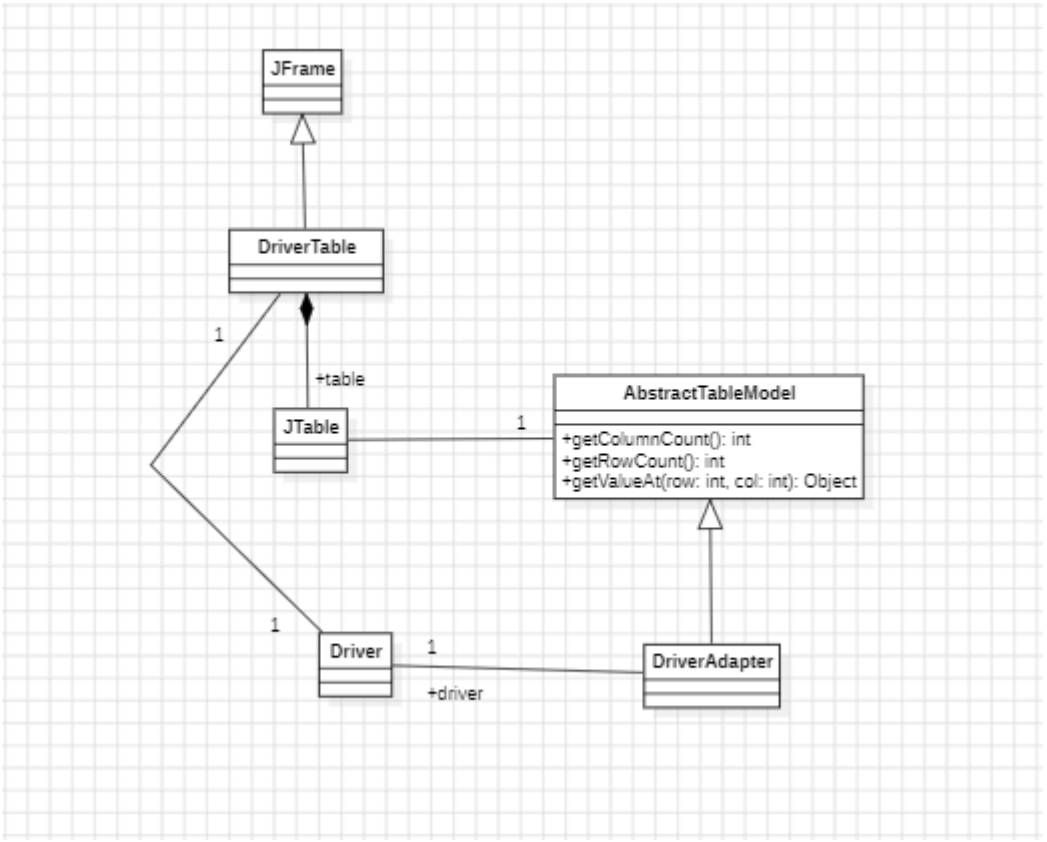
```java
public class IteratorMain {

    public static void main(String[] args) {
        ConfigXML c=ConfigXML.getInstance();

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

        try {

            BLFacade appFacadeInterface;
            IBLFactory factory = new BLFactoryImplementation();

            appFacadeInterface = factory.createBL(c.isBusinessLogicLocal());
            appFacadeInterface=factory.createBL(c.isBusinessLogicLocal());

            ExtendedIterator<String> i = appFacadeInterface.getDepartingCitiesIterator();
            String city;
            System.out.println("_____");
            System.out.println("FROM LAST TO FIRST");
            i.goLast();
            while(i.hasPrevious()) {
                city = i.previous();
                System.out.println(city);
            }
            System.out.println();
            System.out.println("_____");
            System.out.println("FROM FIRST TO LAST");
            i.goFirst();
            while(i.hasNext()) {
                city = i.next();
                System.out.println(city);
            }


        }catch (Exception e) {
            System.out.println("Error in ApplicationLauncher: "+e.toString());
        }


    }
}
```

c) Exekuzio irudia:

```
FROM LAST TO FIRST
donos
Madrid
Irun
Donostia
Barcelona


FROM FIRST TO LAST
Barcelona
Donostia
Irun
Madrid
donos
```

4. Adapter patroia

a)UML diagrama hedatua



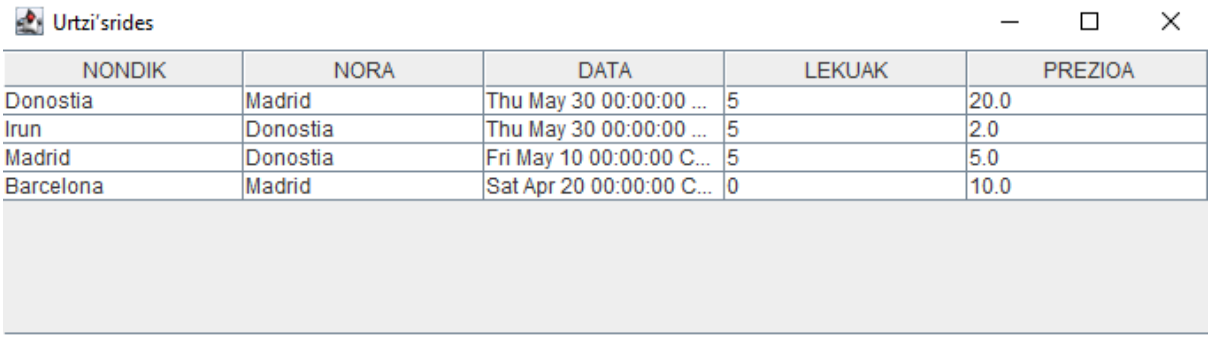
b) Kodeko zenbait aldaketa

DriverAdapter klasean hurrengo kodea inplementatu dugu:

```java
package domain;

import java.util.List;

import javax.swing.table.AbstractTableModel;

public class DriverAdapter extends AbstractTableModel {
    List<Ride> rideList;
    private final String[] columnNames = {"NONDIK", "NORA", "DATA", "LEKUAK", "PREZIOA"};

    public DriverAdapter(Driver d) {
        this.rideList = d.getCreatedRides();
    }

    @Override
    public int getRowCount() {
        return rideList.size();
    }

    @Override
    public int getColumnCount() {
        return 5;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        if(columnIndex==0) return this.rideList.get(rowIndex).getFrom();
        else if(columnIndex==1) return this.rideList.get(rowIndex).getTo();
        else if(columnIndex==2) return this.rideList.get(rowIndex).getDate();
        else if(columnIndex==3) return this.rideList.get(rowIndex).getnPlaces();
        else  return this.rideList.get(rowIndex).getPrice();
    }
      @Override
        public String getColumnName(int column) {
            return columnNames[column];
        }
}
```

Hemen ikusten den bezala metodoak hainbat metodo ditugu baina garrantzitsuena getValueAt metodoa da. Bertan nahi dugun zutabean behar dugun informazioa gehitzea du helburua. Klase hau ondoren beste batean erabiltzen da taula sortzeko eta hau eginda ondorengo emaitza lortu dugu.

c) Exekuzio irudia:



NONDIK	NORA	DATA	LEKUAK	PREZIOA
Donostia	Madrid	Thu May 30 00:00:00 ...	5	20.0
Irun	Donostia	Thu May 30 00:00:00 ...	5	2.0
Madrid	Donostia	Fri May 10 00:00:00 C...	5	5.0
Barcelona	Madrid	Sat Apr 20 00:00:00 C...	0	10.0