Errefaktorizazio laborategia:

SI2 : Software Ingenieritza II

https://github.com/MrAsieru/SI2_Bets21

Indizea:

"Write short units of code" edo Long Method . (2. kapituloa) [Mikel]

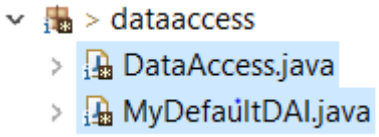
1. Hasierako kodea:

```
public void initializeDB() {

	db.getTransaction().begin();
	try {

		Calendar today = Calendar.getInstance();
		int month = today.get(Calendar.MONTH);
		month += 1;
		int year = today.get(Calendar.YEAR);
		if (month == 12) {
			month = 0;
			year += 1;
		}

		Event ev1 = new Event(1,"Atlï¿1/2tico-Athletic",
UtilDate.newDate(year, month, 17));
			Event ev2 = new Event(2, "Eibar-Barcelona",
UtilDate.newDate(year, month, 17));
			Event ev3 = new Event(3, "Getafe-Celta",
UtilDate.newDate(year, month, 17));

			... // Beste 195 kode lerro
}
		db.persist(pronos12);
		db.persist(pronos13);
		db.persist(pronos14);
		db.persist(pronos15);
		db.persist(pronos16);
		db.persist(pronos17);

		db.getTransaction().commit();
		System.out.println("Db initialized");
	} catch (Exception e) {
		e.printStackTrace();
	}
}
```

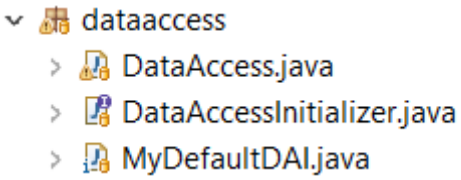
DataAccess-eko metodorik luzeena “initializeDB()” mododoa da. Metodo honek 222 kode lerro ditu eta zehaztutako 15 lerro muga guztiz gainditzen du. Horretako errefaktorizazio teknikak erabiliko ditugu. Erabili diren teknikak hurrengoak dira: “Extract to Method”, “Extract to Class”, “Extract to Interface” eta “Move Method”.

2. Kodea errefaktorizatu:

Hasteko InitializeDB metodoaren funtzioa DataAcces klasearekin zer ikusia duela hauteman daiteke baina egokiagoa izango litzateke datu basearen hasieraketa eta datu basearen moldaketa logika banatzea. Horretarako, klase berri bat sortzea pentsatu da MyDefaultDAI (Default Data Access Initializer) deitutakoa. Klase honek metodo bakarra izango du “initializeDB” deitutakoa. Metodo honek DataAccess objaktuaren EntityManager-a jasoko du, horrela, metodoak datu basean aldaketa efektiboak egin ahal izango ditu.



Beste hasieraketa mota bat egin nahiko balitz, unean errefaktorizatutako diseinuak ez luke funtzionalitate hori aldatzea onartuko, MyDefaultDAI klasea guztiz aldatu behar izango genukeelako. Aurrekoa era erraz batean egin ahal izateko, errefaktorizazio teknikak erabiliz, gure MyDefaultDAI klasea interfaze batera extraituko dugu (“Extract Interface”): DataAccessInitializer. MyDefaultDAI klaseak aurreko interfazea inplementatuko du. Gainara, DataAccess klasean DataAccessInitializer motako atributu bat gehituko dugu eta bertan MyDefaultDAI instantzia bat sortuko dugu. Era honetan, beste hasieraketa mota bat egin nahiko balitz, DataAccessInitializer interfazea inplementatzen duen beste hasieraketa klase bat sortu ahalko litzeteke aurreko kodean aldaketa handirik egin gabe.



DataAccess objektuaren “initializeDB” medoari deitzen zaionean, metodo honek MyDefaultDAI klasearen initializeDB metodoa deituko du eta objektu honen metodoak datu basea hasieratuko du. MyDefaultDAI klasearen initializeDB metodoa definitzeko “Move Method” errefaktorizazio teknika erabili dugu, eta honek DataAccess objektuaren “initializeDB” metodoaren eduki guztia MyDefaultDAI klasearen “initializeDB” metodora mugitu egin du.

- **DataAccess klase errefaktorizatua:**

```
public class DataAccess {
      protected EntityManager db;
```

```
	protected EntityManagerFactory emf;
	protected DataAccessInitializer defaultInitializer = new
MyDefaultDAI();
	...
	/* This is the data access method that initializes the database
with some events
	 * and questions. This method is invoked by the business logic
(constructor of
	 * BLFacadeImplementation) when the option "initialize" is
declared in the tag
	 * dataBaseOpenMode of resources/config.xml file
	 */
	public void initializeDB() {
		defaultInitializer.initializeDB(db);
	}
	...
}
```

- **"Extract Interface" bitartez lortutako DataAccessInitializer Interfazea:**

```
public interface DataAccessInitializer {

	void initializeDB(EntityManager entityManagerDB);

}
```

- **"Move Method"-en laguntzarekin sortutako MyDefaultDAI hasieraketa klasea:**

```
public class MyDefaultDAI implements DataAccessInitializer {

	@Override
	public void initializeDB(EntityManager entityManagerDB) {

		entityManagerDB.getTransaction().begin();
		try {
		... // 222 kode lerro
	}
}
```

Aurreko errefaktorizazioak aplikatu ondoren gure programa lehen bezala lan egingo du egindako errefaktorizazioek metodoen signadura aldatu ez dutelako eta, hortaz, beste klaseetako kodeetan (Testak, Metodoak, …) ez dute eraginik izan, hau da, ez da beste koderik aldatu behar izango DataAccess klasetik kanpo.

Errefaktorizazio hauek aplikatuta ere, oraindik DefaultDAI klasean dagoen "initializeDB(EntityManager)" metodoak lehen zituen beste kode lerro ditu (200 inguru lerro). Hau ebazteko eta 15 kode lerroen mugan sartzen saiatzeko, "initializeDB" metodoa hainbat

azpi-metodo pribatuetan banatuko dugu. Metodo bakoitzak helburu zehatz bat izango du. Hau era erraz batean gauzatzeko eclipsek eskaintzen duen “Extract Method” errefaktorizazioa erabiliko dugu. Era honentan funtzio ezberdinak dituzten kode unitateak metodo ezberdinetan banatu ahal izango ditugu.
Aipatzekoa da ere, metodoen luzeera eta kodearen errepikapena (erredundantzia) sahiesteko DB-an gehitu nahi den entitate bakoitzeko Hiztegi (Map) bat sortzea erabaki dugula. Era honetan, bukle baten bitartez Hiztegi bakoitzeko kolekzioan dauden entitate guztietan “persist(Object)” metodoa era erraz, sinple eta iteratibo batean deitu ahal izago da.

- **“initializeDB” Metodo ezberdinetan banatzeko DefaultDAI-en beharrezkoak izan diren atributuak (Hiztegiak)**:

```
public class MyDefaultDAI implements DataAccessInitializer {

	private Map<String, Event> eventDictionary = new HashMap<>();
	private Map<String, Question> questionDictionary = new
HashMap<>();
	private Map<String, Pertsona> pertsonaDictionary = new
HashMap<>();
	private Map<String, Pronostikoa> pronostikoDictionary = new
HashMap<>();
	private Map<String, Mugimendua> mugimenduaDictionary = new
HashMap<>();
	...
}
```

- **“Extract Method” erabili ostean lortutako initializeDB metodoa (11 lerro):**

```
@Override
public void initializeDB(EntityManager entityManagerDB) {
	entityManagerDB.getTransaction().begin();
	createEvents();
	createQuestionsAndSetRelationsWithEvents();
	createUsers();
	createPronostikoakAndCreateRelationsWithQuestions();
	createErrepikapenakAndCreateRelationsWithBezeroa();
	createApustuakAndCreateRalationsWithPronostikoakBezeroak();
	createMugimenduakAndCreateRelationsWithBezeroak();
	persistAllObjectsInsideDefinedDictionaries(entityManagerDB);
	entityManagerDB.getTransaction().commit();
	System.out.println("Db initialized");
}
```

- **“createEvents()” azpimetodoaren inplementazioa:**

```
private void createEvents() {
	int[] tdy = getTodayDateOnArrayYM();
```

```
      int y = tdy[0];
      int m = tdy[1];
      eventDictionary.put("ev1" , new Event(1,"Atlï¿1/2tico-Athletic",
UtilDate.newDate(y, m, 17)));
      eventDictionary.put("ev2" , new Event(2, "Eibar-Barcelona",
UtilDate.newDate(y, m, 17)));
      ... // 19 kode lerro gehiago
      eventDictionary.put("event1", new Event(21,"Eibar-Celta",
UtilDate.newDate(2021, 2, 17)));
            eventDictionary.put("event2", new
Event(22,"Granada-Athletic", UtilDate.newDate(2021, 2, 17)));
}
```

- **“persistAllObjectsInsideDefinedDictionaries()” azpimetodoaren inplementazioa:**

```
private void persistAllObjectsInsideDefinedDictionaries(EntityManager
pEntManagerDB) {
      persistAllObjectsInsideEventDictionary(pEntManagerDB);
      persistAllObjectsInsideQuestionDictionary(pEntManagerDB);
      persistAllObjectsInsidePertsonaDictionary(pEntManagerDB);
      persistAllObjectsInsidePronostikoDictionary(pEntManagerDB);
      persistAllObjectsInsideMugimenduaDictionary(pEntManagerDB);
}
```

- **“persistAllObjectsInsideEventDictionary()” azpimetodoaren inplementazioa:**

```
private void persistAllObjectsInsideEventDictionary(EntityManager
pEntManagerDB) {
      for(Object objectToPersist: this.eventDictionary.values()) {
            pEntManagerDB.persist(objectToPersist);
      }
}
```

Beste extraitutako metodoak ez dira dokumentu honetan zehaztuko dokumentua asko luzatuko litzatekeelako. Azkenik extraitutako metodo batzuek 15 lerro baino gehiago dituztela esan beharra dago. Uneko hasieraketa inplementazio erarekin, ezinezkoa da metodo hauek prozezu logiko baten bitartez beste metodo txikiago batzuetan banatzea. Banatu daitezke, bai, baina banaketa hauek kodea irakurtzen ari den programatzailea nahasteko bakarrik balioko dute.
Metodoen luzeera bai ala bai txikitu nahi bada, hasieraketa beste era batean egin beharko litzateke, adibidez: XML, TXT, DatuBase … dokumentu baten bitartez. Hasieraketa hau inplementatzeko dokumentu mota espezifikoa interpretatzeko gai den eta DataAccessInitializer iterfazea inplementatzen duen klase bat bakarrik sortu beharko litzateke eta ondoren DataAccess barnean instantziatu.

3. Bukaerako kodea:

Bukaerako kodea nahiko luzea denez, proiektuaren GitHub biltegian kodea ikustea gomendatzen da.

- DataAccess
- DataAccessInitializer
- MyDefaultDAI

"Write simple units of code" (3. kapituloa) [Unai]

1. Hasierako kodea:

```
Bezeroa erabiltzaile = db.find(Bezeroa.class,
bezero.getErabiltzaileIzena());
	Pronostikoa pronos;
	ArrayList<Pronostikoa> pronostikoSorta = new
ArrayList<Pronostikoa>();
	for(Pronostikoa p : pronostikoak) {
		pronos = db.find(Pronostikoa.class,
p.getIdentifikadorea());
		pronostikoSorta.add(pronos);
	}
	db.getTransaction().begin();
	Apustua apus = erabiltzaile.addApustua(pronostikoSorta, a, null);
	for(Pronostikoa p : pronostikoSorta) {
		p.addApustua(apus);
	}
	db.persist(apus);
	Vector<Errepikapena>
jarraitzaile=erabiltzaile.getErrepikatzaileak();
	for(Errepikapena er: jarraitzaile) {
		double apustudiru=0;
		if (er.getHilabeteHonetanGeratzenDena()>0) {
			if
(er.getHilabeteHonetanGeratzenDena()>=er.getApustatukoDena()*a) {
				apustudiru=er.getApustatukoDena()*a;
			} else {
				apustudiru=er.getHilabeteHonetanGeratzenDena();
			}
			if (er.getNork().getDirua() >= apustudiru) {
				Apustua apustu =
```

```
er.getNork().addApustua(pronostikoSorta, apustudiru, erabiltzaile);
                        for (Pronostikoa p : pronostikoSorta) {
                              p.addApustua(apustu);
                        }
                        er.getNork().addMugimendua("Apustu errepikatua
egin ("+erabiltzaile+")", -apustudiru, "jokatu");
                        er.eguneratuHilHonetanGeratzenDena(-apustudiru);
                        db.persist(apus);
                  }
            }
      }
      erabiltzaile.addMugimendua("Apustua egin ", -a, "jokatu");
      db.getTransaction().commit();
      return erabiltzaile;
```

2. Kodea errefaktorizatu egingo dugu:

Hasteko kode bloke hau hartu egingo dugu eta beste metodo batean sartuko dugu, horrela aurreko kodea murriztuko dugu eta ulergarritazuna hobetu. Beste aldetik sortu berri den metodoa berriro erabili ahalko dugu etorkizunean. Hau gauzatzeko metodo batean bihurtu nahi dugun kode zatia aukeratzen dugu eta Refactor -> Extract Method eginez metodo berria sortu ahalko dugu.

```
 for(Pronostikoa p : pronostikoak) {
            pronos = db.find(Pronostikoa.class,
p.getIdentifikadorea());
            pronostikoSorta.add(pronos);
      }
```

→

```
private void gehituPronostikoa(ArrayList<Pronostikoa> pronostikoak,
ArrayList<Pronostikoa> pronostikoSorta) {
      Pronostikoa pronos;
      for(Pronostikoa p : pronostikoak) {
            pronos = db.find(Pronostikoa.class,
p.getIdentifikadorea());
            pronostikoSorta.add(pronos);
      }
   }
```

Beste kode zati handi eta konplexu bat hau izan ahal da, eta beraz metodo batean kapsulatuko dugu, aurrekoan esan bezala, ulergarritazuna eta erabilgarritazuna hobetzeko. Aurrekoan bezala, metodo batean bihurtu nahi dugun kode zatia aukeratzen dugu eta Refactor -> Extract Method eginez metodo berria sortu ahalko dugu.

```
if (er.getHilabeteHonetanGeratzenDena()>0) {
            if
(er.getHilabeteHonetanGeratzenDena()>=er.getApustatukoDena()*a) {
                  apustudiru=er.getApustatukoDena()*a;
            } else {
                  apustudiru=er.getHilabeteHonetanGeratzenDena();
            }
            if (er.getNork().getDirua() >= apustudiru) {
                  Apustua apustu =
er.getNork().addApustua(pronostikoSorta, apustudiru, erabiltzaile);
                  for (Pronostikoa p : pronostikoSorta) {
                        p.addApustua(apustu);
                  }
                  er.getNork().addMugimendua("Apustu errepikatua egin
("+erabiltzaile+")", -apustudiru, "jokatu");
                  er.eguneratuHilHonetanGeratzenDena(-apustudiru);
                  db.persist(apus);
            }
      }
```

→

```
private void parametroarenAraberaProzesatu(double a, Bezeroa
erabiltzaile, ArrayList<Pronostikoa> pronostikoSorta,
            Apustua apus, Errepikapena er) {
      double apustudiru;
      if (er.getHilabeteHonetanGeratzenDena()>0) {
            if
(er.getHilabeteHonetanGeratzenDena()>=er.getApustatukoDena()*a) {
                  apustudiru=er.getApustatukoDena()*a;
            } else {
                  apustudiru=er.getHilabeteHonetanGeratzenDena();
            }
            if (er.getNork().getDirua() >= apustudiru) {
                  Apustua apustu =
er.getNork().addApustua(pronostikoSorta, apustudiru, erabiltzaile);
                  for (Pronostikoa p : pronostikoSorta) {
                        p.addApustua(apustu);
                  }
                  er.getNork().addMugimendua("Apustu errepikatua egin
("+erabiltzaile+")", -apustudiru, "jokatu");
                  er.eguneratuHilHonetanGeratzenDena(-apustudiru);
                  db.persist(apus);
            }
```

```
        }
    }
```

3. Bukaerakio apustuaEgin:

```
public Bezeroa apustuaEgin(ArrayList<Pronostikoa> pronostikoak, double
a, Bezeroa bezero) {
        Bezeroa erabiltzaile = db.find(Bezeroa.class,
bezero.getErabiltzaileIzena());
        Pronostikoa pronos;
        ArrayList<Pronostikoa> pronostikoSorta = new
ArrayList<Pronostikoa>();
        gehituPronostikoa(pronostikoak, pronostikoSorta);
        db.getTransaction().begin();
        Apustua apus = erabiltzaile.addApustua(pronostikoSorta, a, null);
        for(Pronostikoa p : pronostikoSorta) {
              p.addApustua(apus);
        }
        db.persist(apus);
        Vector<Errepikapena>
jarraitzaile=erabiltzaile.getErrepikatzaileak();
        for(Errepikapena er: jarraitzaile) {
              double apustudiru=0;
              parametroarenAraberaProzesatu(a, erabiltzaile,
pronostikoSorta, apus, er);
        }
        erabiltzaile.addMugimendua("Apustua egin ", -a, "jokatu");
        db.getTransaction().commit();
        return erabiltzaile;
    }
```

Duplicate code (gui.EmaitzaIpiniGUI) [Asier]

1. Hasierako kodea. jLabel dituzten 3 lerroak hainbat leku desberdinetan errepikatuta daude), adibidez:

```
jComboBoxEvents.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        jLabelError.setText("");
        jLabelErrorDate.setText("");
```

```
        jLabelSucces.setText("");

        questionModel.removeAllElements();
        pronosticModel.removeAllElements();
```

...

```
jComboBoxQuestions.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        jLabelError.setText("");
        jLabelErrorDate.setText("");
        jLabelSucces.setText("");
        pronosticModel.removeAllElements();
```

...

```
jButtonCreate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        jLabelError.setText("");
        jLabelErrorDate.setText("");
        jLabelSucces.setText("");
            if(selectedQuestion.getResult()!=null) {
               jLabelError.setText(ResourceBundle.getBundle("Etiquetas")
                    .getString("QuestionResult"));
            }else {
```

...

```
jComboBoxPronostics.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        jLabelError.setText("");
        jLabelErrorDate.setText("");
        jLabelSucces.setText("");

        selectedPronostic = ((domain.Pronostikoa)
            jComboBoxPronostics.getSelectedItem());
```

...

```
this.jCalendar.addPropertyChangeListener(new PropertyChangeListener() {
    public void propertyChange(PropertyChangeEvent propertychangeevent)
{
        jLabelError.setText("");
        jLabelErrorDate.setText("");
        jLabelSucces.setText("");
```

2. Errefaktorizatutako kodea

```
jComboBoxEvents.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
```

```
        labelGarbitu();

        questionModel.removeAllElements();
        pronosticModel.removeAllElements();
```

...

```
jComboBoxQuestions.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        labelGarbitu();
        pronosticModel.removeAllElements();
```

...

```
jButtonCreate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        labelGarbitu();
        if(selectedQuestion.getResult()!=null) {
            jLabelError.setText(ResourceBundle.getBundle("Etiquetas")
                .getString("QuestionResult"));
        } else {
```

...

```
jComboBoxPronostics.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        labelGarbitu();
        selectedPronostic = ((domain.Pronostikoa)
            jComboBoxPronostics.getSelectedItem());
```

...

```
this.jCalendar.addPropertyChangeListener(new PropertyChangeListener() {
    public void propertyChange(PropertyChangeEvent propertychangeevent){
        labelGarbitu();
```

...

```
private void labelGarbitu() {
    jLabelError.setText("");
    jLabelErrorDate.setText("");
    jLabelSucces.setText("");
}
```

3. Deskribapena

Errefaktorizazio hau gauzatzeko errepikatutako kodea aukeratuta Refactor -> Extract method erabil egin dut. Honek metodorako izen bat eta atzipen mota eskatzen du, eta gainera errepikapenak detektatzen ditu. Behin burututa fitxategiaren amaieran metodo berri bat sortzen da aukeratutako kodearekin eta kode hau agertzen den leku guztietan metodoaren deiarekin ordezkatzen da

"Keep unit interfaces small" edo Long paramater List (5. kapituloa)

1. Hasierako kodea. Metodo honetan 9 sarrera paremetro ditugu (gomendagarria da gehienez 4 izatea):

```
public Pertsona register(String izena, String abizena1, String abizena2,
String erabiltzaileIzena, String pasahitza, String telefonoZbkia, String
emaila, Date jaiotzeData, String mota) throws UserAlreadyExist{
            TypedQuery<Pertsona> query = db.createQuery("SELECT p FROM
Pertsona p WHERE p.erabiltzaileIzena=?1", Pertsona.class);
            query.setParameter(1, erabiltzaileIzena);
            List<Pertsona> pertsona = query.getResultList();
            if(!pertsona.isEmpty()) {
                  throw new UserAlreadyExist();
            }else {
                  Pertsona berria = null;
                  if(mota.equals("admin")) {
                        berria = new Admin(izena, abizena1, abizena2,
erabiltzaileIzena, pasahitza, telefonoZbkia, emaila, jaiotzeData);
                  }else if (mota.equals("langilea")) {
                        berria = new Langilea(izena, abizena1, abizena2,
erabiltzaileIzena, pasahitza, telefonoZbkia, emaila, jaiotzeData);
                  }else if (mota.equals("bezeroa")) {
                        berria = new Bezeroa(izena, abizena1, abizena2,
erabiltzaileIzena, pasahitza, telefonoZbkia, emaila, jaiotzeData);
                  }
                  db.getTransaction().begin();
                  db.persist(berria);
                  db.getTransaction().commit();
                  return berria;
            }
      }
```

2. Kode aldaketak:

Aldatu nahi dugun metodora goaz eta han metoda aukeratu eta Refactor -> Introduce Parameter Object… erabiliz parametro guzti hauek gordeko dituen klasea sortu egingo dugu. Hau eginez metodoak behar dituen sarrera parametroak klase batean enkapsulatu egingo ditugu eta metodoaren irakurketa sinplifikatu. Ala ere parametro hauek zeintzuk diren jakiteko klase berri batera joan beharko gara, prozesua astirotzen.

```
public class RegisterParameter {
	public String izena;
	public String abizena1;
	public String abizena2;
	public String erabiltzaileIzena;
	public String pasahitza;
	public String telefonoZbkia;
	public String emaila;
	public Date jaiotzeData;
	public String mota;

	public RegisterParameter(String izena, String abizena1, String
abizena2, String erabiltzaileIzena, String pasahitza,
			String telefonoZbkia, String emaila, Date jaiotzeData,
String mota) {
		this.izena = izena;
		this.abizena1 = abizena1;
		this.abizena2 = abizena2;
		this.erabiltzaileIzena = erabiltzaileIzena;
		this.pasahitza = pasahitza;
		this.telefonoZbkia = telefonoZbkia;
		this.emaila = emaila;
		this.jaiotzeData = jaiotzeData;
		this.mota = mota;
	}
}
```

Klase hau erabiltzeko parametrodun metodoa aldatu beharko dugu:

```
public Pertsona register(RegisterParameter parameterObject) throws
UserAlreadyExist{
		TypedQuery<Pertsona> query = db.createQuery("SELECT p FROM
Pertsona p WHERE p.erabiltzaileIzena=?1", Pertsona.class);
		query.setParameter(1, parameterObject.erabiltzaileIzena);
		List<Pertsona> pertsona = query.getResultList();
		if(!pertsona.isEmpty()) {
			throw new UserAlreadyExist();
		}else {
			Pertsona berria = null;
			if(parameterObject.mota.equals("admin")) {
				berria = new Admin(parameterObject.izena,
parameterObject.abizena1, parameterObject.abizena2,
parameterObject.erabiltzaileIzena, parameterObject.pasahitza,
```

```
parameterObject.telefonoZbkia, parameterObject.emaila,
parameterObject.jaiotzeData);
                }else if (parameterObject.mota.equals("langilea")) {
                      berria = new Langilea(parameterObject.izena,
parameterObject.abizena1, parameterObject.abizena2,
parameterObject.erabiltzaileIzena, parameterObject.pasahitza,
parameterObject.telefonoZbkia, parameterObject.emaila,
parameterObject.jaiotzeData);
                }else if (parameterObject.mota.equals("bezeroa")) {
                      berria = new Bezeroa(parameterObject.izena,
parameterObject.abizena1, parameterObject.abizena2,
parameterObject.erabiltzaileIzena, parameterObject.pasahitza,
parameterObject.telefonoZbkia, parameterObject.emaila,
parameterObject.jaiotzeData);
                }
                db.getTransaction().begin();
                db.persist(berria);
                db.getTransaction().commit();
                return berria;
          }
     }
```

3. Bukaerako kodea:

```
public Pertsona register(RegisterParameter parameterObject) throws
UserAlreadyExist{
          TypedQuery<Pertsona> query = db.createQuery("SELECT p FROM
Pertsona p WHERE p.erabiltzaileIzena=?1", Pertsona.class);
          query.setParameter(1, parameterObject.erabiltzaileIzena);
          List<Pertsona> pertsona = query.getResultList();
          if(!pertsona.isEmpty()) {
                throw new UserAlreadyExist();
          }else {
                Pertsona berria = null;
                if(parameterObject.mota.equals("admin")) {
                      berria = new Admin(parameterObject.izena,
parameterObject.abizena1, parameterObject.abizena2,
parameterObject.erabiltzaileIzena, parameterObject.pasahitza,
parameterObject.telefonoZbkia, parameterObject.emaila,
parameterObject.jaiotzeData);
                }else if (parameterObject.mota.equals("langilea")) {
                      berria = new Langilea(parameterObject.izena,
parameterObject.abizena1, parameterObject.abizena2,
parameterObject.erabiltzaileIzena, parameterObject.pasahitza,
```

```
parameterObject.telefonoZbkia, parameterObject.emaila,
parameterObject.jaiotzeData);
                }else if (parameterObject.mota.equals("bezeroa")) {
                      berria = new Bezeroa(parameterObject.izena,
parameterObject.abizena1, parameterObject.abizena2,
parameterObject.erabiltzaileIzena, parameterObject.pasahitza,
parameterObject.telefonoZbkia, parameterObject.emaila,
parameterObject.jaiotzeData);
                }
                db.getTransaction().begin();
                db.persist(berria);
                db.getTransaction().commit();
                return berria;
          }
     }
```

```
public class RegisterParameter {
      public String izena;
      public String abizena1;
      public String abizena2;
      public String erabiltzaileIzena;
      public String pasahitza;
      public String telefonoZbkia;
      public String emaila;
      public Date jaiotzeData;
      public String mota;

      public RegisterParameter(String izena, String abizena1, String
abizena2, String erabiltzaileIzena, String pasahitza,
                String telefonoZbkia, String emaila, Date jaiotzeData,
String mota) {
            this.izena = izena;
            this.abizena1 = abizena1;
            this.abizena2 = abizena2;
            this.erabiltzaileIzena = erabiltzaileIzena;
            this.pasahitza = pasahitza;
            this.telefonoZbkia = telefonoZbkia;
            this.emaila = emaila;
            this.jaiotzeData = jaiotzeData;
            this.mota = mota;
      }
}
```