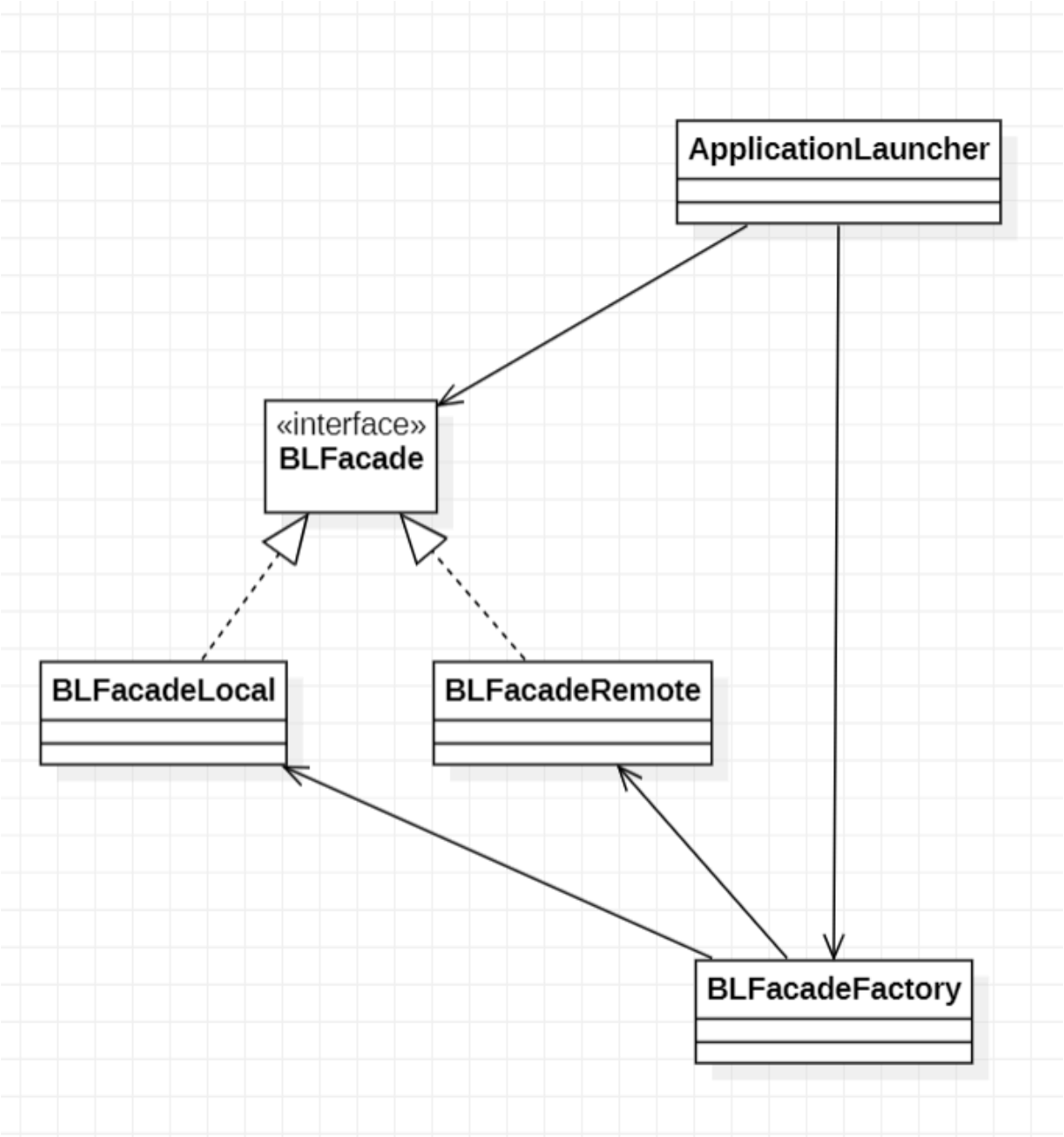
PATROIAK

Oier Unanue, Aretz Gurrutxaga eta Manex Ormazabal

FACTORY METHOD PATROIA

UML DIAGRAMA HEDATUA



ALDATUTAKO KODEA

BLFacadeFactory klasea sortu dugu businessLogic paketean, hau da arduratzen dena dena sortzeaz, “Creator” klasea.

```
package businessLogic;

public class BLFacadeFactory {
    private static boolean isLocal = true; // Ob

    public static BLFacade createBLFacade() {
        if (isLocal) {
            return new BLFacadeLocal();
        } else {
            return new BLFacadeRemote();
        }
    }
}
```

BLFacadeLocal eta BLFacadeRemote ere sortu ditugu, hauek “ConcreteProduct” klaseak izango dira. Bakoitzak bere createBLFacade() a izango du, eta orduan, Factory-n inplementazioan ikusten den bezala, aukeraketa egitean itzuliko da.

```
public class BLFacadeLocal extends BLFacadeImplementation {

    public static BLFacade createBLFacade() throws Exception {
                DataAccess da = new DataAccess();
                return new BLFacadeImplementation(da);
    }
```

```
public class BLFacadeRemote extends BLFacadeImplementation {

    public static BLFacade createBLFacade() throws Exception {
        ConfigXML config = ConfigXML.getInstance();
            String serviceName = "http://" + config.getBusinessLogicNode() + ":" + config.getBusinessL
                                + "/ws/" + config.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);
```

Ez dugu aipatu, baina “Product” funtzioa duen klasea BLFacade izango da.

GUI paketeko ApplicationLauncher klasean egin ditugun aldaketak:

KODE ZAHARRA:

```
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);
}
```

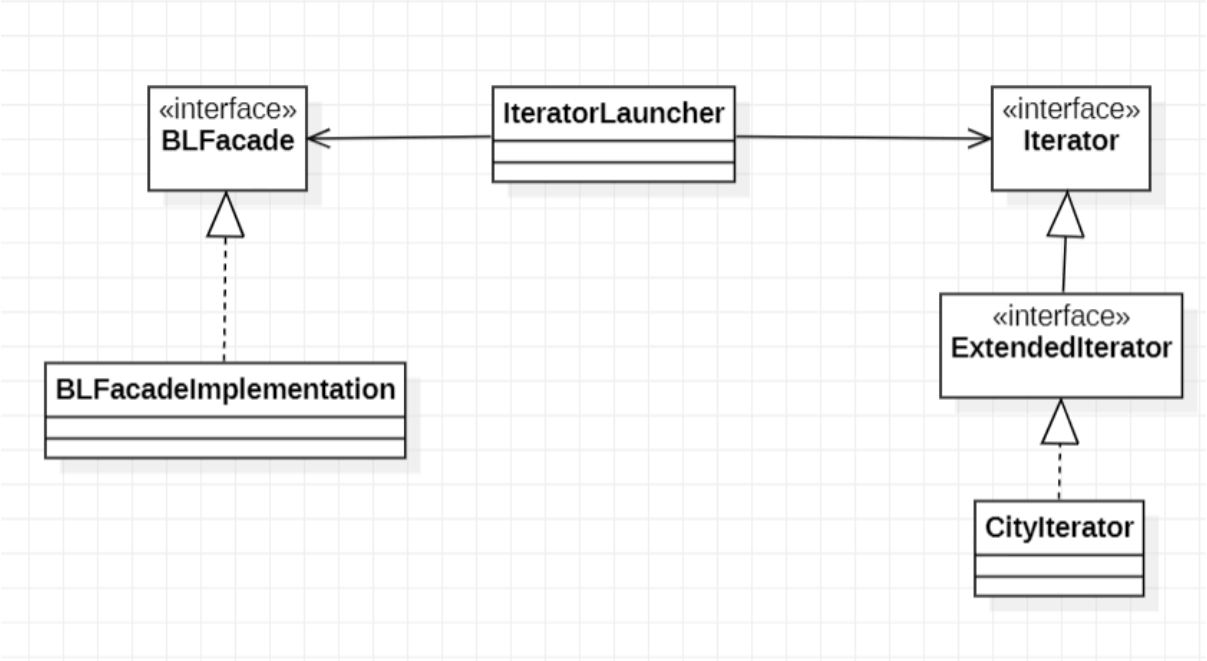
KODE BERRIA:

```
public class ApplicationLauncher {

    public static void main(String[] args) {
        ConfigXML config = ConfigXML.getInstance();
        System.out.println(config.getLocale());
        Locale.setDefault(new Locale(config.getLocale()));
        System.out.println("Locale: " + Locale.getDefault());

        try {
            UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
            BLFacade appFacadeInterface = BLFacadeFactory.createBLFacade();

            // Establecemos la lógica de negocio en la GUI principal
            MainGUI.setBussinessLogic(appFacadeInterface);
            MainGUI mainGui = new MainGUI();
            mainGui.setVisible(true);

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

If, else- a kendu dugu, eta kodea asko sinplifikatu da, ez duelako begiratu behar ea lokala edo remotoa den.

ITERATOR PATROIA

UML DIAGRAMA HEDATUA



ALDATUTAKO KODEA

Lehenik, eskatzen zen bezala ExtendedIterator sortu dugu:

```
package businessLogic;

import java.util.Iterator;

public interface ExtendedIterator<Object> extends Iterator<Object> {
    public Object previous();
    public boolean hasPrevious();
    public void goFirst();
    public void goLast();
}
```

Ondoren CityIterator sortu dugu, hau da hiri lista bat bi direkzioetan gurutzatzeko ahalmena izango duena, hortretarako indize bat erabiliz.

```java
package businessLogic;

import java.util.List;

public class CityIterator implements ExtendedIterator<String> {
    private List<String> cities;
    private int position;

    public CityIterator(List<String> cities) {
        this.cities = cities;
        this.position = 0; // Comienza en el primer elemento
    }

    @Override
```

Honek metodo guztiak gainidatzita izango ditu. goFirst(), hasNext(), goLast(),...

Azkenik, BLFacadeImplementation-en, eskatzen zen metodoa inplementatu dugu, getDepartingCitiesIterator().

```java
@Override
public ExtendedIterator<String> getDepartingCitiesIterator() {
    dbManager.open();
    List<String> departLocations = dbManager.getDepartCities();
    dbManager.close();
    return new CityIterator(departLocations);
}
```

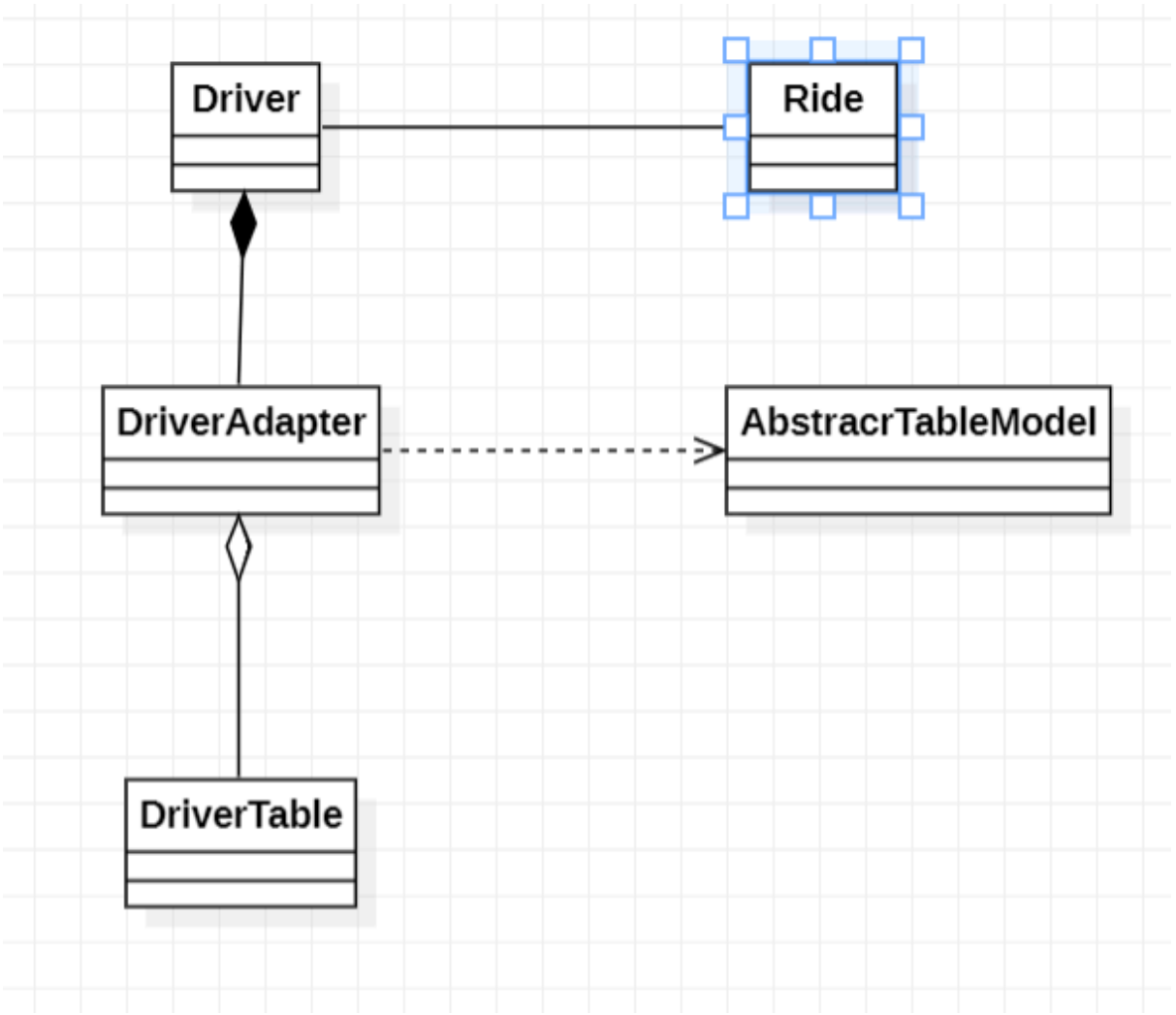
Metodo honek CityIterator bat bueltatuko du, hirien artean iteratzea ahalbidetzen duena.

EXEKUZIOAREN IRUDIA

```
Read from config.xml:    businessLogicLocal=true        databaseLocal=true      dataBaseInitialized=true
File deleted
DataAccess opened => isDatabaseLocal: true
Db initialized
DataAccess created => isDatabaseLocal: true isDatabaseInitialized: true
DataAcess closed
DataAccess opened => isDatabaseLocal: true
DataAcess closed
_____________________
FROM    LAST    TO      FIRST
Madrid
Irun
Donostia
Barcelona


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

ADAPTER PATROIA

UML DIAGRAMA HEDATUA



ALDATUTAKO KODEA

DriverAdapter klase egiten hasi gara, Driver klaseko datuak (eta Ride-rekin lotutako bidaiak) JTablek erabil dezakeen formatu batera egokitzeko. Driver klaseak gidariaren eta bere bidaietako informazioa dauka, baina datu horiek JTable batean erakusteko, formatu zehatz bat behar nuen. JTablek TableModel interfazea ezartzen duen egitura bat espero du. DriverAdapter sortzean, interfazea ezartzen da eta Driver-en informazioa egokitzen da, zuzenean JTablek erabil dezan.

Horrela gelditu da sortutako klase berria:

```java
package gui;

import javax.swing.table.AbstractTableModel;

public class DriverAdapter extends AbstractTableModel {
    private static final long serialVersionUID = 1L;
    private List<Ride> rides;
    private final String[] columnNames = {"From", "To", "Date", "Places", "Price"};

    public DriverAdapter(Driver driver) {
        this.rides = driver.getCreatedRides();
    }

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

    @Override
    public int getColumnCount() {
        return columnNames.length;
    }

    @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];
    }
}
```

Beste aldetik, DriverTable klasea sortu dugu, zein den leiho bat (JFrame) Driver baten bidaietako datuak erakusteko JTable baten bidez. Helburua gidari baten eta bere bidaietako informazioa modu grafikoan erakustea da, taula batean erabiltzailearen interfazearen barruan. DriverTable da leihoa sortzeko eta JTable-a erabiltzeko ardura, eta DriverAdapter erabiltzen du datuen modelo gisa.

Horrela gelditu da sortutako klase berria:

```java
package gui;

import java.awt.BorderLayout;

public class DriverTable extends JFrame{
private Driver driver;
private JTable tabla;
public DriverTable(Driver driver){
super(driver.getUsername()+"'s rides ");
this.setBounds(100, 100, 700, 200);
this.driver = driver;
DriverAdapter adapt = new DriverAdapter(driver);
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);
}
}
```

Bukatzeko AplicationLauncher-en aldaketa batzuk egin ditugu kodigoak ondo funtzionatzen zuela baieztatzeko.

```java
public class ApplicationLauncher {

    public static void main(String[]    args)   {
            // The BL is local
            boolean isLocal = true;
            BLFacade blFacade = new BLFacadeLocal();

            Driver d = blFacade.getDriver("Urtzi");
            DriverTable dt = new DriverTable(d);
            dt.setVisible(true);
        }
}
```

EXEKUZIOAREN IRUDIA

From	To	Date	Places	Price
Donostia	Madrid	Thu May 30 00:00:00 CE...	5	20.0
Irun	Donostia	Thu May 30 00:00:00 CE...	5	2.0
Madrid	Donostia	Fri May 10 00:00:00 CE...	5	5.0
Barcelona	Madrid	Sat Apr 20 00:00:00 CE...	0	10.0