SI2

Patroiak

Ioritz Tubío eta Unai Fraile

2023/11/12

- # Factory Method Patroia:

businessLogic paketean interfaze eta bi klase berri sortu ditugu.
Alde batetik, interfazea **BLFactory** sortu dugu BLFacade bat sortzeko bai lokalean edo remotoan.

Interfaze edo klase abstractu bat izan behar da metodo bera behar dugulako bi funtzio desberdin gauzatzeko.

Produktua sortzen duenez, BLFacade Creator izango da.

BLFactory-n erabilitako kodea:

```
public interface BLFactory {
	public BLFacade createBLFacade (ConfigXML c);

}
```

Bestalde, gure helburua lortzeko BusinessLogicLocal eta BusinessLogicRemote klaseak sortu ditugu. Izenak esaten duen bezala, bi kasu desberdinetan BLFacadea sortzeko erabiliko ditugu.

Orain negozio logikako objektuaren lorpena faktoria objektu batean zentralizatuta dago.

Behar izan dugun kodea:

```
public class BusinessLogicLocal implements BLFactory{
	public BLFacade createBLFacade (ConfigXML c) {
		BLFacade appFacadeInterface;
		DataAccess da = new DataAccess(c.getDataBaseOpenMode().
				equals("initialize"));
		appFacadeInterface=new BLFacadeImplementation(da);
		return appFacadeInterface;
	}
}
```

```
public class BusinessLogicRemote implements BLFactory{
	public BLFacade createBLFacade (ConfigXML c) {
		try {
			BLFacade appFacadeInterface;
			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("Ezin izan da sortu!!");
			return null;
		}
	}
}
```

Aplication Launcherren try catch bat sortu dugu bertan negozio logikoa lokalean edo remoton sortzeko.

```
try {
	BLFacade appFacadeInterface;
	BLFactory blFactory;
	UIManager.setLookAndFeel
		("javax.swing.plaf.metal.MetalLookAndFeel");

	if (c.isBusinessLogicLocal()) {
		blFactory = new BusinessLogicLocal();
	}
	else { // If remote
		blFactory = new BusinessLogicRemote();
	}
	appFacadeInterface = blFactory.createBLFacade(c);
	MainGUI.setBussinessLogic(appFacadeInterface);
} catch (Exception e) {
	a.jLabelSelectOption.setText("Error: " + e.toString());
	a.jLabelSelectOption.setForeground(Color.RED);
	System.out.println("Error in ApplicationLauncher: "
				+ e.toString());
}
```

- Iterator patroia:

Lehenik eta behin klase hau sortu dugu:

```
public interface ExtendedIterator<Event> extends Iterator {
	public Event previous();
	public boolean hasPrevious();
	public void goFirst();
	public void goLast();
}
```

Honek GUI erabiliko du, hortaz beste klase hau implementatu dugu, extendedIterator implementatuko duena:

```
public class ExtendedIteratorEvents implements
  ExtendedIterator<Event> {
	List<Event> gertaerak;
	int pos;
	public ExtendedIteratorEvents(List<Event> gertaerak) {
		this.gertaerak = gertaerak;
		this.pos =0;
	}
	public Event previous() {
		Event prev = gertaerak.get(pos);
		pos--;
		return prev;
	}
	public boolean hasPrevious() {
		if (pos >= 0)
			return true;
		return false;
	}
	public void goFirst() {
		if (gertaerak.size() > 0)
			pos = 0;
	}
	public void goLast() {
		if (gertaerak.size() > 0)
			pos = gertaerak.size() - 1;
	}
	public boolean hasNext() {
		return pos < gertaerak.size();
	}
	public Event next() {
		Event ev = gertaerak.get(pos);
		pos++;
		return ev;
	}
}
```

Segidan dataAccess klasearen getEvents() metodoaren itzulere-parametroa aldatu behar izan dugu, hona hemen BLFacadeImplementation klasean egindako aldaketa:

```
@WebMethod
public ExtendedIterator<Event> getEvents(Date date)  {
	dbManager.open(false);
	ExtendedIterator<Event> events=dbManager.getEvents(date);
	dbManager.close();
	return events;
}
```

Metodoa vector-etik ExtendedIteratorEvents-era pasatzeagatik GUI paketeko beste GUIak adaptatu behar izan ditugu ere bai.

```
ExtendedIteratorEvents events =
		(ExtendedIteratorEvents) facade.getEvents(firstDay);

while (events.hasPrevious()) {

		modelEvents.addElement(events.previous());
}
```

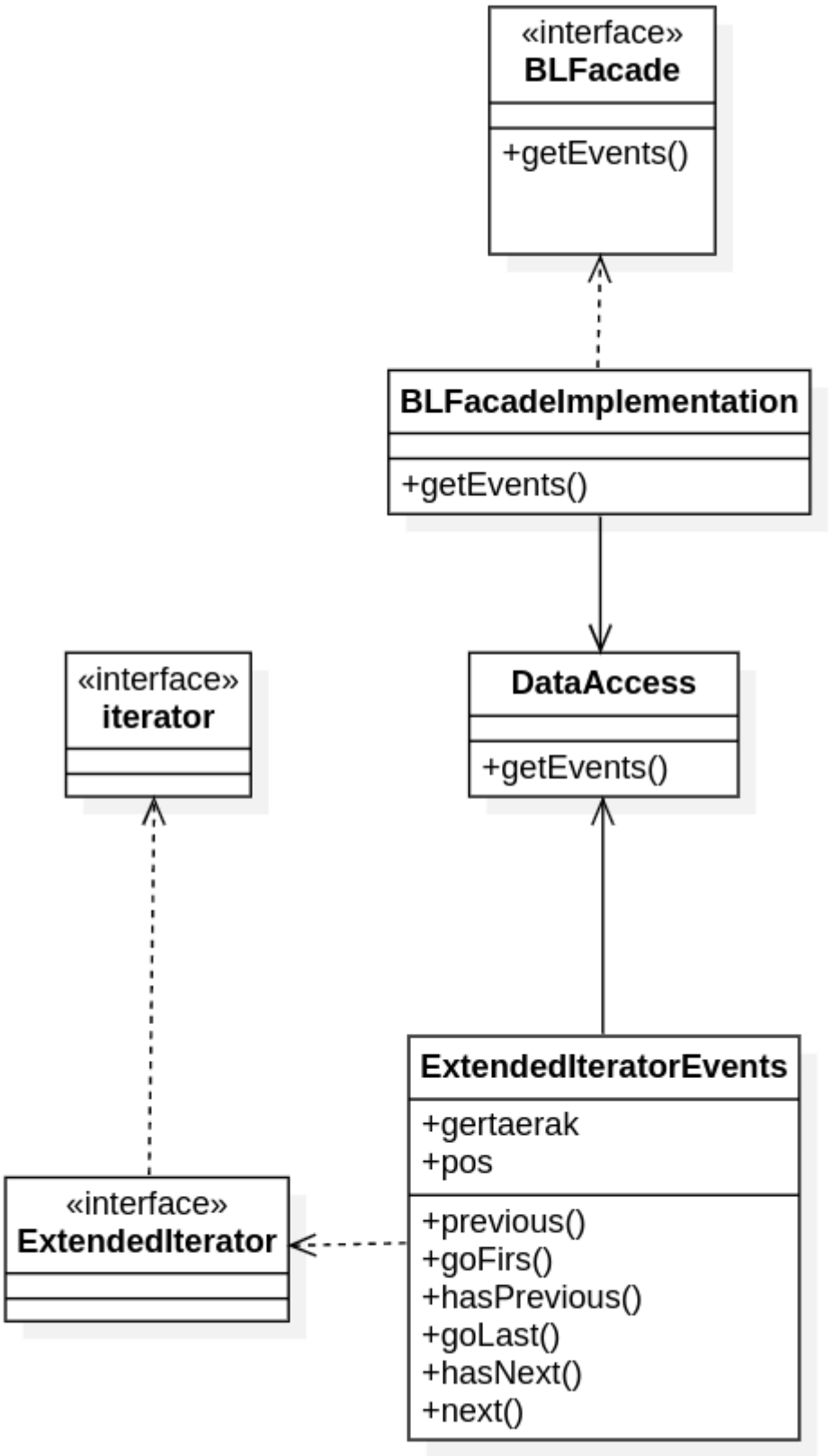
Azkenik implementazio honen programa bat eskatzen da, honen adibide gixa, eta hau egin dugu:

```
public class Adibidea {
	public static void main(String[] args) {
		boolean isLocal=true;
		BLFactory factory;
		factory = new BusinessLogicLocal();
		ConfigXML c = ConfigXML.getInstance();
		BLFacade facadeInterface= factory.createBLFacade(c);
		SimpleDateFormat sdf
				= new SimpleDateFormat("dd/MM/yyyy");
		java.util.Date data;
		try {
			data = sdf.parse("17/12/2023");
			ExtendedIterator<Event> ite
			   = facadeInterface.getEvents(data);
			Event ev;
			System.out.println
		("***************ATZERAKA ****************");
			ite.goLast();
			while(ite.hasPrevious()) {
				ev = ite.previous();
				System.out.println(ev.toString());
			}
			System.out.println("\n");
			System.out.println
		("***************AURRERAKA****************");
			ite.goFirst();
			while(ite.hasNext()) {
				ev = (Event) ite.next();
				System.out.println(ev.toString());
			}
		}catch(ParseException e) {
			System.out.println("errorea");
		}
	}
}
```

Eta hau litzateke adibide honen emaitza:

```
*************** ATZERAKA ****************
    27;Djokovic-Federer
    24;Miami Heat-Chicago Bulls
    23;Atlanta Hawks-Houston Rockets
    22;LA Lakers-Phoenix Suns
    10;Betis-Real Madrid
    9;Real Sociedad-Levante
    8;Girona-Leganes
    7;Malaga-Valencia
    6;Las Palmas-Sevilla
    5;Espanol-Villareal
    4;Alaves-Deportivo
    3;Getafe-Celta
    2;Eibar-Barcelona
    1;Atletico-Athletic
*************** AURRERAKA ****************
    1;Atletico-Athletic
    2;Eibar-Barcelona
    3;Getafe-Celta
    4;Alaves-Deportivo
    5;Espanol-Villareal
    6;Las Palmas-Sevilla
    7;Malaga-Valencia
    8;Girona-Leganes
    9;Real Sociedad-Levante
    10;Betis-Real Madrid
    22;LA Lakers-Phoenix Suns
    23;Atlanta Hawks-Houston Rockets
    24;Miami Heat-Chicago Bulls
    27;Djokovic-Federer
```

Implementatutako UMLa:



- Adapter Patroia:

Lehenik RegisteredAdapter deituriko klasea sortu dugu patroia sortzeko, hau da implementatu duguna:

```
public class RegisteredAdapter extends AbstractTableModel {
	private static final long serialVersionUID = 1L;
	protected Registered erabiltzailea;
	protected String[] zutabeenInfo = new String[] {
		ResourceBundle.getBundle("Etiquetas").getString("Event"),
		ResourceBundle.getBundle("Etiquetas").getString("Queries"),
		ResourceBundle.getBundle("Etiquetas").getString("Date"),
		ResourceBundle.getBundle("Etiquetas")
			.getString("MainTitle") + "(€)"
	};
	public RegisteredAdapter(Registered erabiltzailea) {
		this.erabiltzailea = erabiltzailea;
	}
	public String getColumnName(int columnIndex) {
		return zutabeenInfo[columnIndex];
	}
	public int getColumnCount() {
		return 4;
	}
	public int getRowCount() {
		return erabiltzailea.getApustuAnitzak().size();
	}
	public String getValueAt(int errenI, int zutabeI) {
		Apustua bet = getBetInTable(errenI);
		if (zutabeI == 0) {
			return bet.getKuota().getQuestion().
				getEvent().getDescription();
		} else if (zutabeI == 1) {
			return bet.getKuota().getQuestion().getQuestion();

		} else if (zutabeI == 2) {
			return bet.getApustuAnitza().getData()
			    .toLocaleString();

		} else if (zutabeI == 3) {
			return bet.getKuota().getQuote().toString();
		} else {
			return null;
		}
	}



	public Apustua getBetInTable(int errenI) {
```

```
                int totalBets = 0;
                for (ApustuAnitza apustuAnitza
                    : erabiltzailea.getApustuAnitzak()) {
                      for (Apustua apustua : apustuAnitza.getApustuak()) {
                            if (totalBets == errenI) {
                                  return apustua;
                            }
                            totalBets++;
                      }
                }
                return null;
          }
    }
```

Eta hau tauletan gordeko dugu, hona hemen:

```
    public class TableBetsGUI extends JFrame {
          private JTable taula;
          private Registered erabiltzaile;
          public TableBetsGUI(Registered erabiltzaile) {
                super(ResourceBundle.getBundle("Etiquetas").
                getString("TableUser")+ erabiltzaile.getUsername());
                this.setBounds(100, 100, 700, 400);
                this.erabiltzaile = erabiltzaile;
                RegisteredAdapter adapter =
                     new RegisteredAdapter(erabiltzaile);
                taula = new JTable(adapter);
                taula.setPreferredScrollableViewportSize
                      (new Dimension(500, 70));
                JScrollPane scrollPane = new JScrollPane(taula);
                getContentPane().add(scrollPane, BorderLayout.CENTER);
          }
    }
```

Azkenik funtzio berri honetara sartu ahal izateko MainGUI klasean botoi bat gehitu diogu:

```
private JButton getBtnAdapter(Registered u) {
	if(jButtonAdapter == null) {
		jButtonAdapter = new
		JButton(ResourceBundle.getBundle("Etiquetas")
		   .getString("TableUser") + " " + u.getUsername()   );
		jButtonAdapter.setFont(new Font
				("Tahoma", Font.PLAIN, 16));
		jButtonAdapter.setForeground(Color.DARK_GRAY);
		jButtonAdapter.setBackground(Color.PINK);
		jButtonAdapter.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				JFrame a =new TableBetsGUI(user);
				a.setVisible(true);
			}
		});
		jButtonAdapter.setBounds(10, 391, 282, 68);
	}
	return jButtonAdapter;
}
```

Horrela geratu da: