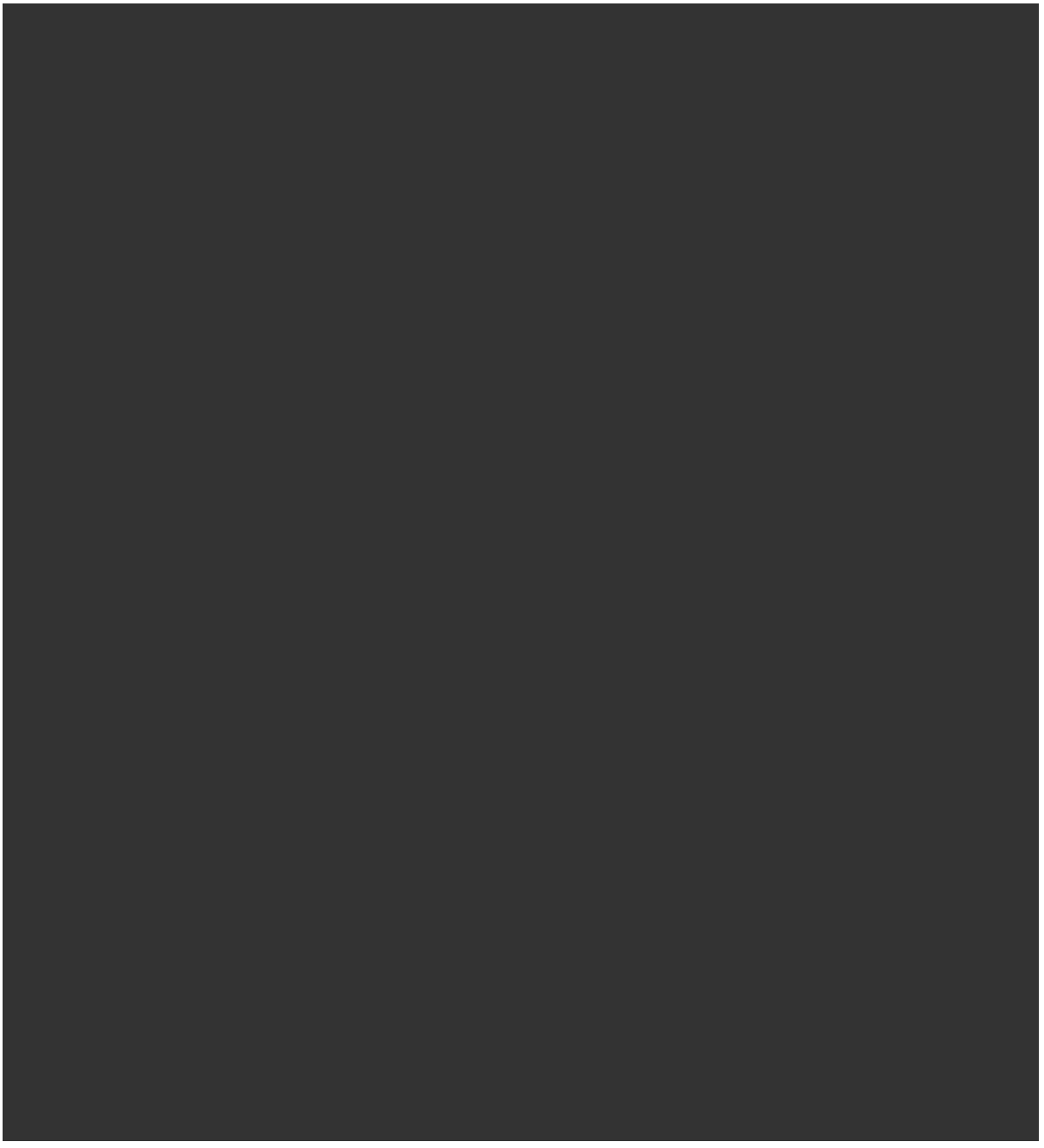
ERREFAKTORIZAZIOA

Aritz Azkunaga eta Jon Jauregi

Github-eko linka: https://github.com/Azkunaga/SI2

WRITE SHORT UNITS OF CODE

Errefaktorizazio honen helburua da metodo baten lerro exekuzioa gutxitzea. Metodo baten kodea zuzen egoteko 15 lerro edo gutxiago izan beharko ditu. Horretarako, “**Extract Method**” teknika erabiliko dugu, hau da, metodoaren barruan dauden kode zati batzuk hortik atera, beste metodo bat sortu tera



Errefakturizatutako kodea

```
public void emaitzaIpini(Event ev, Question q, Pronostikoa pi) {
		boolean ordaindu = true;
		Question qi = db.find(Question.class, q.getQuestionNumber());
		Pronostikoa p = db.find(Pronostikoa.class, pi.getPronostikoaNumber());
		qi.setResult(p);
		db.getTransaction().begin();
		db.persist(qi);
		db.getTransaction().commit();
		Vector<Apustua> ap = p.getApustuak();
		apustuakOrdaindu(ordaindu, ap);

	}

	private void apustuakOrdaindu(boolean ordaindu, Vector<Apustua> ap) {
		for (Apustua api : ap) {
			Bezero b = db.find(Bezero.class,
api.getBezeroa().getErabiltzailea());
			Float kuota = api.getKuota();
			Float apustuDirua = api.getApustuDirua();
			Vector<Pronostikoa> pronostikoak = api.getPronostikoak();
			ordaindu = pronostikoakKonparatu(ordaindu, pronostikoak);
			ordaindu(ordaindu, api, b, kuota, apustuDirua);
		}
	}

	private void ordaindu(boolean ordaindu, Apustua api, Bezero b, Float kuota, Float
apustuDirua) {
		if (ordaindu) {
			b.addDirua(kuota * apustuDirua);
			b.addMugimendua(kuota * apustuDirua,
ResourceBundle.getBundle("Etiquetas").getString("Win"), false);
			db.getTransaction().begin();
			db.persist(b);
			db.getTransaction().commit();
			Bezero jabea = api.getJabea();
			jabeaDa(b, kuota, apustuDirua, jabea);

		}
	}

	private void jabeaDa(Bezero b, Float kuota, Float apustuDirua, Bezero jabea) {
		if (jabea != null) {
			Bezero aur = db.find(Bezero.class, jabea.getErabiltzailea());
			aur.addDirua((float) (kuota * apustuDirua * 0.1));
			aur.addMugimendua((float) (kuota * apustuDirua * 0.1),

ResourceBundle.getBundle("Etiquetas").getString("WinThanksTo") + b.getErabiltzailea(),
					true);
			db.getTransaction().begin();
			db.persist(aur);
			db.getTransaction().commit();
		}
	}
```

```
	private boolean pronostikoakKonparatu(boolean ordaindu, Vector<Pronostikoa>
pronostikoak) {
		for (Pronostikoa pr : pronostikoak) {
			Question qr = pr.getQ();
			if (!pr.equals(qr.getResult())) {
				ordaindu = false;
			}
		}
		return ordaindu;
	}
```

Deskribapena

Metodo hau oso luzea denez, hainbat zatitan banatu dut metodoa. Hain zuzen ere, lau metodo gehiago sortuz: apustuakOrdaindu, ordaindu, jabeaDa eta pronostikoakKonparatu.

- **jabeaDa**: metodo honek egiaztatzen du egindako apustuaren bezeroa bere apustuaren jabea dela. Ez bada jabea, ez du ezer egiten. Bestela, komisio bat irabaziko du apustu horrekiko.

- **ordaindu:** Egiaztatzen du apustu hori ordaindu behar zaion bezeroari edo ez. Ordaindu aldagaia true bada, ordainduko zaio.

- **pronostikoakKonparatu**: Pronostiko bakoitza konparatzen du apustuaren galderaren emaitzarekin. Berdinak badira, ordaindu true itzuliko luke.

- **apustuakOrdaindu**: Egindako apustu guztien begizta sortzen du, egiaztatzeko apustu horiek irabazi edo galtzen diren.

DELETE EVENT METODOA - Aritz Azkunaga

Hasierako kodea

```
public void deleteEvent(Event evi) {
	Event ev = db.find(Event.class, evi.getEventNumber());
	Bezero b;
	Bezero bez;
	Vector<Question> questions;
	Vector<Pronostikoa> pronostikoak;
	Vector<Apustua> apustuak;
	float dirua;
	questions = ev.getQuestions();
	for (Question qi : questions) {
		pronostikoak = qi.getPronostikoak();
		for (Pronostikoa pi : pronostikoak) {
			apustuak = pi.getApustuak();
			for (Apustua ai : apustuak) {
				dirua = ai.getApustuDirua();
				b = ai.getBezeroa();
				bez = db.find(Bezero.class, b.getErabiltzailea());
				bez.addDirua(dirua);
				bez.addMugimendua(dirua,
ResourceBundle.getBundle("Etiquetas").getString("EventDeleted") + ": "
							+ ev.getDescription(), false);
				bez.removeApustua(ai);
				db.getTransaction().begin();
				db.persist(bez);
				db.getTransaction().commit();
			}
		}
	}
	db.getTransaction().begin();
	db.remove(ev);
	db.getTransaction().commit();
}
```

Errefakturizatutako kodea

```
public void deleteEvent(Event evi) {
	Event ev = db.find(Event.class, evi.getEventNumber());
	Vector<Question> questions;
	Vector<Pronostikoa> pronostikoak;
	Vector<Apustua> apustuak;
	questions = ev.getQuestions();
	for (Question qi : questions) {
		pronostikoak = qi.getPronostikoak();
		for (Pronostikoa pi : pronostikoak) {
			apustuak = pi.getApustuak();
			for (Apustua ai : apustuak) {
				diruaItzuli(ev, ai);
			}
		}
	}
```

```
        db.getTransaction().begin();
        db.remove(ev);
        db.getTransaction().commit();
}

private void diruaItzuli(Event ev, Apustua ai) {
        Bezero b;
        Bezero bez;
        float dirua;
        dirua = ai.getApustuDirua();
        b = ai.getBezeroa();
        bez = db.find(Bezero.class, b.getErabiltzailea());
        bez.addDirua(dirua);
        bez.addMugimendua(dirua,
ResourceBundle.getBundle("Etiquetas").getString("EventDeleted") + ": "
                            + ev.getDescription(), false);
        bez.removeApustua(ai);
        db.getTransaction().begin();
        db.persist(bez);
        db.getTransaction().commit();
}
```

Deskribapena

Metodo luze bat zenez, bitan banatzea erabaki dut. Alde batetik lehen zegoen **deleteEvent** eta beste aldetik, **dirualtzuli**. **deleteEvent** metodoak bere funztioa beteko du, hau da, gertaera bat ezabatu eta lehen dirua itzultzeaz enkargatzen zen kodea beste metodo batera eraman da. Honela erroreen kudeaketa erraztuko da. Honez gain, hianbat aldagai lokalen ezabaketa behar izan da deleteEvent funtzioan, orain ez baitziren erabiltzen: *bezeroa b* eta bezeroa bez.

WRITE SIMPLE UNITS OF CODE

Lehenengo errefaktorizazioaren oso antzekoa da. Beraz, kode berdina eta errefaktorizazio berdina erabili dugu. Kasu honetan, ez dugu lerro kode kopuruan begiratu behar, konplexutasun ziklomatikoan baizik. Metodo bat egoki egoteko konplexutasun ziklomatikoa 4 edo gutxiago izan behar da.

Aurreko metodo berdina erabiliko dugu errefaktorizazio hau isladatzeko, bakarrik metodo hori dugulako 4 baino gehiago duen konplexutasun ziklomatikoa.

Deskribapena - Jon Jauregi

Errefaktorizazio honekin baita ere konplexutasun ziklomatikoa gutxitzea lortzen dugu. Hasieran, konplexutasun ziklomatikoa 6 zen, baina metodo hori 5 metodotan banatu dugunez, metodo bakoitza gehienez bikoa da. Beraz, horrekin lortzen dugu jatorrizko metodoa azkarrago testeatzea.

DUPLICATE CODE

Atal honetan errepikatutako kodea nola kendu behar dugun ikusiko dugu. Ez da batere eraginkorra kode errepikatua edukitzea, behin eta berriz kode berdina exekutatzen delako. Horretarako, aldagai lokaletaz baliatuko gara. Aldagai lokal bat sortu eta aldagai horren balioa errepikatzen ari den zatia izango da.

INITIALIZEDB METODOA - Jon Jauregi

Hasierako kodea

```
Question q7 = ev111.addQuestion("Zeinek irabaziko du partidua?", 1);
p3 = q7.addPronostikoa("Atletico", 2);
p4 = q7.addPronostikoa("Athletic", 4);
Pronostikoa p5 = q1.addPronostikoa("Atletico", (float) 1.2);
Pronostikoa p6 = q1.addPronostikoa("Athletic", 2);
Pronostikoa p7 = q2.addPronostikoa("Atletico", (float) 1.5);
Pronostikoa p8 = q2.addPronostikoa("Athletic", 2);
```

Errefakturizatutako kodea

```
Question q7 = ev111.addQuestion("Zeinek irabaziko du partidua?", 1);
String taldea1 = "Athletic";
String taldea2 = "Atletico";
p3 = q7.addPronostikoa(taldea2, 2);
p4 = q7.addPronostikoa(taldea1, 4);
Pronostikoa p5 = q1.addPronostikoa(taldea2, (float) 1.2);
Pronostikoa p6 = q1.addPronostikoa(taldea1, 2);
Pronostikoa p7 = q2.addPronostikoa(taldea2, (float) 1.5);
Pronostikoa p8 = q2.addPronostikoa(taldea1, 2);
```

Deskribapena

Zati honetan errepikatzen den atala pronostikoaren emaitzaren karaktere katea da. Beraz, karaktere kate horiek aldagai lokal batean gorde eta gero azaltzen diren ataletan atzitu ditzazkegu.

DATA ACCESS KLASEA - Aritz Azkunaga

Hasierako kodea

Kode hau jarri arren, klase guztian ematen da aldaketa.

```
b1.addMugimendua(30, ResourceBundle.getBundle("Etiquetas").getString("DiruaSartu"),
true);
b1.restDirua(5);
b1.addMugimendua(-5, ResourceBundle.getBundle("Etiquetas").getString(APUSTUA), false);
b1.addDirua(7);
b1.addMugimendua(7, ResourceBundle.getBundle("Etiquetas").getString("Win"), false);
b1.restDirua(3);
b1.addMugimendua(-3, ResourceBundle.getBundle("Etiquetas").getString(APUSTUA), false);
b1.addDirua(10);
b1.addMugimendua(10, ResourceBundle.getBundle("Etiquetas").getString("Win"), false);
Bezero b2 = new Bezero("Bezero2", "Bezero2", "W", "E", "111", UtilDate.newDate(2000, 1,
1), 2, "w@m");
Bezero b3 = new Bezero("Bezero3", "Bezero3", "W", "E", "111", UtilDate.newDate(2000, 1,
1), 2, "w@m");
b3.setKopiatu(true);
b3.addDirua(5);
b3.addMugimendua(5, ResourceBundle.getBundle("Etiquetas").getString("DiruaSartu"),
true);
b3.restDirua(3);
b3.addMugimendua(-3, ResourceBundle.getBundle("Etiquetas").getString(APUSTUA), false);
Bezero b4 = new Bezero("Bezero4", "Bezero4", "W", "E", "111", UtilDate.newDate(2000, 1,
1), 2, "w@m");
Bezero b5 = new Bezero("Bezero5", "Bezero5", "W", "E", "111", UtilDate.newDate(2000, 1,
1), 2, "w@m");
Bezero b6 = new Bezero("Bezero6", "Bezero6", "W", "E", "111", UtilDate.newDate(2000, 1,
1), 2, "w@m");
b6.setKopiatu(true);
b6.addDirua(30);
b6.addMugimendua(30, ResourceBundle.getBundle("Etiquetas").getString("DiruaSartu"),
true);
b6.restDirua(10);
b6.addMugimendua(-10, ResourceBundle.getBundle("Etiquetas").getString(APUSTUA), false);
b6.addDirua(20);
b6.addMugimendua(20, ResourceBundle.getBundle("Etiquetas").getString("Win"), false);
b6.restDirua(7);
b6.addMugimendua(-7, ResourceBundle.getBundle("Etiquetas").getString(APUSTUA), false);
```

Errefakturizatutako kodea

```
public class DataAccess {
private static final ResourceBundle etiketa = ResourceBundle.getBundle("Etiquetas");
```

```
b1.addMugimendua(30, etiketa.getString("DiruaSartu"), true);
b1.restDirua(5);
b1.addMugimendua(-5, etiketa.getString(APUSTUA), false);
b1.addDirua(7);
b1.addMugimendua(7, etiketa.getString("Win"), false);
```

```
b1.restDirua(3);
b1.addMugimendua(-3, etiketa.getString(APUSTUA), false);
b1.addDirua(10);
b1.addMugimendua(10, etiketa.getString("Win"), false);
Bezero b2 = new Bezero("Bezero2", "Bezero2", "W", "E", "111", UtilDate.newDate(2000, 1,
1), 2, "w@m");
Bezero b3 = new Bezero("Bezero3", "Bezero3", "W", "E", "111", UtilDate.newDate(2000, 1,
1), 2, "w@m");
b3.setKopiatu(true);
b3.addDirua(5);
b3.addMugimendua(5, etiketa.getString("DiruaSartu"), true);
b3.restDirua(3);
b3.addMugimendua(-3, etiketa.getString(APUSTUA), false);
Bezero b4 = new Bezero("Bezero4", "Bezero4", "W", "E", "111", UtilDate.newDate(2000, 1,
1), 2, "w@m");
Bezero b5 = new Bezero("Bezero5", "Bezero5", "W", "E", "111", UtilDate.newDate(2000, 1,
1), 2, "w@m");
Bezero b6 = new Bezero("Bezero6", "Bezero6", "W", "E", "111", UtilDate.newDate(2000, 1,
1), 2, "w@m");
b6.setKopiatu(true);
b6.addDirua(30);
b6.addMugimendua(30, etiketa.getString("DiruaSartu"), true);
b6.restDirua(10);
b6.addMugimendua(-10, etiketa.getString(APUSTUA), false);
b6.addDirua(20);
b6.addMugimendua(20, etiketa.getString("Win"), false);
b6.restDirua(7);
b6.addMugimendua(-7, etiketa.getString(APUSTUA), false);
```

Deskribapena

Kasu honetan, dataAccess klase guztian errepikatzen den kodigoa aldagai batean gordetzen da. Mantenimendua bermatzen da, *ResourceBundle.getBundle("Etiquetas")* ordez *etiketa* aldagaia erabilita.

KEEP UNIT INTERFACES SMALL

Honen helburua metodoak 4 parametro edo gutxiago izatea da. Parametroak objektu bihurtzea da helburua eta parametro gutxiago izateak kodea ulergarriagoa eta erabilgarria bihurtzen du.

ORDAINDU METODOA - Aritz Azkunaga

Hasierako kodea

```
private void apustuakOrdaindu(boolean ordaindu, Vector<Apustua> ap) {
	for (Apustua api : ap) {
		Bezero b = db.find(Bezero.class, api.getBezeroa().getErabiltzailea());
		Float kuota = api.getKuota();
		Float apustuDirua = api.getApustuDirua();
		Vector<Pronostikoa> pronostikoak = api.getPronostikoak();
		ordaindu = pronostikoakKonparatu(ordaindu, pronostikoak);
		ordaindu(ordaindu, api, b, kuota, apustuDirua);
	}
}
```

```
private void ordaindu(boolean ordaindu, Apustua api, Bezero b, Float kuota, Float
apustuDirua) {
	if (ordaindu) {
		b.addDirua(kuota * apustuDirua);
		b.addMugimendua(kuota * apustuDirua, etiketa.getString("Win"), false);
		db.getTransaction().begin();
		db.persist(b);
		db.getTransaction().commit();
		Bezero jabea = api.getJabea();
		jabeaDa(b, kuota, apustuDirua, jabea);
	}
}
```

Errefakturizatutako kodea

```
public class KuotaDirua {
	public Float kuota;
	public Float apustuDiru;

	public KuotaDirua(Float kuota, Float apustuDirua) {
		this.kuota=kuota;
		this.apustuDiru=apustuDiru;
	}

	public Float getKuota() {return kuota;}

	public void setKuota(Float kuota) {this.kuota = kuota;}
```

```
	public Float getApustuDiru() {return apustuDiru;}

	public void setApustuDiru(Float apustuDiru) {this.apustuDiru = apustuDiru;}

}
```

```
private void apustuakOrdaindu(boolean ordaindu, Vector<Apustua> ap) {
	for (Apustua api : ap) {
		Bezero b = db.find(Bezero.class, api.getBezeroa().getErabiltzailea());
		Float kuota = api.getKuota();
		Float apustuDirua = api.getApustuDirua();
		Vector<Pronostikoa> pronostikoak = api.getPronostikoak();
		ordaindu = pronostikoakKonparatu(ordaindu, pronostikoak);
		ordaindu(ordaindu, api, b, new KuotaDirua(kuota, apustuDirua));
	}
}

private void ordaindu(boolean ordaindu, Apustua api, Bezero b, KuotaDirua dirua) {
	if (ordaindu) {
		float kuota=dirua.getKuota();
		float apustuDirua= dirua.getApustuDiru();
		b.addDirua(kuota * apustuDirua);
		b.addMugimendua(kuota * apustuDirua, etiketa.getString("Win"), false);
		db.getTransaction().begin();
		db.persist(b);
		db.getTransaction().commit();
		Bezero jabea = api.getJabea();
		jabeaDa(b, kuota, apustuDirua, jabea);
	}
}
```

Deskribapena

Metodo hau aldatzea erabaki dut dataAccess-eko testetan eragina izango ez duelako. Aldaketak egin ondoren metodoa 5 parametro izatetik 4 parametro izatera pasa da, dirurako klase berri bat sortzen.

APUSTUAEGIN METODOA - Jon Jauregi

Hasierako kodea

```
public void apustuaEgin(Vector<Pronostikoa> pronostikoak, float dirua, Bezero b, Bezero
jabea, float kuota) {
		Pronostikoa pri;
		Bezero bez = db.find(Bezero.class, b.getErabiltzailea());
		Apustua a = bez.addApustua(dirua, pronostikoak, jabea, kuota);
		bez.restDirua(dirua);
		bez.addMugimendua(-(dirua), etiketa.getString(APUSTUA), false);
		for (Pronostikoa p : pronostikoak) {
			pri = db.find(Pronostikoa.class, p.getPronostikoaNumber());
			pri.addApustua(a);
			db.getTransaction().begin();
			db.persist(pri);
			db.getTransaction().commit();
		}
		db.getTransaction().begin();
		db.persist(bez);
		db.getTransaction().commit();
	}
```

Errefakturizatutako kodea

DataAccess.java

```
public void apustuaEgin(Vector<Pronostikoa> pronostikoak, Bezero b, Bezero jabea,
KuotaDirua kuota) {
	Pronostikoa pri;
	Bezero bez = db.find(Bezero.class, b.getErabiltzailea());
	Apustua a = bez.addApustua(kuota.getApustuDiru(), pronostikoak, jabea,
kuota.getKuota());
	bez.restDirua(kuota.getApustuDiru());
	bez.addMugimendua(-(kuota.getApustuDiru()), etiketa.getString(APUSTUA), false);
	for (Pronostikoa p : pronostikoak) {
		pri = db.find(Pronostikoa.class, p.getPronostikoaNumber());
		pri.addApustua(a);
		db.getTransaction().begin();
		db.persist(pri);
		db.getTransaction().commit();
	}
	db.getTransaction().begin();
	db.persist(bez);
	db.getTransaction().commit();
}
```

BLFacade.java

```
@WebMethod
public void apustuaEgin(Vector<Pronostikoa> pronostikoak, Bezero b, Bezero jabea,
KuotaDirua kuota);
```

BLFacadeImplementation.java

```
@WebMethod
public void apustuaEgin(Vector<Pronostikoa> pronostikoak, Bezero b, Bezero jabea,
KuotaDirua kuota) {
        dbManager.open(false);
        dbManager.apustuaEgin(pronostikoak, b, jabea, kuota);
        dbManager.close();
}
```

ApustuaConfirmGUI.java

```
facade.apustuaEgin(pronostikoak, b, null, new
KuotaDirua(Float.parseFloat(textAmount.getText()),kuota));

if(facade.getKopiatu(b)) {
        Vector<Bezero> kopiatzaileak = facade.getKopiatzaileak(b);
        for (Bezero a : kopiatzaileak) {
                float por = (float) a.getPortzentaia();
                if (!(a.getDirua() < Float.parseFloat(textAmount.getText())*por ||
Float.parseFloat(textAmount.getText())*por< minBet)) {

facade.apustuaEgin(pronostikoak, a, b, new
KuotaDirua(Float.parseFloat(textAmount.getText()) * por, kuota));
                }
        }

}
```

Deskribapena

Errefaktorizazio hau egiteko hainbat aldaketa egin behar izan ditugu. Lehenik eta behin, “DataAccess” klaseko “ApustuaEgin” metodoaren parametroak aldatu behar izan ditugu 5 parametro izan beharrean 4 izateko. Horretarako, “KuotaDirua” lehen sortutako klasea erabili dugu, kuota eta dirua klase horren barruan egoteko.

Hala ere, arazo batzuk sortu zaizkigu, beste klaseetan “apustuaEgin” metodoa erabiltzen delako. Beraz, klase horietan ere metodo horren deia egokitu behar izan dugu. Klase horiek “BLFacade”, “BLFacadeImplementation” eta “ApustuaConfirmGUI.java” dira.