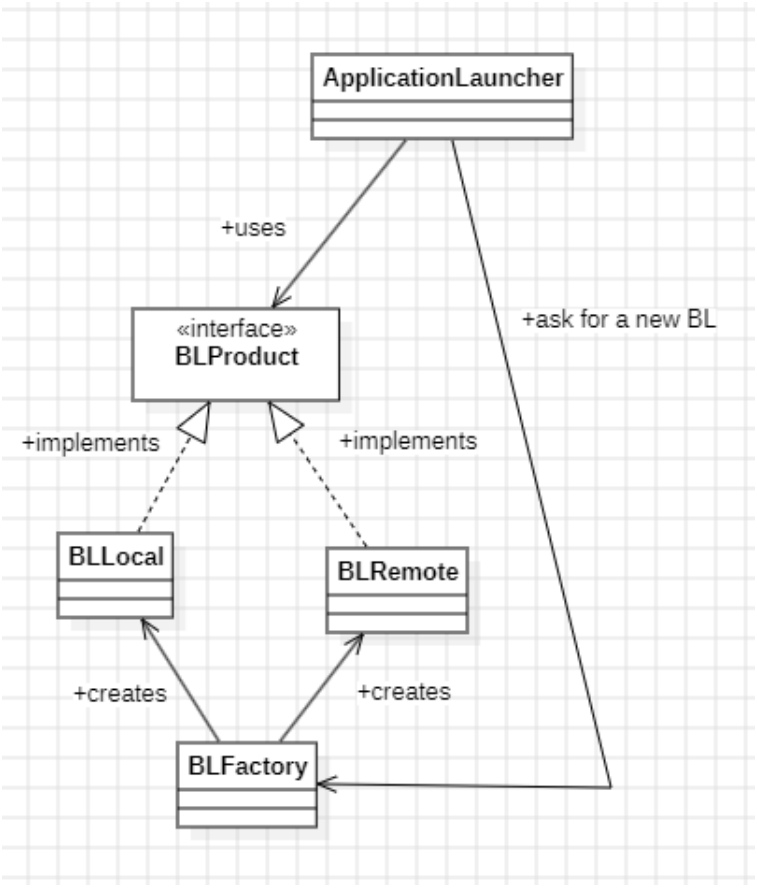
Proiektuaren diseinu patroiak

Egileak: **Ander Mena, Ekhi Ugarte, Haritz Eizagirre**
Data: **2025/11/15**
Irakaslea: Jon Iturrioz
Irakasgaia: Software Ingeniaritza II
Github errepositorioa: https://github.com/Eiza10/Rides24

1. Factory Method Patroia

a. UML diagrama



b. Aldatutako kodea

BLFactory klasea sortu dugu. Klase honek ConfigXML fitxategian bussinessLogicLocal true edo false izanik BLProduct interfaze berria implementatzen duten BLLocal edo BLRemote klase bat sortuko du.

BLFactory klasea

```
public class BLFactory {
  public BLProduct createBLProduct() {
     BLProduct product = null;
     ConfigXML config=ConfigXML.getInstance();
     if (config.isBusinessLogicLocal()) {
        product = new BLLocal(config);
     } else {
        try {
           product = new BLRemote(config);
        } catch (Exception e) {
           e.printStackTrace();
        }
     }
     return product;
```

```
  }
}
```

BLProduct klasea

```
public interface BLProduct {

  public BLFacade getBLFacade();
}
```

BLLocal klasea

```
public class BLLocal implements BLProduct {
  DataAccess dataAccess;
  BLFacade appFacadeInterface;
  public BLLocal(ConfigXML c) {
    dataAccess = new DataAccess();
                appFacadeInterface =new BLFacadeImplementation(dataAccess);
  }
  @Override
  public BLFacade getBLFacade() {
    return appFacadeInterface;
  }
}
```

BLRemote klasea

```
public class BLRemote implements BLProduct {
  BLFacade appFacadeInterface;

  // Implementation for remote business logic
  public BLRemote(ConfigXML config) throws Exception {
    // Constructor implementation
    String serviceName= "http://"+config.getBusinessLogicNode() +":"+
config.getBusinessLogicPort()+"/ws/"+config.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);
    appFacadeInterface = service.getPort(BLFacade.class);
  }
  @Override
  public BLFacade getBLFacade() {
    return appFacadeInterface;
  }
}
```

ApplicationLauncher klasea hasieran

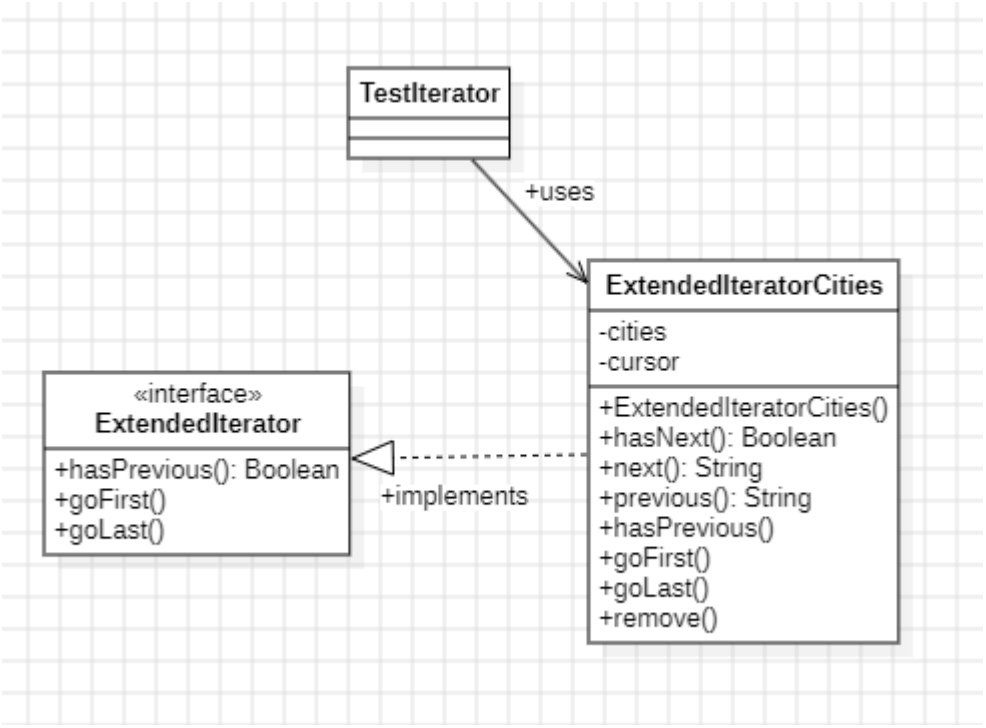
```
public class ApplicationLauncher {

	public static void main(String[] args) {
		ConfigXML config=ConfigXML.getInstance();

		System.out.println(config.getLocale());

		Locale.setDefault(new Locale(config.getLocale()));

		System.out.println("Locale: "+Locale.getDefault());

		MainGUI mainWindow=new MainGUI();
		mainWindow.setVisible(true);
		try {

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

			if (config.isBusinessLogicLocal()) {

				DataAccess dataAccess= new DataAccess();
				appFacadeInterface=new BLFacadeImplementation(dataAccess);

			}

			else { //If remote

				 String serviceName= "http://"+config.getBusinessLogicNode()
+":"+ config.getBusinessLogicPort()+"/ws/"+config.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);
			 appFacadeInterface = service.getPort(BLFacade.class);
			}

			MainGUI.setBussinessLogic(appFacadeInterface);


		}catch (Exception e) {
			mainWindow.jLabelWelcome.setText("Error: "+e.toString());
			mainWindow.jLabelWelcome.setForeground(Color.RED);

			System.out.println("Error in ApplicationLauncher: "+e.toString());
		}
		//a.pack();
	}
}
```

ApplicationLauncher klasea aldatuta

```
public class ApplicationLauncher {

	public static void main(String[] args) {
		ConfigXML config=ConfigXML.getInstance();

		System.out.println(config.getLocale());

		Locale.setDefault(new Locale(config.getLocale()));

		System.out.println("Locale: "+Locale.getDefault());

		MainGUI mainWindow=new MainGUI();
		mainWindow.setVisible(true);
		try {

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

			BLFactory factory = new BLFactory();
			BLProduct BLproduct = factory.createBLProduct();
			MainGUI.setBussinessLogic(BLproduct.getBLFacade());
		}catch (Exception e) {
			mainWindow.jLabelWelcome.setText("Error: "+e.toString());
			mainWindow.jLabelWelcome.setForeground(Color.RED);

			System.out.println("Error in ApplicationLauncher: "+e.toString());
		}
		//a.pack();
	}
}
```

2. Iterator patroia

a. UML diagrama



b. Aldatutako kodea

ExtendedIteratorCities klaseak ExtendedIterator interfaze berria implementatzen du eta TestIterator klaseko main metodoan probatzen da. ExtendedIteratorCities klaseak cities lista iteratzeko metodo batzuk ditu.

ExtendedIterator klasea

```
public interface ExtendedIterator<E> extends Iterator<E> {

   E previous();

   boolean hasPrevious();
   void goFirst();
   void goLast();
}
```

ExtendedIteratorCities klasea

```
public class ExtendedIteratorCities implements ExtendedIterator<String> {
        private final List<String> cities;
        /** Cursor points to the index of the next element to be returned by next() */
        private int cursor;
        /** Construct an iterator over a copy of the provided list (null treated as empty). */
        public ExtendedIteratorCities(List<String> cities) {
```

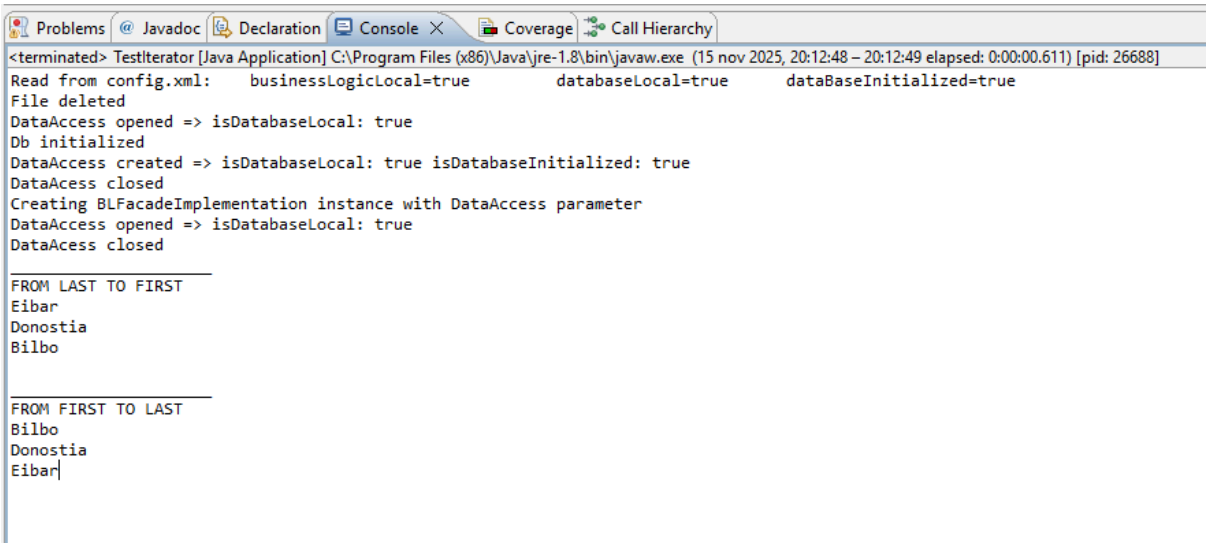
```
                if (cities == null) {
                        this.cities = new ArrayList<>();
                } else {
                        this.cities = new ArrayList<>(cities);
                }
                this.cursor = 0;
        }
        /** Construct an empty iterator. */
        public ExtendedIteratorCities() {
                this.cities = new ArrayList<>();
                this.cursor = 0;
        }
        @Override
        public boolean hasNext() {
                return cursor < cities.size();
        }
        @Override
        public String next() {
                if (!hasNext()) {
                        throw new NoSuchElementException();
                }
                return cities.get(cursor++);
        }
        @Override
        public String previous() {
                if (!hasPrevious()) {
                        throw new NoSuchElementException();
                }
                return cities.get(--cursor);
        }
        @Override
        public boolean hasPrevious() {
                return cursor > 0;
        }
        @Override
        public void goFirst() {
                cursor = 0;
        }
        @Override
        public void goLast() {
                // Position cursor so that previous() will return the last element
                cursor = cities.size();
        }
        @Override
        public void remove() {
                // Optional operation not supported for this simple iterator
                throw new UnsupportedOperationException("remove");
        }
}
```

TestIterator klasea

```
public class TestIterator {
        public static void main(String[] args) {
        // obtain BLFacade via factory
        BLFacade blFacade = new BLFactory().createBLProduct().getBLFacade();
        ExtendedIterator<String> i = new ExtendedIteratorCities(blFacade.getDepartCities());
                String c;
                System.out.println("_____");
```

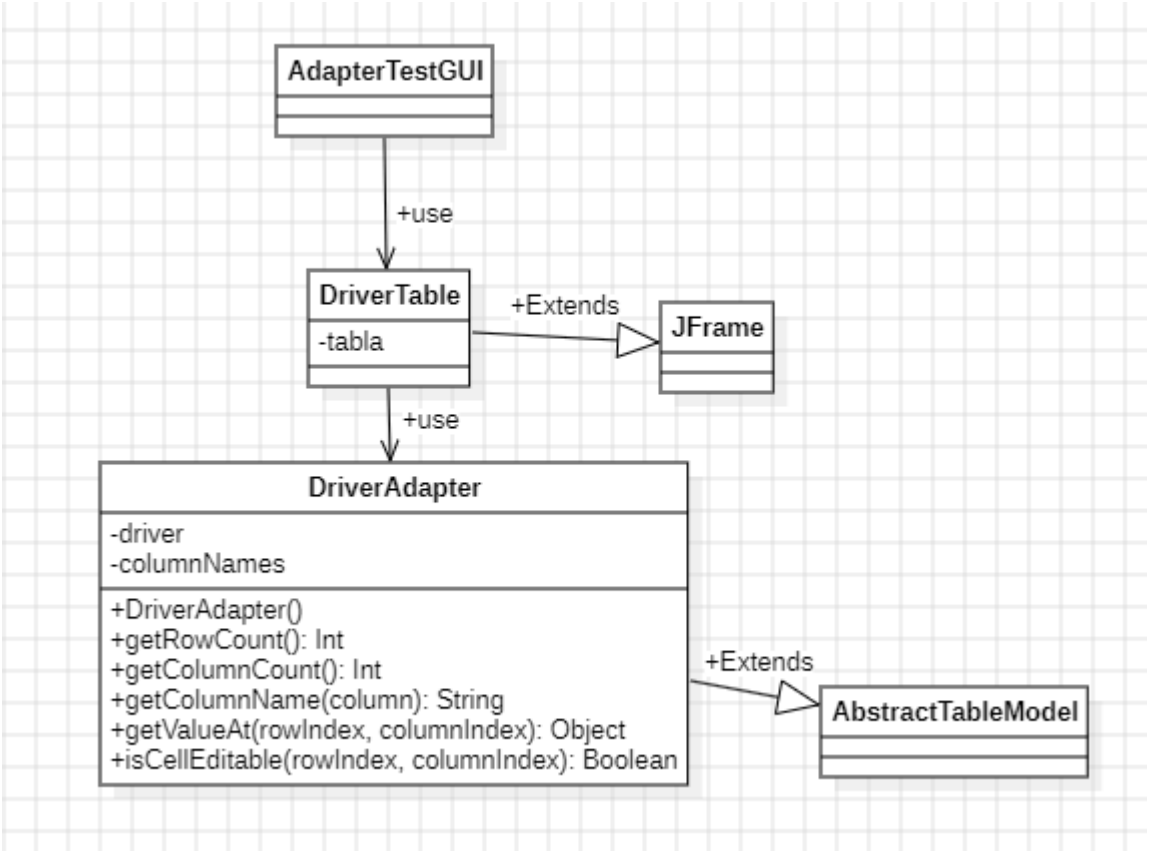
```
                System.out.println("FROM LAST TO FIRST");
                i.goLast(); // Go to last element
                while (i.hasPrevious()) {
                        c = i.previous();
                        System.out.println(c);
                }
                System.out.println();
                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);
                }
        }
}
```

c. Exekuzioa



3. Adapter Patroia

a. UML diagrama



b. Aldatutako kodea

DriverAdapter klase berri bat sortu dugu. Klase honek driver objektu bat emanda honen bidaien lista adaptatzen du (adapter patroia erabiliz) JTable formatuan datuak erakutsi ahal izateko.

Bestalde, DriverTable klase bat sortu dugu DriverAdapter klasea erabiliz gidari baten bidaien informazioaren tabla sortzeko.

Azkenik, AdapterTestGUI klasean main metodo bat sortu dugu aurrekoak probatzeko.

DriverAdapter klasea

```
public class DriverAdapter extends AbstractTableModel {

   private Driver driver;
   private String[] columnNames = new String[] {"Origin", "Destination", "Date", "Price", "nPlaces"};

   /**
    * Constructor that creates an adapter for the given driver
    * @param driver The driver whose rides will be displayed
    */
   public DriverAdapter(Driver driver) {
      this.driver = driver;
      System.out.println("DriverAdapter created for: " + driver.getEmail());
      System.out.println("Number of rides: " + (driver.getRides() != null ? driver.getRides().size() :
"NULL"));
   }

   @Override
   public int getRowCount() {
      if (driver == null || driver.getRides() == null) {
         return 0;
      }
      return driver.getRides().size();
   }

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

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

   @Override
   public Object getValueAt(int rowIndex, int columnIndex) {
      Ride ride = driver.getRides().get(rowIndex);

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

   @Override
```

```
    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return false;
    }
}
```

DriverTable klasea

```
public class DriverTable extends JFrame {
    private static final long serialVersionUID = 1L;
    private JTable tabla;
    public DriverTable(Driver driver) {
        super(driver.getEmail() + "'s rides");
        this.setBounds(100, 100, 700, 300);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        DriverAdapter adapt = new DriverAdapter(driver);
        tabla = new JTable(adapt);
        tabla.setPreferredScrollableViewportSize(new Dimension(650, 200));

        // Creamos un JscrollPane y le agregamos la JTable
        JScrollPane scrollPane = new JScrollPane(tabla);

        // Agregamos el JScrollPane al contenedor
        getContentPane().add(scrollPane, BorderLayout.CENTER);

        // Pack to fit content
        pack();
    }
}
```

AdapterTestGUI klasea

```
public class AdapterTestGUI {

    public static void main(String[] args) {
        try {
            // the BL is local
            BLFacade blFacade = new BLFactory().createBLProduct().getBLFacade();
            System.out.println("Facade created");

            Driver d = blFacade.getDriverByEmail("driver1@gmail.com", "123");
            System.out.println("Driver retrieved: " + d.getEmail());
            System.out.println("Driver name: " + d.getName());
            System.out.println("Number of rides: " + (d.getRides() != null ? d.getRides().size() :
"NULL"));

            if (d.getRides() != null && !d.getRides().isEmpty()) {
                System.out.println("Rides list:");
                for (int i = 0; i < d.getRides().size(); i++) {
                    System.out.println("  Ride " + (i+1) + ": " +
                        d.getRides().get(i).getFrom() + " -> " +
                        d.getRides().get(i).getTo());
                }
            }
```

```
            DriverTable dt = new DriverTable(d);
            dt.setVisible(true);
        } catch (UserDoesNotExistException | PasswordDoesNotMatchException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

c. Exekuzioa