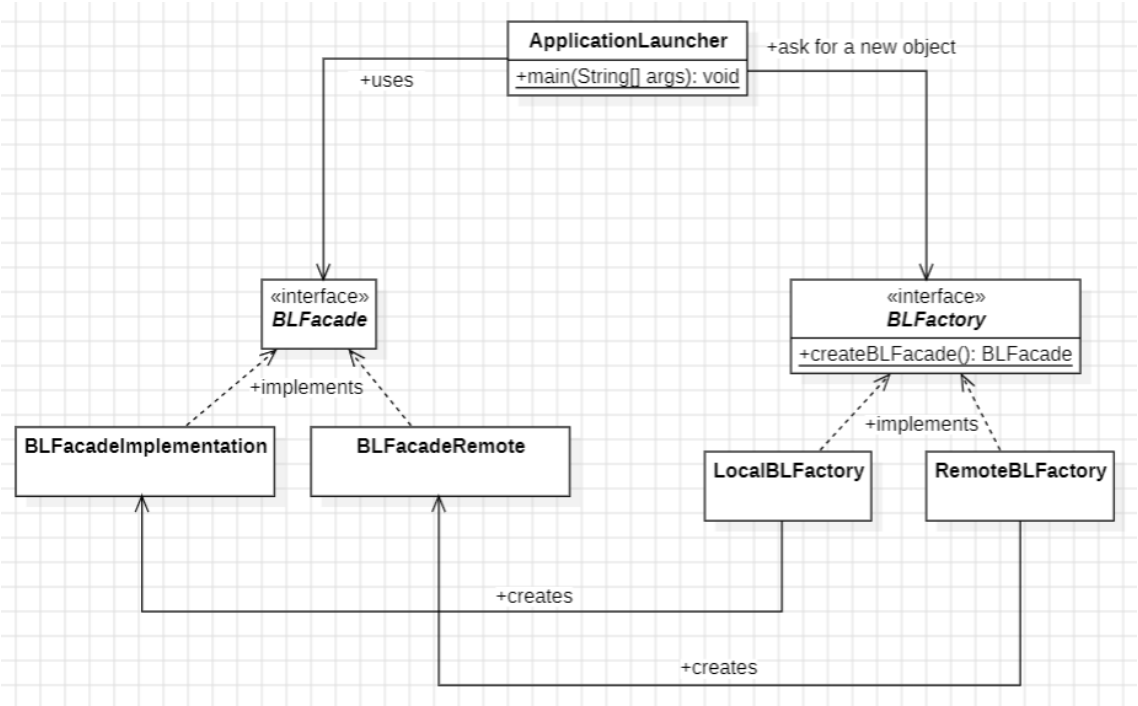
Patroiak Proiektua

- **Egileak:** Asier Aldai eta Aimar Villegas
- **Github helbidea:** https://github.com/Aimarville/Rides25.git

Factory Method Patroia

a) UML diagrama:



b) Aldatutako kodea:

```java
public class LocalBLFactory implements BLFactory {

    @Override
    public BLFacade createBLFacade() {
        DataAccess da = new DataAccess();
        return new BLFacadeImplementation(da);
    }
}
```

- BLFacade lokalean hasieratzen denerako sortutako klasea.

```java
public class RemoteBLFactory implements BLFactory {

    @Override
    public BLFacade createBLFacade() {
        try {
            ConfigXML c = ConfigXML.getInstance();

            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) {
            throw new RuntimeException("Error creating remote BLFacade", e);
        }
    }
}
```

- BLFacade forma remotoan hasieratu nahi denerako klasea.

```java
if (c.isBusinessLogicLocal()) {
    factory = new LocalBLFactory();
}else { //If remote
    factory = new RemoteBLFactory();
}

appFacadeInterface = factory.createBLFacade();

HomeGUI.setBussinessLogic(appFacadeInterface);
```

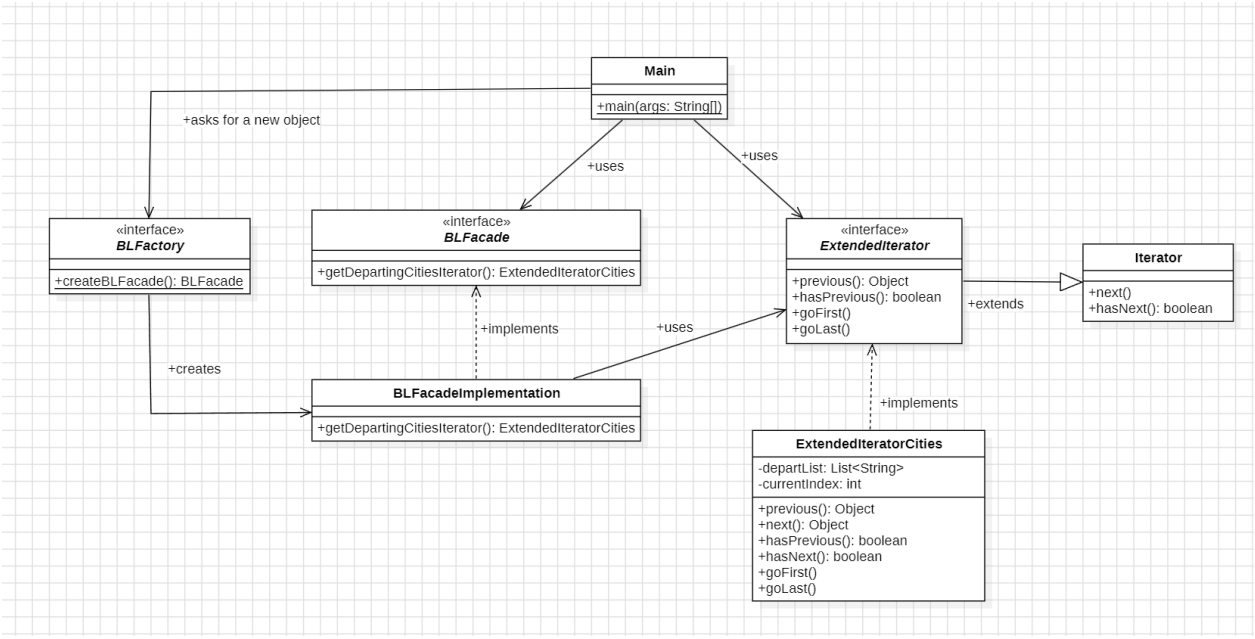
- Main metodoan bi inplementazioak hasieratu.

c) Rolen azalpena

1. **Creator:** BLFactory interfazea.
2. **ConcreteCreator:** LocalBLFactory BLFacadImplementation klasea hasieratzen duela eta RemoteBLFactory BLFacadeRemote klasea hasieratzen duela, BLFactory interfazearen inplementazioak.
3. **Product:** BLFacade interfazea.
4. **ConcreteProduct:** BLFacadeImplementation eta BLFacadeRemote, BLFacade interfazearen inplementazioa. Kasu honetan, pasaden urteko proiektua denez eta ez genuenez klasea ongi funtzionatzea ez dugu inplementatua. Honen ondorioz, sortzen saiatzen da factory-aren bidez. Sortzen saiatzean, ez bada lortzen exzepzio bat botatzen du.

Iterator Patroia

a) UML Diagrama:



b) Aldatutako Kodea:

```
public interface BLFacade  {

    public ExtendedIterator<String> getDepartingCitiesIterator();

    /**
     * This method returns all the cities where rides depart
     * @return collection of cities
     */
    @WebMethod public List<String> getDepartCities();
```

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

    ExtendedIterator<String> departLocationsIterator = new ExtendedIteratorCities(dbManager.getDepartCities());

    dbManager.close();

    return departLocationsIterator;
}
```

- BLFacade eta BLFacadeImplementation klaseetan getDepartingCititesIterator metodo berria sortu dut, ExtendedIterator<String> objektu bat bueltatzen duen. getDepartingCitiesIterator, dbManager.getDepartCities deitu ordez (getDepartCities metodoa egiten duena), ExtendedIterator motako objektu berri bat sortzen du eta parametro bezala dbManager.getDepartCities metodoak itzultzen duen lista sartzen zaio. Azkenik, objektu hori itzultzen du.

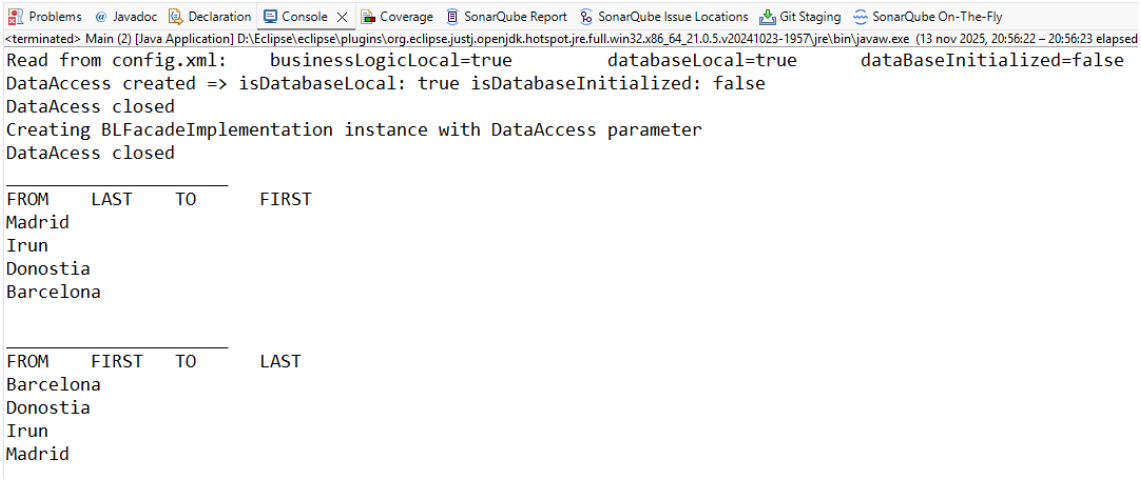
```java
package iterators;

import java.util.Iterator;

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

- ExtendedIterator interfaze berria sortu dut, Iterator interfazea-ri extends eginez.

```java
package iterators;

import java.util.List;

public class ExtendedIteratorCities implements ExtendedIterator

    private List<String> departList;
    private int currentIndex;

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

    @Override
    public Object previous() {
        return departList.get(currentIndex--);
    }

    public Object next() {
        return departList.get(currentIndex++);
    }

    @Override
    public boolean hasPrevious() {
        return currentIndex >= 0;
    }

    public boolean hasNext() {
        return currentIndex <= (departList.size()-1);
    }

    @Override
    public void goFirst() {
        currentIndex = 0;
    }

    @Override
    public void goLast() {
        currentIndex = departList.size()-1;
    }

}
```

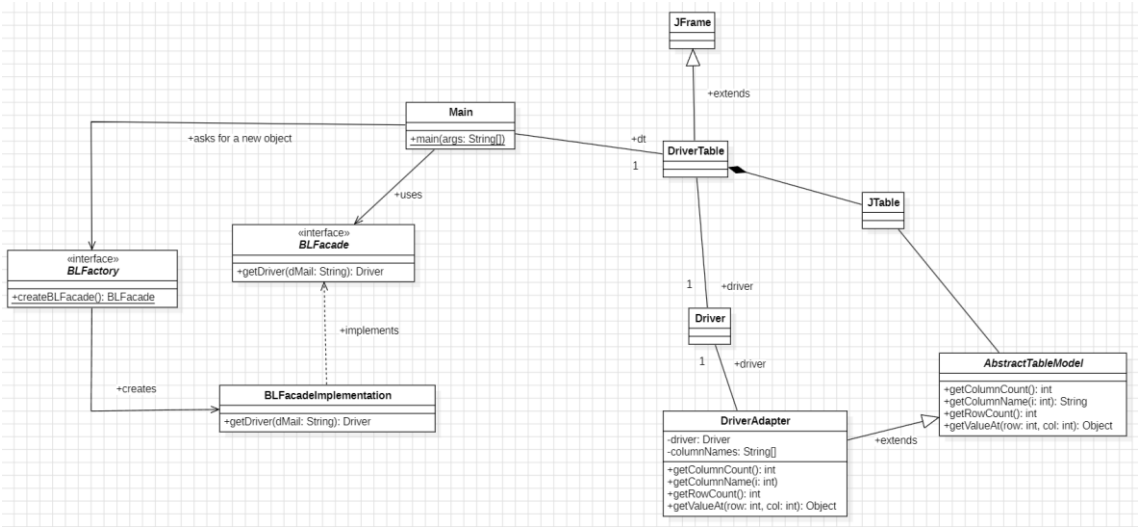
- ExtendedIteratorCities klasea sortu dut, ExtendedIterator interfazea inplementatzen duena. ExtendedIterator-en (eta Iterator-en) interfazearen metodo guztiak inplementatu ditut sartzen zaion listaren arabera funtzionatzeko.

c) Programa exekutatu ondorengo emaitza:



Adapter Patroia

a) UML Diagrama:



b) Aldatutako kodea:

```
package adapter;

import java.awt.*;

public class DriverTable extends JFrame {
    private Driver driver;
    private JTable tabla;

    public DriverTable(Driver driver) {
        super(driver.getEmail() + "'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);
    }
}
```

- DriverTable GUI sortu dut (PDF-tik hartuta).

```java
package adapter;

import businessLogic.*;
import domain.Driver;
import factories.*;

public class Main {

    public static void main(String[]args)   {
        BLFacade    blFacade =  new LocalBLFactory().createBLFacade();
        Driver  d= blFacade.getDriver("asier");
        DriverTable dt=new  DriverTable(d);
        dt.setVisible(true);
    }

}
```

- Gero exekutatuko den programan inplementatu dut (PDF-tik hartuta) eta metodoen izenak aldatu ditut.

```java
    private Driver driver;
    private String[] columnNames = new String[] {"from", "to", "date", "places", "price"};

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

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

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

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

    @Override
    public Object getValueAt(int row, int col) {
        List<Ride> rideList = driver.getRides();
        Ride r = rideList.get(row);

        Object itzul = null;
        switch (col) {
            case 0:
                itzul = r.getFrom();
                break;
            case 1:
                itzul = r.getTo();
                break;
            case 2:
                itzul = r.getDate();
                break;
            case 3:
                itzul = r.getnPlaces();
                break;
            case 4:
                itzul = r.getPrice();
                break;
            default:
                itzul = null;
        }
        return itzul;
    }
}
```

- Lehendabizi, taularen zutabeen izenak array batean sartu ditut. getColumnCount metodoan, tablak izango duen zutabe kopurua lortzeko columnNames-en luzera hartu dut. getColumnName inplementatzeko, eskatzen den posizioan dagoen columnNames-en balioa itzuli dut. getRowCount metodoa metodorako driver-en bidaien listaren luzera itzuli dut. Azkenik, getValueAt metodoan, lehendabizi, eskatzen den bidaia gorde dut driver-aren bidai listatik

hartuta. Ondoren, switch-case erabiliz, parametroz sartzen den zutabearen arabera, bidaiaren atributu bat edo bestea itzuli dut.

```
@WebMethod public Driver getDriver(String dMail);

@WebMethod
public Driver getDriver(String dMail) {
    dbManager.open();
    Driver d = dbManager.getDriver(dMail);
    dbManager.close();
    return d;
}

public Driver getDriver(String dMail) {
    return db.find(Driver.class, dMail);
}
```

- BLFacade, BLFacadeImplementation eta DataAccess klaseetan getDriver metodoa gehitu dut, Main programan erabiltzen delako, zuzenean eskatzen de driver-a lortzeko.

c) Programa exekutatu ondorengo emaitza: