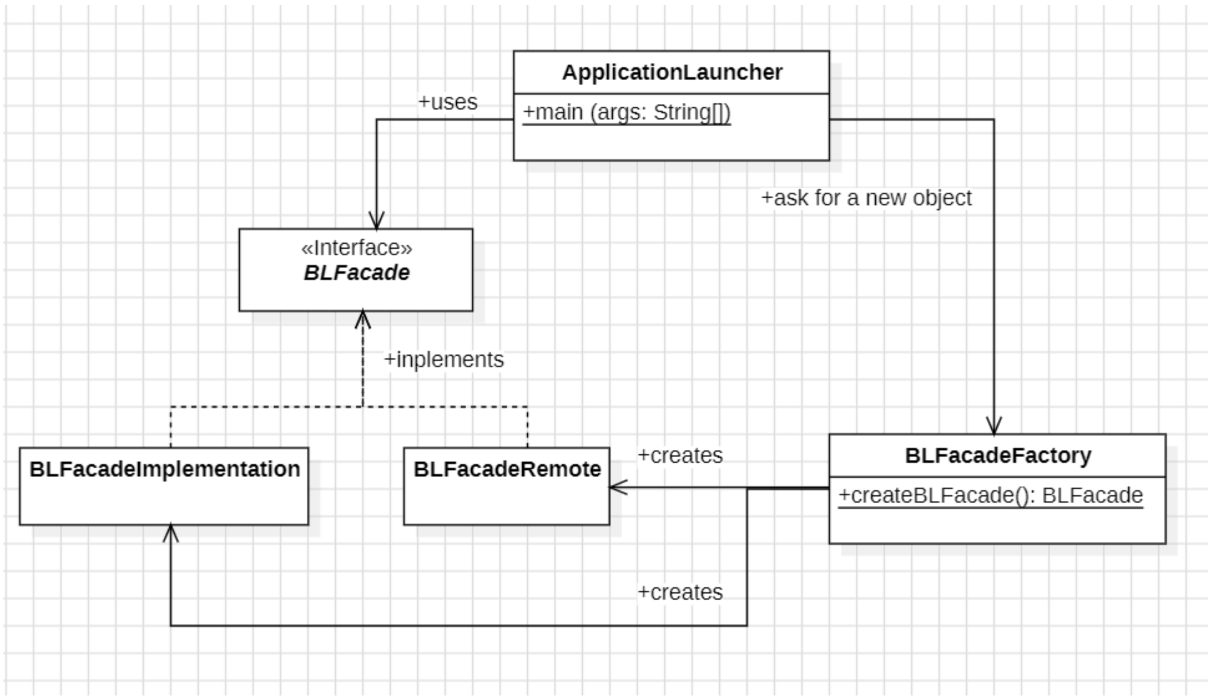
FACTORY METHOD PATROIA

1. UML DIAGRAMA:



ROLAK:

- *CREATOR:* BLFacadeFactory klasea
- *PRODUCT:* BLFacade interfazea
- *CONCRETE PRODUCT:* BLFacadeImplementation (local) eta BLFacadeRemote (Remote)

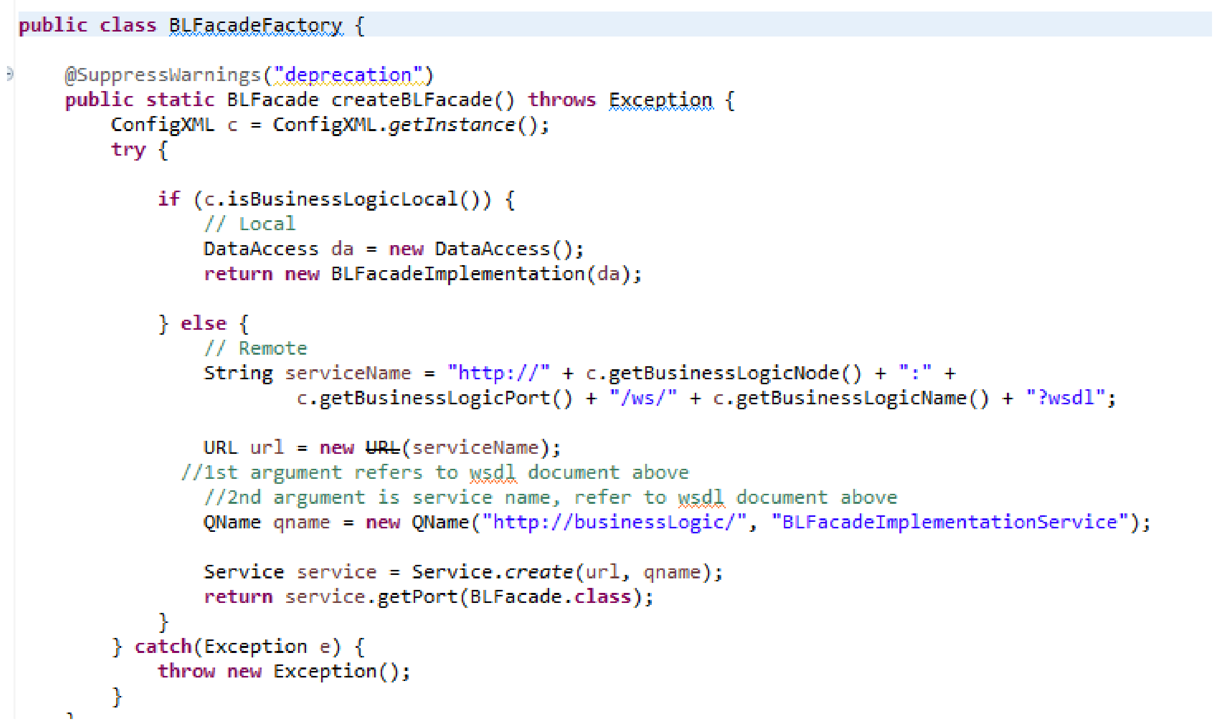
[BLFacadeRemoteren inplementazioa ez dago egina aurreko urteko proiektuan ez genuelako lortu inplemenzatioa zuzen funtzionatzea]

2. ALDATUTAKO KODEA

Orain objetuen, zehazki *BLFacaderen*, sorkuntzak *Factoryan* egiten direnez, *ApplicationLauncheretik* main metodoan honen sorkuntza eskatzen diogu zuzenean hemen sortu beharrean.

```
try {

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

    BLFacade appFacadeInterface = BLFacadeFactory.createBLFacade(); //Sorkuntza eskatu

    MainDriverGUI.setBussinessLogic(appFacadeInterface);
```

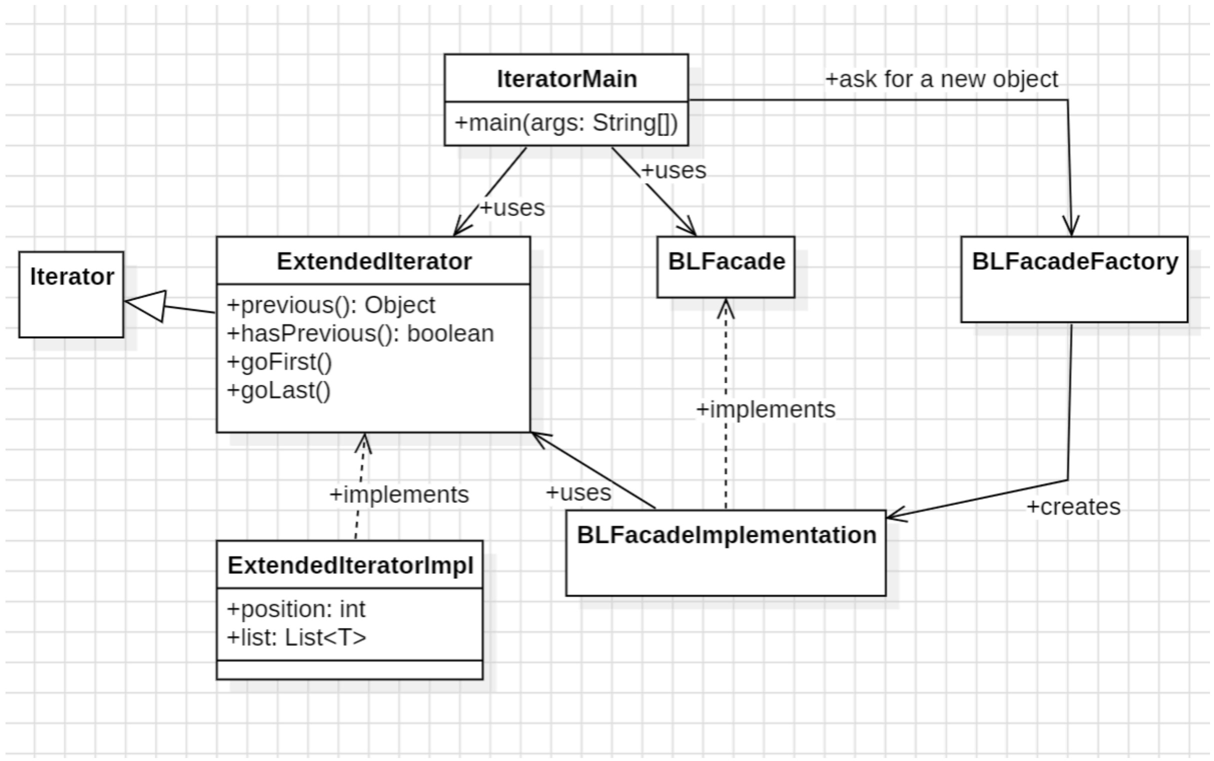
Eta sorkuntzaren kodea *Factory* klasera pasa dugu:



```java
public class BLFacadeFactory {

    @SuppressWarnings("deprecation")
    public static BLFacade createBLFacade() throws Exception {
        ConfigXML c = ConfigXML.getInstance();
        try {

            if (c.isBusinessLogicLocal()) {
                // Local
                DataAccess da = new DataAccess();
                return new BLFacadeImplementation(da);

            } else {
                // 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);
            }
        } catch(Exception e) {
            throw new Exception();
        }
    }
```

ITERATOR PATROIA

1. UML DIAGRAMA:



2. ALDATUTAKO KODEA

ExtendedIterator interfazea sortu dugu, non Iterator interfazean existitzen ez diren *previous*, *hasPrevious*, *goFirst* eta *goLast* funtzioak definitu ditugun. Ondoren *ExtendedIteratorImpl* klasea sortu dugu, *ExtendedIterator* inplementatzen duena, aurreko 4 funtzio horiek gehi *hasNext* eta *next*, lista eta index batez baliaturik.

BLFacade klasean *getDepartCitiesIterator* funtzioa definitu dugu, zein *BLFacadeImplementation* klasean inplementatu dugun.

Bukatzeko, *IteratorMain* klasea sortu dugu bi proba egiteko, non *ExtendedIterator*-ekin eta *BLFacadeFactory* erabiliz behar izan ditugun objetuak sortu ditugun proba burutzeko.

ExtendedIterator:

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

ExtendedIteratorImpl:

```java
public class ExtendedIteratorImpl<T> implements ExtendedIterator<T> {
    private List<T> list;
    private int position = 0;
    public ExtendedIteratorImpl(List<T> list) {
        this.list = list;
    }
    @Override
    public boolean hasNext() {
        return position < list.size();
    }
    @Override
    public T next() {
        return list.get(position++);
    }
    @Override
    public boolean hasPrevious() {
        return position > 0;
    }
    @Override
    public T previous() {
        return list.get(--position);
    }
    @Override
    public void goFirst() {
        position = 0;
    }
    @Override
    public void goLast() {
        position = list.size();
    }
}
```

BLFacadeImplementation:

```java
@WebMethod public ExtendedIterator<String> getDepartCitiesIterator() {
     return new ExtendedIteratorImpl<String>(getDepartCities());
}
```

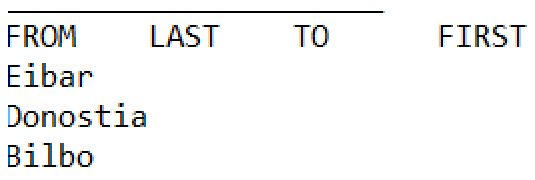
BLFacade:

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

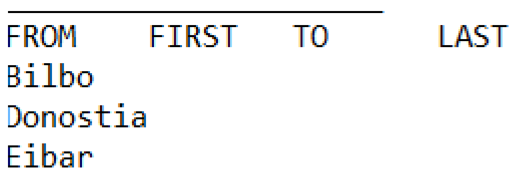
IteratorMain:

```java
public class IteratorMain {

    public static void main(String[]    args)    {
        //the    BL   is   local
        boolean isLocal =    true;
        try {
            BLFacade      blFacade =   BLFacadeFactory.createBLFacade();
            ExtendedIterator<String> i = blFacade.getDepartCitiesIterator();
            String c;
            System.out.println("_____________________");
            System.out.println("FROM     LAST      TO   FIRST");
            i.goLast(); //  Go  to  last    element
            while (i.hasPrevious()) {
                c = i.previous();
                System.out.println(c);
            }
            System.out.println();
            System.out.println("_____________________");
            System.out.println("FROM     FIRST    TO   LAST");
            i.goFirst();     //  Go  to  first   element
            while (i.hasNext()) {
                c = i.next();
                System.out.println(c);
            }
        }catch (Exception e) {
            System.out.println("Error in ApplicationLauncher: "+e.toString());
        }
    }
}
```

3. LORTUTAKO EMAITZA



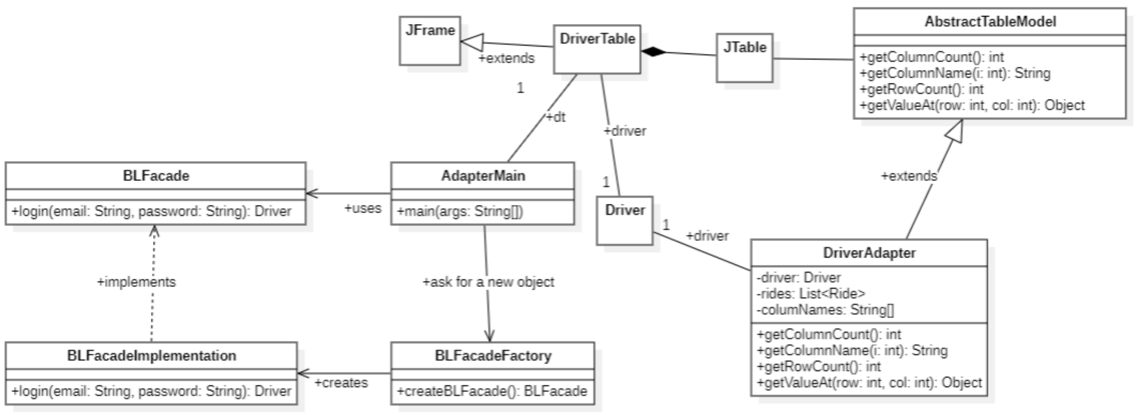
```
FROM     LAST     TO       FIRST
Eibar
Donostia
Bilbo
```



```
FROM     FIRST    TO       LAST
Bilbo
Donostia
Eibar
```

ADAPTER PATROIA

1. UML DIAGRAMA:

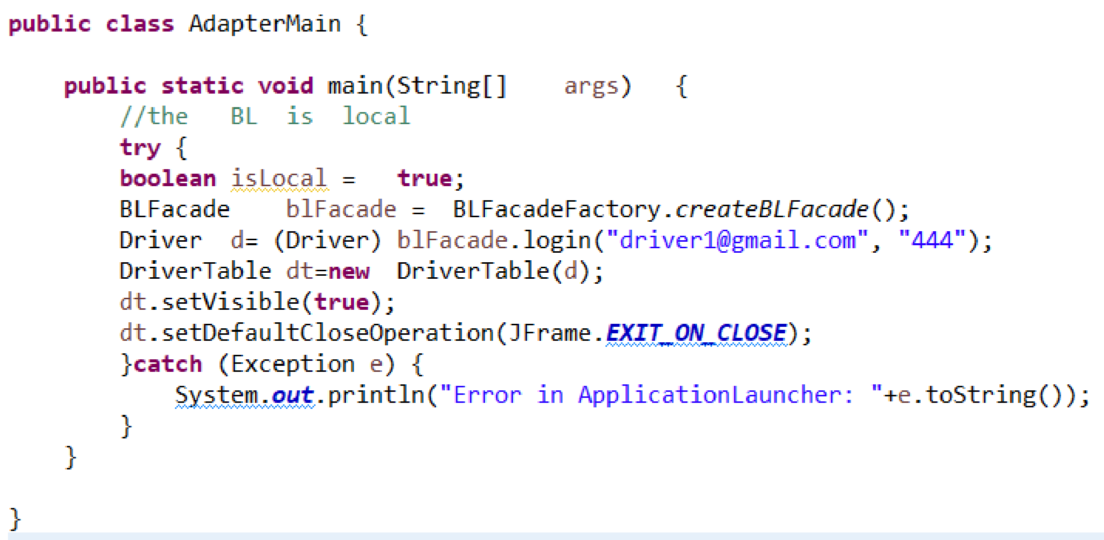


2. ALDATUTAKO KODEA:

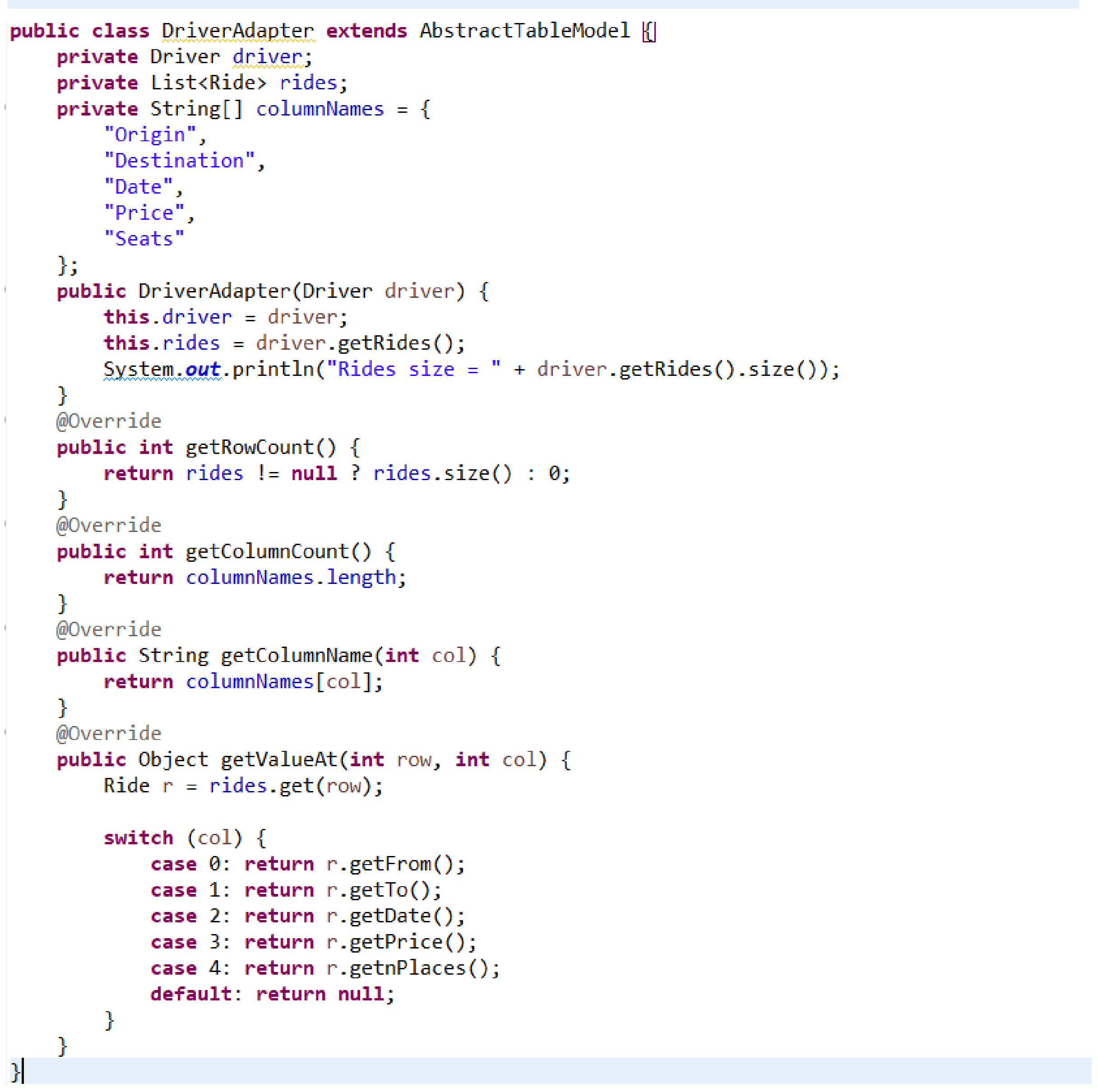
DriverTable klasea sortu dugu, non JFrame-az baliatuz, tabla bat sortzen dugun. Horretarako, *DriverAdapter* klasea erabili dugu non *Driver* baten bidaiak kudeatzen ditugun taulara gehitu ahal izateko.

Guzti hau praktikan jartzeko, *AdapterMain* klasea sortu dugu, non *DriverTable* bat sortzen duen sartutako *Driver*rarekin. *Driver* hau lortzeko *BLFacade* Erabili dugu, datu basea kontsultatzeko.

```
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);
    }
}
```

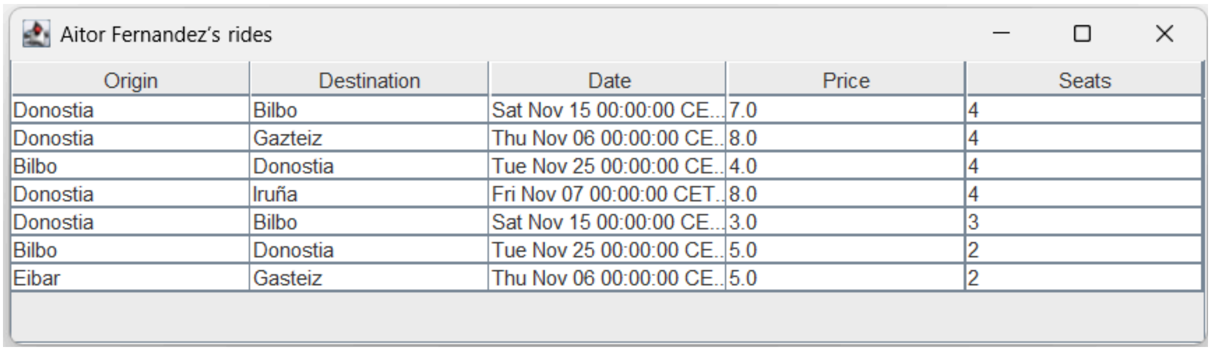


```java
public class AdapterMain {

    public static void main(String[]    args)    {
        //the   BL  is  local
        try {
        boolean isLocal =   true;
        BLFacade    blFacade =  BLFacadeFactory.createBLFacade();
        Driver  d= (Driver) blFacade.login("driver1@gmail.com", "444");
        DriverTable dt=new  DriverTable(d);
        dt.setVisible(true);
        dt.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }catch (Exception e) {
            System.out.println("Error in ApplicationLauncher: "+e.toString());
        }
    }

}
```



```java
public class DriverAdapter extends AbstractTableModel {
    private Driver driver;
    private List<Ride> rides;
    private String[] columnNames = {
        "Origin",
        "Destination",
        "Date",
        "Price",
        "Seats"
    };
    public DriverAdapter(Driver driver) {
        this.driver = driver;
        this.rides = driver.getRides();
        System.out.println("Rides size = " + driver.getRides().size());
    }
    @Override
    public int getRowCount() {
        return rides != null ? rides.size() : 0;
    }
    @Override
    public int getColumnCount() {
        return columnNames.length;
    }
    @Override
    public String getColumnName(int col) {
        return columnNames[col];
    }
    @Override
    public Object getValueAt(int row, int col) {
        Ride r = rides.get(row);

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

3. LORTUTAKO EMAITZA:


Aitor Fernandez's rides

Origin	Destination	Date	Price	Seats
Donostia	Bilbo	Sat Nov 15 00:00:00 CE...	7.0	4
Donostia	Gazteiz	Thu Nov 06 00:00:00 CE...	8.0	4
Bilbo	Donostia	Tue Nov 25 00:00:00 CE...	4.0	4
Donostia	Iruña	Fri Nov 07 00:00:00 CET...	8.0	4
Donostia	Bilbo	Sat Nov 15 00:00:00 CE...	3.0	3
Bilbo	Donostia	Tue Nov 25 00:00:00 CE...	5.0	2
Eibar	Gasteiz	Thu Nov 06 00:00:00 CE...	5.0	2

Egileak: Aimar Esparza eta Iker Mayordomo