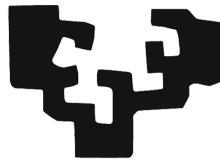


eman ta zabal zazu



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

IZENBURUA

Rides Proiektua

INFORMATIKA INGENIARITZA GRADUA

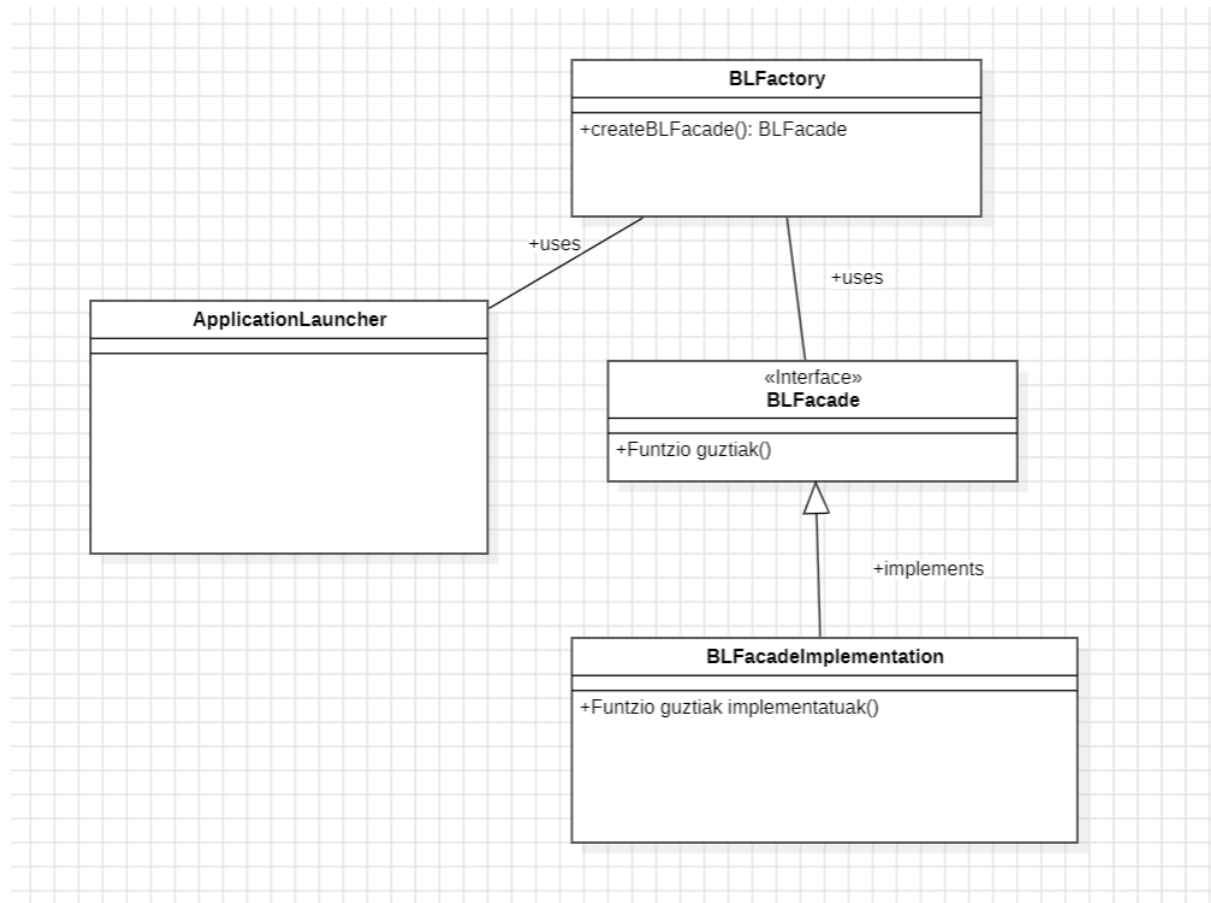
Egileak:

JULEN ETXEBERRIA

AIMAR SAN MARTIN

2024-ko azaroak 9a

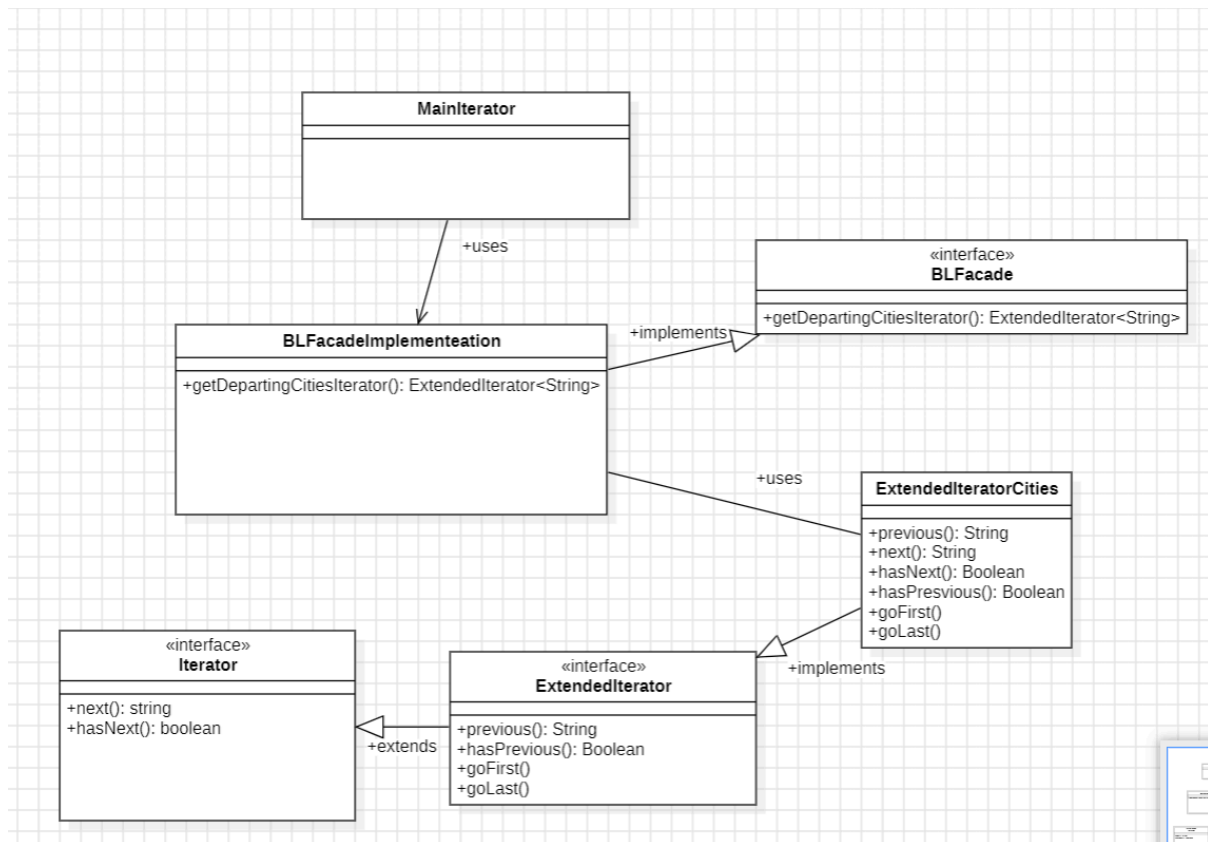
Factory Method Patroia:



Gure proiektuan patroia hau implementatzeko lehenik *BLFactory* klasea sortu dugu. Honek begiratzeko duen ea modu lokalean exekutatu behar den, honetarako, `config.isBusinessLogicLocal()` erabiltzen da. *True* bueltatzen badu modu lokalean exekutatzen da, eta bestela urrunekoan. Ondoren *ApplicationLauncher* klasea aldatu dugu *BLFactory* sortu dugun klase erabiltzeko, modu honetan aplikazioa exekutatzean modu lokalean edo urrunekoan izan dezake.

UML-an ikusten dez bezala *ApplicationLauncher* klaseak ez du *BLFacade* erabiltzen hasieran bezala, orain, guk sortutako klasea erabiltzen du.

Iterator patroia



Lehenik, patroi hau inplementatzeko, *ExtendedIterator* interfazea sortu dugu eta *ExtendedIteratorCities* klasea, aipatutako interfazea inplementatzen duena. Ondoren, *BLFacadeImplementation* klasean metodo berri bat sortu dugu, *getDepartingCitiesIterator()*, honek bueltatzen du lehen sortutako interfaze bat. Amaitzeko *MainIterator* klasea sortu dugu egindakoa ondo funtzionatzen duela ikusteko.

UML-a begiratzuz, guk sortutako *ExtendedIterator* interfazea ikusten dugu *Iterator* interfazea inplementatzen duela, hau noski, ez da guk sortutakoa, java.util -etik importatua da. Beste aldetik, ikusten dira lehen aipatutako aldaketak, bai *ExtendedIteratorCities* klaseak gure interfazea inplementatzen duela eta *BLFacadeImplementation* klasean metodo berri bat egin dugula, aipatutako klasea erabiliz.

Exekuzioaren irudia:

```

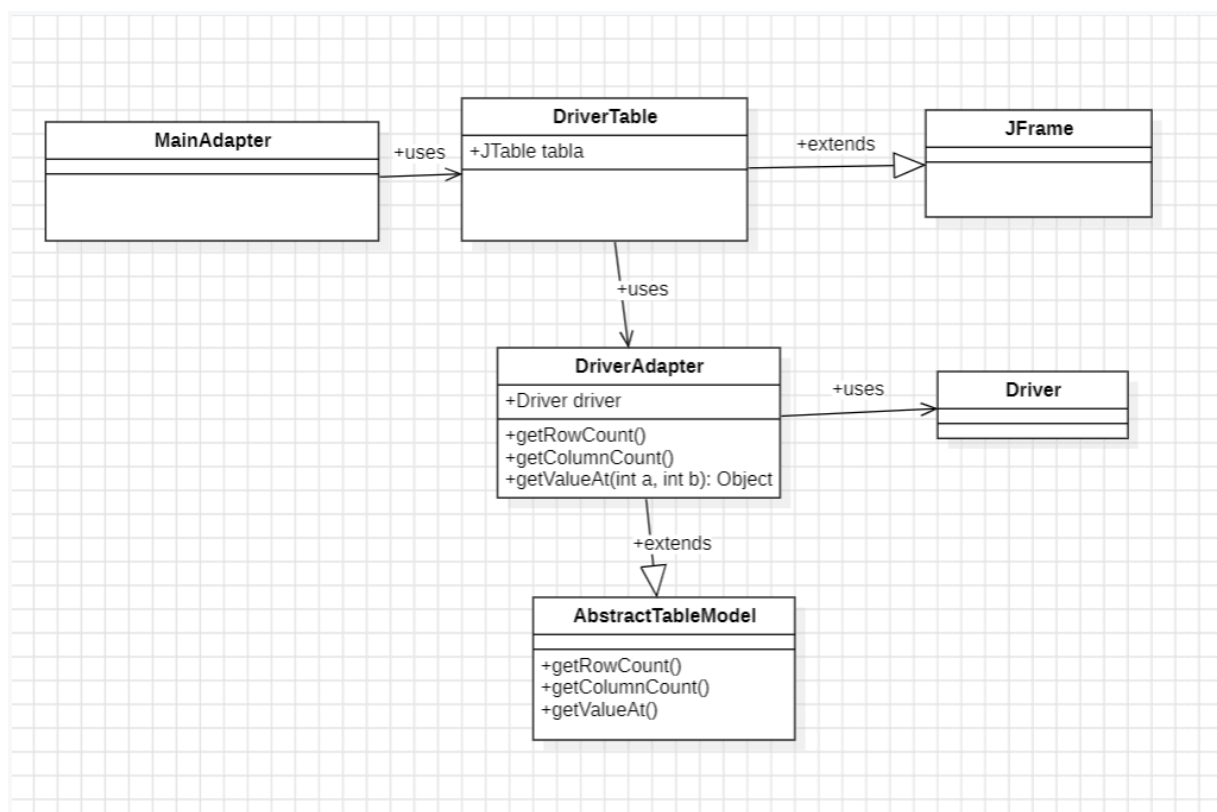
nov 10, 2024 4:37:51 P. M. businesslogic.BLFacadeImplementation <init>
INFO: Creating BLFacadeImplementation instance with DataAccess parameter
DataAccess opened => isDatabaseLocal: true
DataAccess closed

FROM LAST TO FIRST
Madrid
Irun
Donostia
Barcelona

FROM FIRST TO LAST
Barcelona
Donostia
Irun
Madrid

```

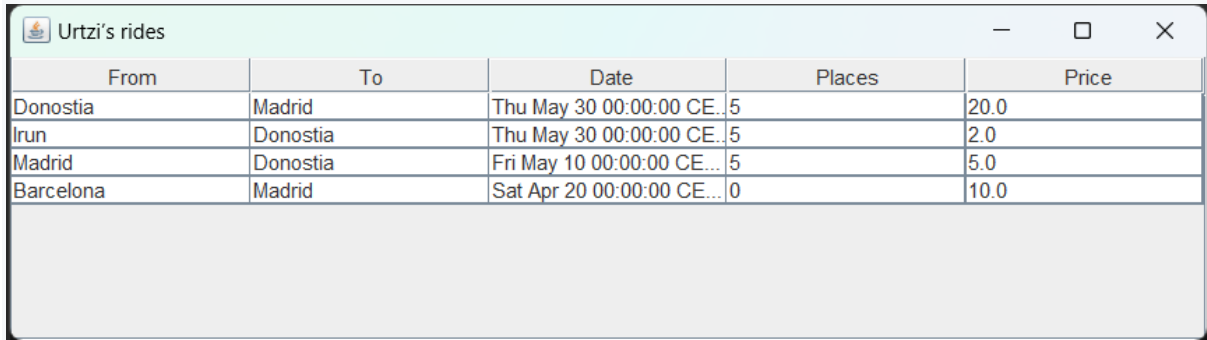
Adapter Patroia



Lehenik, patroia inplementatzeko, *DriverAdapter* klasea sortu dugu, honek, *AbstractTableModel* erabiltzen du. Honekin *Driver* objektuak tabla batean ikusi daitezke. Sortu berri dugun klasea erabiltzeko *DriverTable* klasea sortu dugu, tabla ikusteko. Azkenik *MainAdapter* klasea nagusia da, honekin erabili eta probatzen dugu.

UML-an ikusten da *DriverTable* klaseak erabiltzen duela *AbstractTableModel*, lehen aipatu bezala, hau, `javax.swing.table.AbstractTableModel`-tik importatua da. Beste aldetik, *DriverAdapter* erabiltzen duen *Driver* klasea guk proiektuan genuen klase bat da.

Exekuzioaren irudia:



The screenshot shows a Java Swing window titled "Urtzi's rides". Inside the window is a table with five columns: "From", "To", "Date", "Places", and "Price". The table contains four rows of data. Below the table is a large, empty rectangular area.

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

GITHUB LINK: <https://github.com/aimarsanm/Rides24Complete-Aimar>