PROIEKTUAREN DISEINU PATROIAK

Egileak: Bilal Fouzir, Unax Arruti eta Egoitz Elosua
Irakasgaia: Software Ingeniaritza II

➔ **Github-eko esteka:** https://github.com/Bilal-Fouzir/Rides25

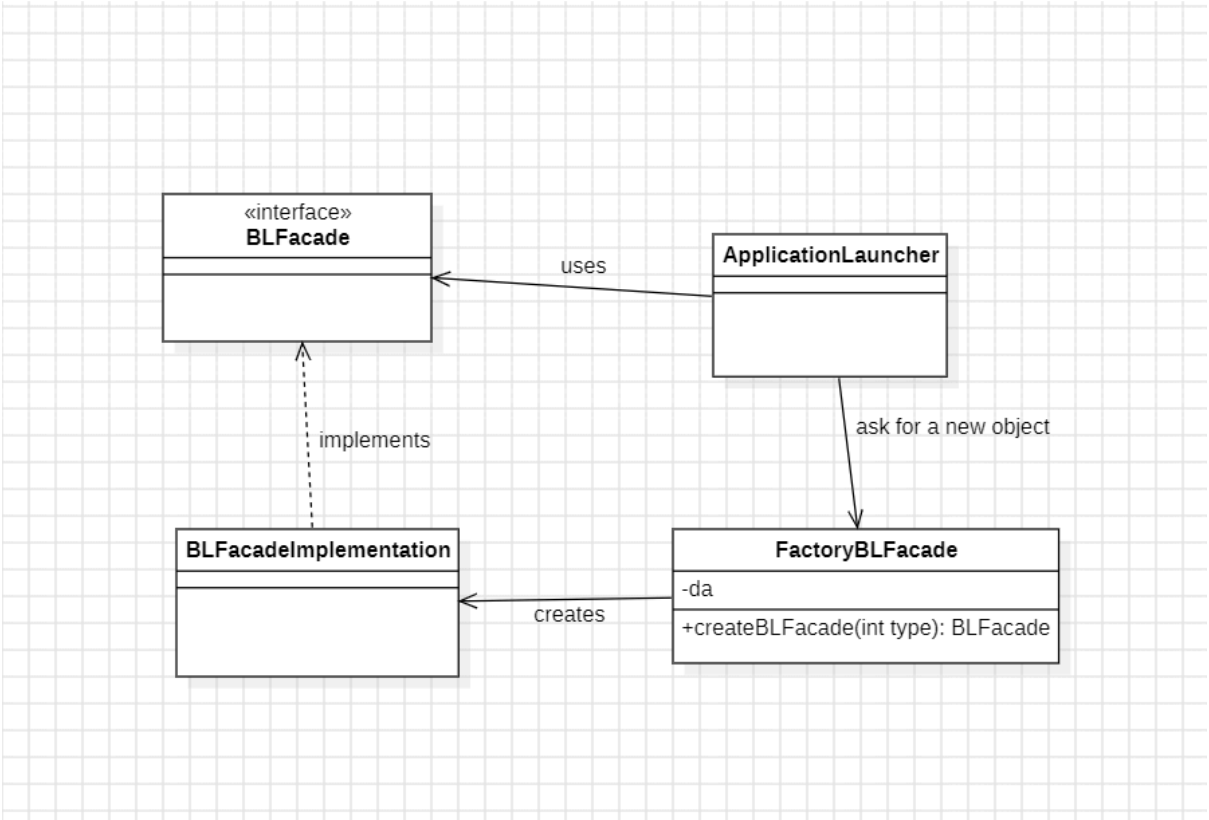
AURKIBIDEA:

PATROIAK

❖ FACTORY METHOD PATROIA

Eskatzen da: Aplikazioa aldatu negozio logikako objektuaren lorpena faktoria objektu batean zentralizatuta egoteko, eta aurkezpenak zein negozio logikako inplementazio erabili erabaki dezatela. Diseina eta inplementatu ebazpena Creator, Product eta ConcreteProduct jokatzen duten klaseen rolak garbi aurkeztuz.

1. **UML diagrama hedatua, egin dituzun aldaketak aurkeztuz.**



ApplicationLauncher bezeroa da, BLFacade produktua, BLFacadeImplementation produktu zehatza, eta FactoryBLFacade factoria.

2. **Aldatu duzun kodea, lerro garrantzitsuenak azalduz.**

Factory patroia jarraitzeko,FactoryBLFacade factoria-klasea sortu dugu. Klase hau, BLFacade implementazio ezberdinak sortzeaz arduratzen da, ApplicationLauncherrak (bezeroak) erabil ditzan.

```
package service;
import dataAccess.DataAccess;
```

```
public class FactoryBLFacade {
	private static DataAccess da= new DataAccess();
	public static BLFacade createBLFacade(int type) {
		if(type==1) {
			return new BLFacadeImplementation(da);
		}else {
			return null;
		}
	}
}
```

ApplicationLauncherrak jada ez du BLFacade bat sortzen. Horren ordez, FactoryBLFacaderi ezkatzen dio sortzeko. Honela, erabiltzaileak erabaki ahal izango luke zein implementazio erabili, naiz eta guk bakarra eduki.

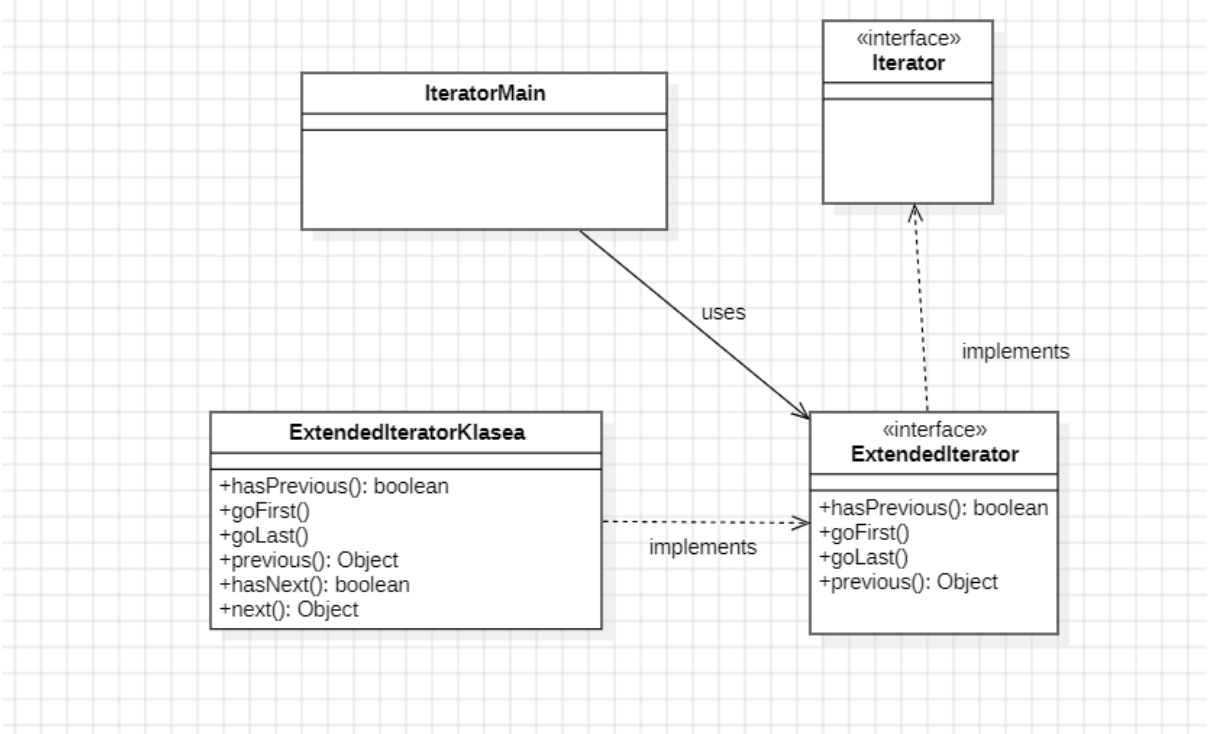
```
public static void main(String[] args) {
		ConfigXML c=ConfigXML.getInstance();
		System.out.println(c.getLocale());
		Locale.setDefault(new Locale(c.getLocale()));
		System.out.println("Locale: "+Locale.getDefault());
		//Driver driver = new Driver("a", "b");
		//MainGUI a=new MainGUI(driver);
		//a.setVisible(true);
		Scanner scanner = new Scanner(System.in);
		try {
			BLFacade appFacadeInterface;
			UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

			if (c.isBusinessLogicLocal()) {
				System.out.print("Aukeratu zenbaki bat:  ");
				System.out.print("1 XML");
				System.out.print("Beste edozer(null)");
				int zenbaki = scanner.nextInt();
				appFacadeInterface= FactoryBLFacade.createBLFacade(zenbaki);
			}
			else { //If remote
				....
```

❖ ITERATOR PATROIA

Eskatzen da: Iteratzaile Hedatua inplementatu, eta, adibidearen antzeko programa bat inplementatuz, hiriak aurkeztutako ordenean inprimatu.

1. **UML diagrama hedatua, egin dituzun aldaketak aurkeztuz.**



2. **Aldatu duzun kodea, lerro garrantzitsuenak azalduz.**

Atzeraka korritu daitezkeen iteradoreak definitzen dituen interfazea sortu du:

```
public interface ExtendedIterator extends Iterator {
        //return the actual elementand 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();
        }
```

Eta interfazea inplementatzen duen klase bat, BLFacadek sor dezan.

```
        package domain;
        import java.util.List;
        import java.util.Iterator;
        public class ExtendedIteratorKlasea implements ExtendedIterator{
```

```
		private List<String> departingCities;
		private int ind=-1;
		public ExtendedIteratorKlasea(List<String> departingCities) {
			this.departingCities=departingCities;
		}

		@Override
		public boolean hasNext() {
			// TODO Auto-generated method stub
			return ind+1 < departingCities.size();
		}
		@Override
		public Object next() {
			// TODO Auto-generated method stub
			ind++;
			return departingCities.get(ind);
		}
		@Override
		public Object previous() {
			// TODO Auto-generated method stub
			ind--;
			return departingCities.get(ind);
		}
		@Override
		public boolean hasPrevious() {
			// TODO Auto-generated method stub
			return ind-1 >= 0;
		}
		@Override
		public void goFirst() {
			// TODO Auto-generated method stub
			ind=-1;
		}
		@Override
		public void goLast() {
			// TODO Auto-generated method stub
			ind=departingCities.size();
		}
	}
```

BLFacadek sortu behar duenez Iteratzaile mota berri hau, sortzeko metodo bat behar du:

```
@WebMethod public ExtendedIterator getDepartingCitiesIterator() {
    return new ExtendedIteratorKlasea(this.getDepartCities());
  }
```

Hau da exekuzioaren kodea:

```
public class IteratorMain{

	public static void main(String[] args)    {
```

```
        try {

                boolean isLocal = true;


                BLFacade bl= FactoryBLFacade.createBLFacade(1);
                Driver d= new Driver("b","a");
                bl.createDriver(d);
                bl.addCar("marka", "modeloa", "matrikula", d);
                Kotxea k= new Kotxea("marka", "modeloa", "matrikula");
                bl.createRide("Madrid", "Mandril", new Date(2030, 02, 11), 2, 2, "b", k.toString());
                bl.createRide("Mandril", "Madrid", new Date(2030, 02, 11), 2, 2, "b", k.toString());

                ExtendedIterator  i = bl.getDepartingCitiesIterator();
                String c;

                System.out.println("_____");
                System.out.println("FROM      LAST   TO      FIRST");
                i.goLast();         //        Go      to      last      element
                while (i.hasPrevious())  {
                c =       (String) 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 =       (String) i.next();
                System.out.println(c);
                }
        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        }
}
```

3. Exekuzioaren irudi bat.

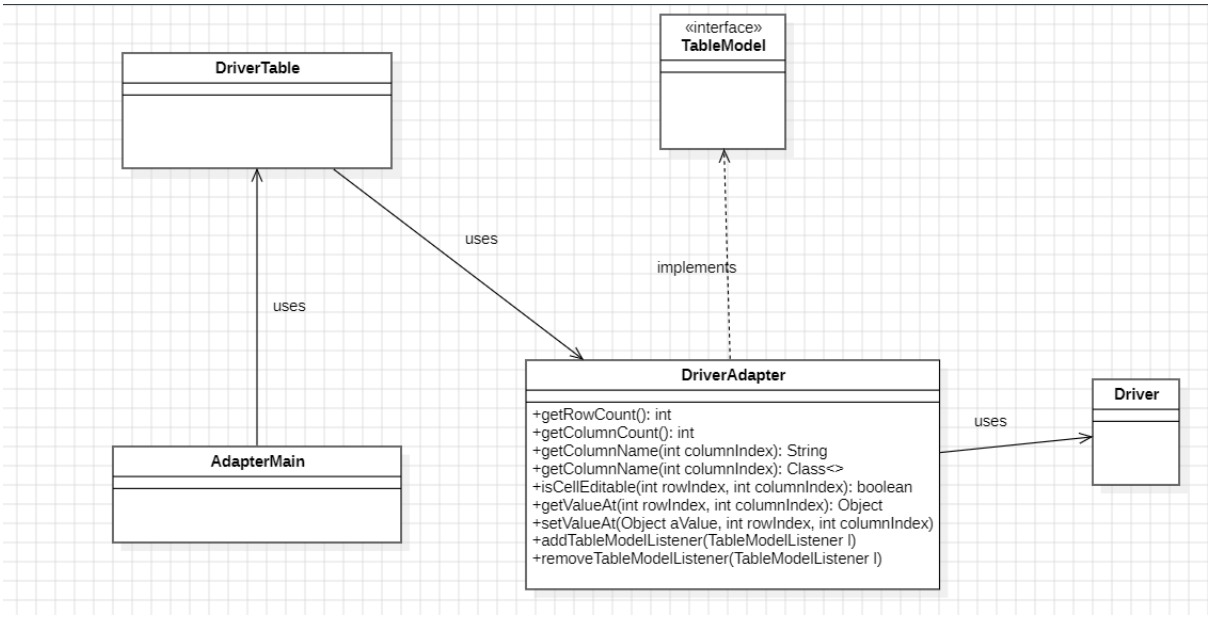
```
_____
FROM    LAST    TO      FIRST
Mandril
Madrid


_____
FROM    FIRST   TO      LAST
Madrid
Mandril
```

❖ ADAPTER PATROIA

Eskatzen da: DriverTable klasean deitzen den DriverAdapter klasea inplementatu, hasieran agertzen den taula aurkezteko (UML diseinua ere aurkeztu beharko da Adapter patroian parte hartzen duten klase guztiak aurkeztuz).

1. **UML diagrama hedatua, egin dituzun aldaketak aurkeztuz.**



2. **Aldatu duzun kodea, lerro garrantzitsuenak azalduz.**

Driver klasea tabla batera adaptatzen duen klasea, DriverAdapter, TablaModel interfazea implementatzen duena:

```
public class DriverAdapter implements TableModel {
	private Driver d;
	public String[] columnNames = new String[5];
	public DriverAdapter(Driver d) {
		this.d=d;
		columnNames[0]= "from";
		columnNames[1]= "to";
		columnNames[2]= "date";
		columnNames[3]= "places";
		columnNames[4]= "prices";
	}
	@Override
	public int getRowCount() {
		// TODO Auto-generated method stub
		return d.getRides().size();
	}
	@Override
	public int getColumnCount() {
		// TODO Auto-generated method stub
		return 5;
	}
	@Override
	public String getColumnName(int columnIndex) {
```

```
		// TODO Auto-generated method stub
		if(columnIndex<0 || columnIndex>4) {
			return "Null";
		}else {
			return columnNames[columnIndex];
		}
	}
	@Override
	public Class<?> getColumnClass(int columnIndex) {
		// TODO Auto-generated method stub
		if(columnIndex<=1) {
			return String.class;
		}else if(columnIndex==2){
			return Date.class;
		}else if(columnIndex==3){
			return Integer.class;
		}else if(columnIndex==4){
			return Float.class;
		}else {
			return null;
		}
	}
	@Override
	public boolean isCellEditable(int rowIndex, int columnIndex) {
		// TODO Auto-generated method stub
		return false;
	}
	@Override
	public Object getValueAt(int rowIndex, int columnIndex) {
		// TODO Auto-generated method stub

		if(rowIndex<this.getRowCount() && rowIndex>=0) {

			Ride r=d.getRides().get(rowIndex);
			if(columnIndex == 0) return r.getFrom();
			if(columnIndex == 1) return r.getTo();
			if(columnIndex == 2) return r.getDate();
			if(columnIndex == 3) return r.getnPlaces();
			if(columnIndex == 4) return r.getPrice();
		}
		return null;
	}
	@Override
	public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
		// TODO Auto-generated method stub

	}
	@Override
	public void addTableModelListener(TableModelListener l) {
		// TODO Auto-generated method stub

	}
```

```
	@Override
	public void removeTableModelListener(TableModelListener l) {
		// TODO Auto-generated method stub

	}
```

Adaptadoreak ongi funzionatzen duela konprobatzeko klasea:

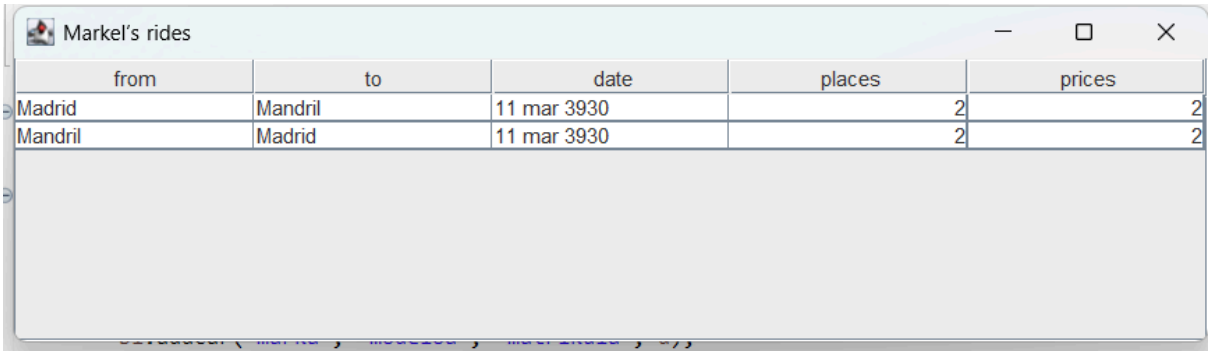
```
public class AdapterMain {
	public static void main(String[] args)	{
//			the	BL	is	local
	try {
		boolean isLocal = true;
		BLFacade bl = FactoryBLFacade.createBLFacade(1);
		Driver d= new Driver("Markel","a");
		bl.createDriver(d);
		bl.addCar("marka", "modeloa", "matrikula", d);
		Kotxea k= new Kotxea("marka", "modeloa", "matrikula");
		bl.createRide("Madrid", "Mandril", new Date(2030, 02, 11), 2, 2, "Markel",
k.toString());
		bl.createRide("Mandril", "Madrid", new Date(2030, 02, 11), 2, 2, "Markel",
k.toString());
		Driver d1= bl.badagoDriver("Markel");
		DriverTable dt=new DriverTable(d1);
		dt.setVisible(true);
	} catch (Exception e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
	}
}
```

Eta hau da tabla erakusten duen JFramea:

```
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);
	}
}
```

3. **Exekuzioaren irudi bat.**


Markel's rides
from
to
date
places
prices
Madrid
Mandril
11 mar 3930
2
2
Mandril
Madrid
11 mar 3930
2
2