RIDES24 PROIEKTUKO DISEINU PATROIAK

- Factory Method patroia:

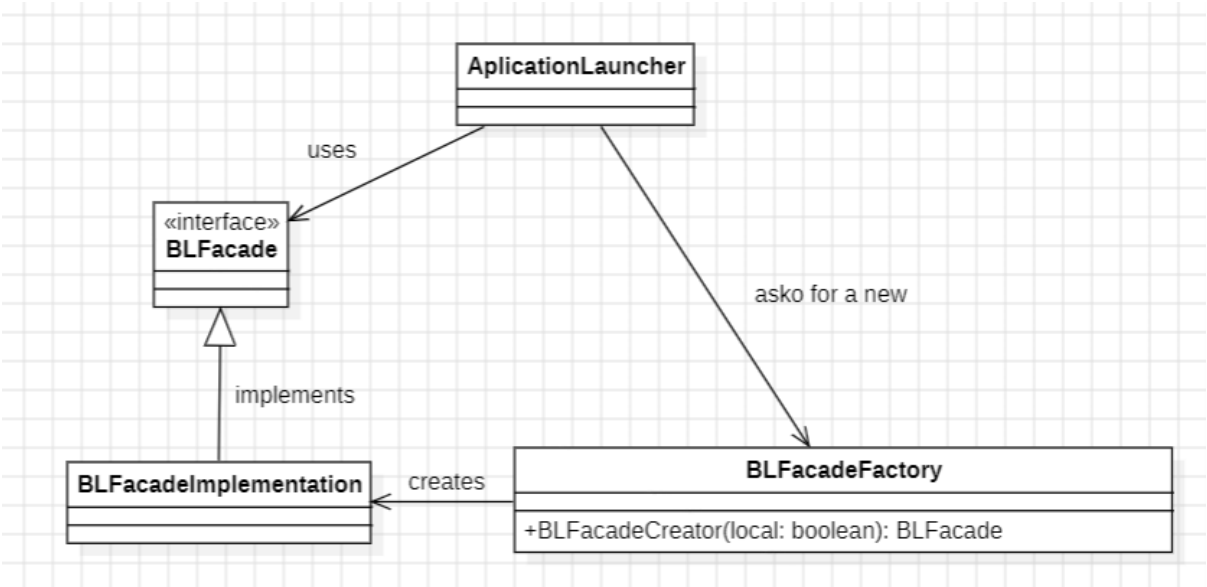
AplicationLauncher klaseko main() metodoan zegoen kodearen zati bat BLFacadeFactory klaseko BLFacadeCreator() metodora atera dugu. Izan ere, main() metodo horretan if baten bitartez aukeratzen zen zer motatako negozio logika erabili behar zuen aplikazioak, lokalean sortutakoa edota web zerbitzuen bitartez jaso behar zuen. Horrela, honako hau da BLFacadeCreator() metodoa:

```
public BLFacade BLFacadeCreator(boolean isLocal) {

    ConfigXML c = ConfigXML.getInstance();
    try {
        BLFacade appFacadeInterface;
        if(isLocal) {
            DataAccess da = new DataAccess();
            appFacadeInterface = 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);

            appFacadeInterface = service.getPort(BLFacade.class);
        }
        return appFacadeInterface;
    } catch(Exception e) {
        System.out.println("Error in ApplicationLauncher: " + e.toString());
        return null;
    }
}
```

Ondorioz, AplicationLauncher klasean horrela jasoko da BLFacade-a:

```
BLFacadeFactory blff = new BLFacadeFactory();
BLFacade appFacadeInterface = blff.BLFacadeCreator(c.isBusinessLogicLocal());
```

Horrela geratuko litzateke UMLa Factory patroia aplikatu eta gero:



- Iterator patroia:

Gure proiektuan eskatutako guztiarekin iterator patroia implementatu ahal izateko, lehenik, ematen zitzaigun ExtendedIterator interfazea sortu dugu, Iterator interfaze orokorra implementatzen duena. Ondoren, ExtendedIterator interfazea implementatzen duen ExtendedIteratorCities klasea sortu dugu:

```
public class ExtendedIteratorCities implements ExtendedIterator<String> {

    private List<String> citiesList;
    private int position;

    public ExtendedIteratorCities(List<String> cities) {
        citiesList = cities;
        position = 0;
    }

    @Override
    public boolean hasNext() {
        if(position <= citiesList.size() - 1) return true;
        else return false;
    }

    @Override
    public String next() {
        String ret = citiesList.get(position);
        this.position++;
        return ret;
    }

    @Override
    public String previous() {
        String ret = citiesList.get(position - 1);
        this.position--;
        return ret;
    }

    @Override
    public boolean hasPrevious() {
        if(position > 0) return true;
        else return false;
    }
```

```
    @Override
    public void goFirst() {
        this.position = 0;
    }

    @Override
    public void goLast() {
        this.position = citiesList.size();
    }
```

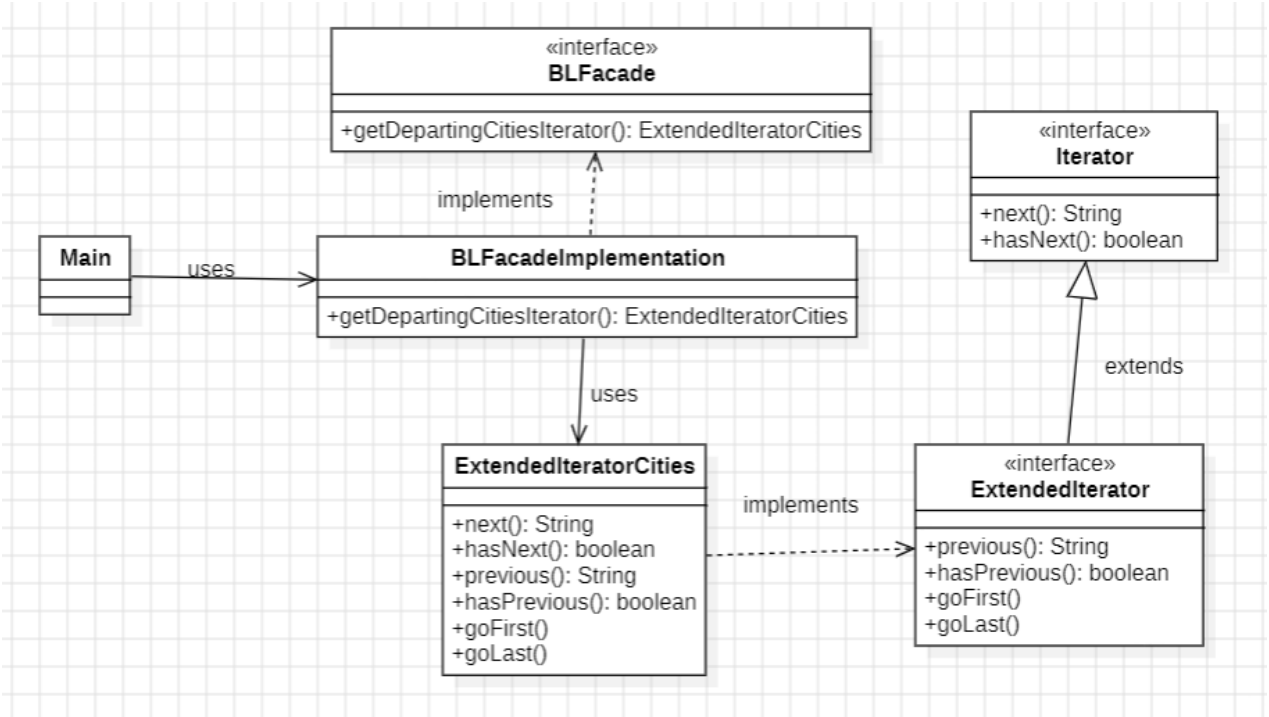
Jarraian, BLFacade interfazean eta BLFacadeImplementation klasean getDepartingCitiesIterator() metodoari ondorengo implementazioa eman diogu:

```
@WebMethod
public ExtendedIterator<String> getDepartingCitiesIterator() {
    ExtendedIterator<String> i = new ExtendedIteratorCities(this.getDepartCities());
    return i;
}
```

Metodo honek, hirien zerrendaren gaineko iteradorea itzuliko du. Azkenik, emandako metodo nagusia exekutatzen badugu, honako emaitzak lortuko ditugu:

```
Read from config.xml:    businessLogicLocal=true         databaseLocal=true      dataBaseInitialized=true
nov 07, 2024 6:44:02 P. M. data.access.DataAccess <init>
INFORMACIÓN: File deleted
nov 07, 2024 6:44:03 P. M. data.access.DataAccess open
INFORMACIÓN: DataAccess opened => isDatabaseLocal: true
nov 07, 2024 6:44:04 P. M. data.access.DataAccess initializeDB
INFORMACIÓN: Db initialized
nov 07, 2024 6:44:04 P. M. data.access.DataAccess <init>
INFORMACIÓN: DataAccess created => isDatabaseLocal: true isDatabaseInitialized: true
nov 07, 2024 6:44:04 P. M. data.access.DataAccess close
INFORMACIÓN: DataAcess closed
Creating BLFacadeImplementation instance with DataAccess parameter
nov 07, 2024 6:44:04 P. M. data.access.DataAccess open
INFORMACIÓN: DataAccess opened => isDatabaseLocal: true
nov 07, 2024 6:44:04 P. M. data.access.DataAccess close
INFORMACIÓN: DataAcess closed

FROM    LAST    TO      FIRST
Madrid
Irun
Donostia
Barcelona

FROM    FIRST   TO      LAST
Barcelona
Donostia
Irun
Madrid
```

Patroi hau aplikatuta, amaierako UMLa ondorengoa izango litzateke:



- Adapter patroia:

Gure proiektuan eskatutako guztiarekin adapter patroia implementatu ahal izateko, lehenik, ematen zitzaigun DriverTable klasea sortuko dugu, gui paketearen barruan. Ondoren, businessLogic paketearen barruan, DriverAdapter interfazea sortu dugu, AbstractTableModel klase orokorra heredatzen duena. Azkenik, main-a businessLogic paketean inplementatu dugu, AdapterProba klasean.

Hona hemen sortutako kodea:

DriverTable:

```
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);
    }
}
```

(Enuntziatuan emandako kodearekin)

DriverAdapter:

```java
public class DriverAdapter extends AbstractTableModel{

    protected Driver d;
     protected String[] columnNames =
                new String[] {"From", "To", "Date", "Places", "Price" };
     public DriverAdapter(Driver d) {
         super();
         this.d=d;
     }
    @Override
    public int getRowCount() {
         return d.getCreatedRides().size();
    }

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

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
         Iterator<Ride> iterator = d.getCreatedRides().iterator();
          Ride currentRide = null;

          for(int i = 0; i <= rowIndex; i++) {
              if(iterator.hasNext()) {
                  currentRide = iterator.next();
              } else {
                  return null;
              }
          }
          if(currentRide == null) {
              return null;
          }
          switch(columnIndex) {
            case 0:
                return currentRide.getFrom();
            case 1:
                return currentRide.getTo();
            case 2:
                return currentRide.getDate();
            case 3:
                return currentRide.getnPlaces();
            case 4:
                return currentRide.getPrice();
            default:
                return null;
          }
    }
```

AbstractTableModel-en metodoak inplementatu ditugu, zutabeak beti 5 izango dira, eta beraien izenengatik lortzen ditugu. Lerroak berriz, gidariaren bidaia kopurua izango dira.

getValueAt() metodoa inplementatzeko berriz, iterator bat erabili dugu, nahi dugun lerrora iristeko, eta ondoren zutabearen arabera informazio ezberdina bueltatzen dugu.

AdapterProba:

```java
public class AdapterProba {
    public static void main(String[]    args)   {
//      the BL  is  local
    boolean isLocal =   true;
    BLFacade    blFacade =  new BLFacadeFactory().BLFacadeCreator(isLocal);
    Driver  d= blFacade. getDriver("Urtzi");
    DriverTable dt=new  DriverTable(d);
    dt.setVisible(true);
    }
}
```

(Enuntziatuan emandako kodearekin)

Hori eginda, gidari baten, kasu honeta, Urtziren bidaiak agertuko diren taula bat sortuko dugu:

Urtzi's rides

A	B	C	D	E
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

Patroi hau aplikatuta, amaierako UMLa ondorengoa izango litzateke: