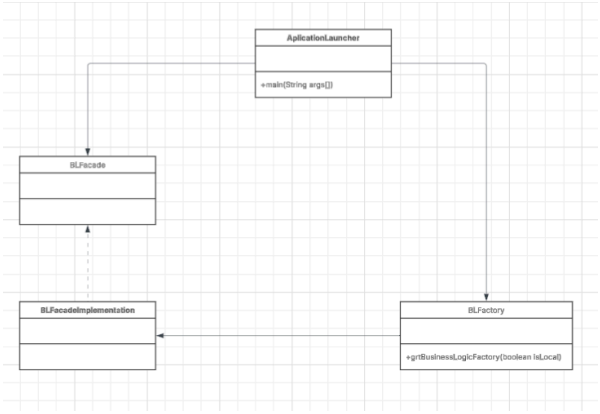
Rides diseinu patroiak

https://github.com/etxeg/Rides24

1. Factory Method patroia

UML



Aldatutako kodea

Hau zen hasieran geneukan kodea:

```
package gui;

import java.awt.Color;
import java.net.URL;
import java.util.Locale;

import javax.swing.UIManager;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;

import configuration.ConfigXML;
import dataAccess.DataAccess;
import domain.Driver;
import businessLogic.BLFacade;
import businessLogic.BLFacadeImplementation;

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());

        Driver driver=new Driver("driver3@gmail.com","Test Driver");


        MainGUI a=new MainGUI(driver);
        a.setVisible(true);


        try {

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

            if (c.isBusinessLogicLocal()) {

                DataAccess da= new DataAccess();
                appFacadeInterface=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);

                 appFacadeInterface = service.getPort(BLFacade.class);
            }

            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();


    }
}
```

BLFactory izeneko klase berri bat sortu da buisnessLogik paketean, modu honetan ApplikationLauncher klaseak soilik deiak egingo ditu eta erabakiak sortutako klase berrian egingo dira.

BLFactory:

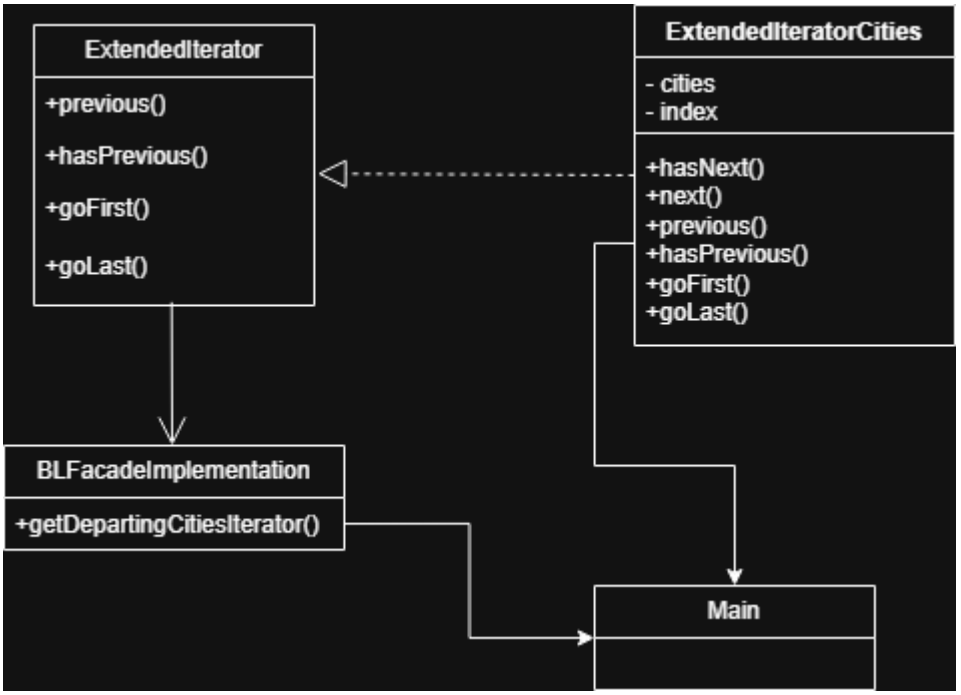
```
package businessLogic;

import java.net.URL;

public class BLFactory {
    public BLFacade getBusinessLogicFactory(boolean isLocal) {
        ConfigXML c = ConfigXML.getInstance();

        try {
            if (isLocal) {
                DataAccess da = new DataAccess();
                return new BLFacadeImplementation(da);
            } else {
                String serviceName= "http://"+c.getBusinessLogicNode() +":"+ c.getBusinessLogicPort()+"/ws/"+c.getBusinessLogicName()+"?wsdl";
                URL url = new URL(serviceName);
                QName qname = new QName("http://businessLogic/", "BLFacadeImplementationService");
                Service service = Service.create(url, qname);
                return service.getPort(BLFacade.class);
            }
        } catch (Exception e) {
            System.out.println("Error in BLFactory: " + e.toString());
            return null;
        }
    }
}
```

AplicationLauncher:

```
package gui;

import java.awt.Color;

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());

        Driver driver=new Driver("driver3@gmail.com","Test Driver","123");


        //a.setVisible(true);


        try {

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

            boolean isLocal = c.isBusinessLogicLocal();
            BLFacade appFacadeInterface = new BLFactory().getBusinessLogicFactory(isLocal);

            MainGUI a=new MainGUI();
            MainGUI.setBussinessLogic(appFacadeInterface);

            a.setVisible(true);
            //MainGUI a=new MainGUI(driver);



        }catch (Exception e) {
            /*a.jLabelSelectOption.setText("Error: "+e.toString());
            a.jLabelSelectOption.setForeground(Color.RED);  */

            System.out.println("Error in ApplicationLauncher: "+e.toString());
        }
        //a.pack();


    }

}
```

2. Iterator patroia

UML



Aldatutako kodea

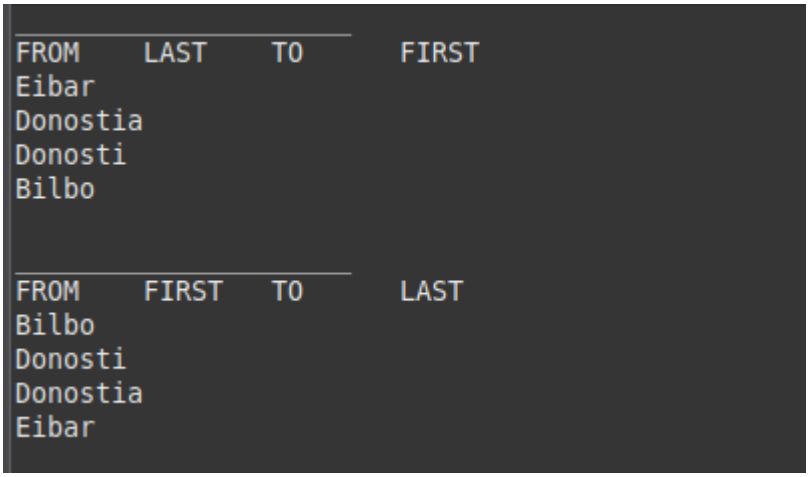
```java
package businessLogic;

import java.util.Iterator;

public interface ExtendedIterator<Object> extends Iterator<Object> {
    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
package businessLogic;

import java.util.List;
import java.util.NoSuchElementException;

public class ExtendedIteratorCities implements ExtendedIterator<String> {

    private List<String> cities;
    private int index = -1;

    public ExtendedIteratorCities(List<String> cities) {
        this.cities = cities;
    }

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

    @Override
    public String next() {
        if (!this.hasNext()) throw new NoSuchElementException();
        index++;
        return cities.get(index);
    }

    @Override
    public String previous() {
        if(!this.hasPrevious()) throw new NoSuchElementException();
        index--;
        return cities.get(index);
    }

    @Override
    public boolean hasPrevious() {
        return (index - 1) >= 0;
    }

    @Override
    public void goFirst() {
        index= -1;

    }

    @Override
    public void goLast() {
        index=cities.size();

    }

}
```

```java
public  ExtendedIterator<String>getDepartingCitiesIterator(){
    dbManager.open();

    List<String> departLocations = dbManager.getDepartCities();

    dbManager.close();

    return new ExtendedIteratorCities(departLocations);

}
```

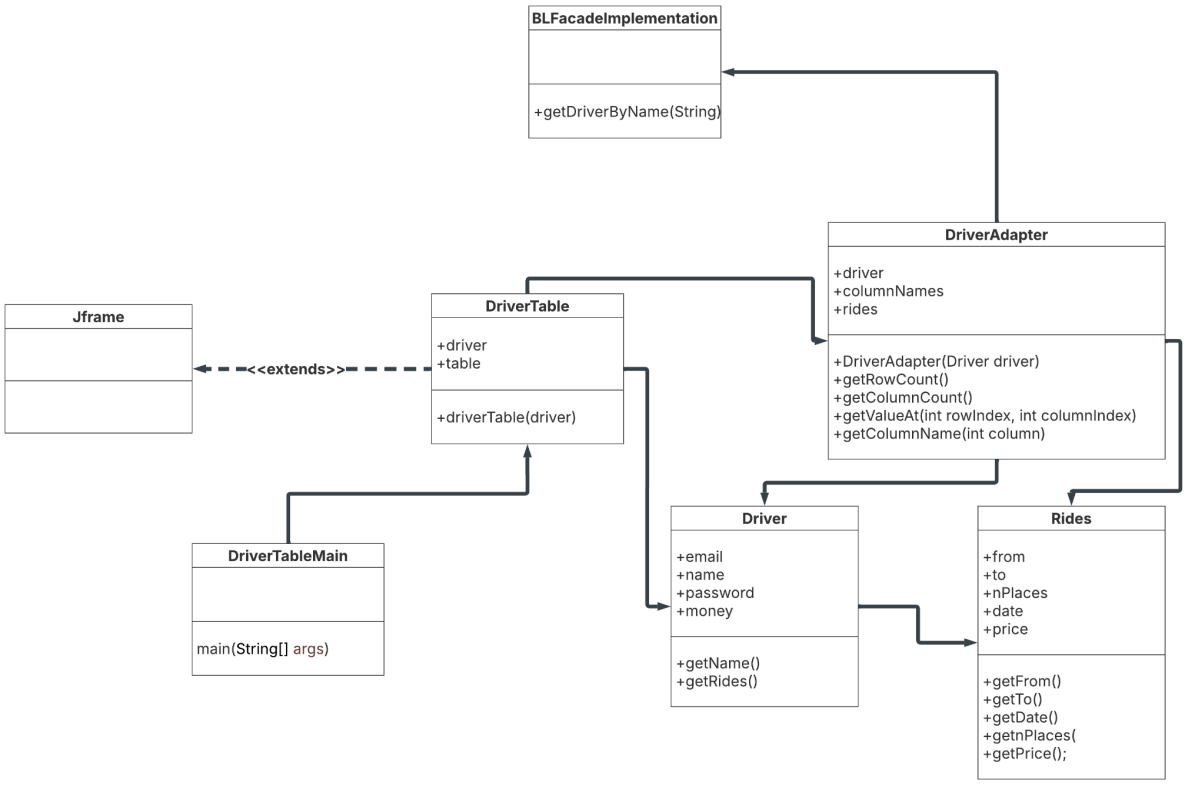

```java
public  ExtendedIterator<String>getDepartingCitiesIterator(){
    dbManager.open();

    List<String> departLocations = dbManager.getDepartCities();

    dbManager.close();

    return new ExtendedIteratorCities(departLocations);

}
```


Exekuzioaren irudia



3. Adapter patroia

UML



Aldatutako kodea

Adapter pakete barruan egin da dena.

```java
package adapter;

import domain.Driver;
import domain.Ride;

import java.util.List;

import javax.swing.table.AbstractTableModel;

public class DriverAdapter extends AbstractTableModel {

    private Driver driver;
    private String[] columnNames = {"From", "To", "Date", "Places", "Price"};
    private List<Ride> rides;

    public DriverAdapter(Driver driver) {
        this.driver = driver;
        this.rides = driver.getRides(); // Driver-en bidai zerrenda eskuratzen dugu
    }

    @Override
    public int getRowCount() {
        return rides.size(); // errenkada kopurua: bidaia kopurua
    }

    @Override
    public int getColumnCount() {
        return columnNames.length; // zutabe kopurua: 5
    }

   @Override
   public Object getValueAt(int rowIndex, int columnIndex) {
       Ride ride = rides.get(rowIndex);
       switch (columnIndex) {
           case 0: return ride.getFrom();
           case 1: return ride.getTo();
           case 2: return ride.getDate();
           case 3: return ride.getnPlaces();
           case 4: return ride.getPrice();
           default: return null;
       }
   }

   @Override
   public String getColumnName(int column) {
       return columnNames[column];
   }
```

Exekuzioaren irudia