Errefaktorizazioa eclipsen

Zdravko Todorov, Aitor Paredes eta Iñaki Moreno

Iñaki Moreno

1- Write short units of code

-LEHEN:

```
public Vector<Double> emaitzaIpini(Question question, Pronostikoa pronostikoa){
            Pronostikoa p =
                        db.find(Pronostikoa.class, pronostikoa.getIdentifikadorea());
            Question q = db.find(Question.class, question.getQuestionNumber());
            db.getTransaction().begin();
            q.setResult(pronostikoa.getDeskripzioa());
            Vector<Apustua> apustuak = p.getApustuak();
            Bezeroa bezeroa;
            double komisioa,apustuKop=0,irabazia=0;
            boolean irabazi;
            for(Apustua a : apustuak) {
                  apustuKop++;
                  irabazi=a.eguneratuAsmatutakoKop();
                  komisioa=0;
                  if(irabazi) {
                        if (a.getErrepikatua()!=null) {
                              Bezeroa bez = a.getErrepikatua();
                              bezeroa = a.getBezeroa();
                              Errepikapena errepikapen=bezeroa.getErrepikapena(bez);
                              System.out.println(a.getKopurua()+
" "+a.getKuotaTotala()+" "+a.getKopurua()+" "+errepikapen.getKomisioa());

komisioa=(a.getKopurua()*a.getKuotaTotala()-a.getKopurua())*errepikapen.getKomisioa()
;
                              System.out.println(komisioa);
                              bez.addMugimendua("Apustu errepikatuaren komisioa
("+bezeroa+")", komisioa,"irabazi");
                        }
                        bezeroa=a.getBezeroa();
                        irabazia=a.getKopurua()*a.getKuotaTotala()-komisioa;
                        bezeroa.addMugimendua("Apustua irabazi
```

```
("+a.getIdentifikadorea()+")", irabazia, "irabazi");
                }
            }
            Vector<Double> em = new Vector<Double>();
            em.add(apustuKop);
            em.add(irabazia);
            db.getTransaction().commit();
            return em;
      }
```

-ORAIN:

```
public Vector<Double> emaitzaIpini(Question question, Pronostikoa pronostikoa){
            Pronostikoa p =
                        db.find(Pronostikoa.class, pronostikoa.getIdentifikadorea());
            Question q = db.find(Question.class, question.getQuestionNumber());
            db.getTransaction().begin();
            q.setResult(pronostikoa.getDeskripzioa());
            Vector<Apustua> apustuak = p.getApustuak();
            Vector<Double> em = eguneratuApustuak(apustuak);
            db.getTransaction().commit();
            return em;
      }

      private Vector<Double> eguneratuApustuak(Vector<Apustua> apustuak) {
            Bezeroa bezeroa;
            double komisioa,apustuKop=0,irabazia=0;
            boolean irabazi;
            for(Apustua a : apustuak) {
                  apustuKop++;
                  irabazi=a.eguneratuAsmatutakoKop();
                  komisioa=0;
                  if(irabazi) {
                        irabazia = apustuaIrabazi(komisioa, a);
                  }
            }
            Vector<Double> em = new Vector<Double>();
            em.add(apustuKop);
            em.add(irabazia);
            return em;
      }
```

```
private double apustuaIrabazi(double komisioa, Apustua a) {
            Bezeroa bezeroa;
            double irabazia;
            if (a.getErrepikatua()!=null) {
                  Bezeroa bez = a.getErrepikatua();
                  bezeroa = a.getBezeroa();
                  Errepikapena errepikapen=bezeroa.getErrepikapena(bez);
                  System.out.println(a.getKopurua()+" "+a.getKuotaTotala()+"
"+a.getKopurua()+" "+errepikapen.getKomisioa());

komisioa=(a.getKopurua()*a.getKuotaTotala()-a.getKopurua())*errepikapen.getKomisioa()
;
                  System.out.println(komisioa);
                  bez.addMugimendua("Apustu errepikatuaren komisioa ("+bezeroa+")",
komisioa,"irabazi");
            }
            bezeroa=a.getBezeroa();
            irabazia=a.getKopurua()*a.getKuotaTotala()-komisioa;
            bezeroa.addMugimendua("Apustua irabazi ("+a.getIdentifikadorea()+")",
irabazia, "irabazi");
            return irabazia;
      }
```

-AZALPENA:

Lehen gauza asko egiten zituen metodo luze bat zen emaitzaIpini(), orain, aldiz, ulergarriagoak eta laburragoak diren hiru metodoetan banatua izan da. Horrela programa ulergarriagoa suertatuko da irakurri behar duen edozeinentzat, eta aldaketak egin behar badira errazagoa izango da hauek non egin behar diren lokalizatzea

2- Write simple units of code

- LEHEN:

```
      public void ezabatuGertaera(Event event) {
            Bezeroa bezeroa;
            int x;
            Event e = db.find(Event.class, event.getEventNumber());
            db.getTransaction().begin();
            for (Question question : e.getQuestions()) {
                  for (Pronostikoa pronostic : question.getPronostics()) {
                        for (Apustua bet : pronostic.getApustuak()) {
                              x = bet.removePronostikoa(pronostic);
                              if (x == 0) {
                                    itzuliMugimendua(bet);
                              } else if
(bet.getAsmatutakoKop().equals(bet.getPronostikoKop())) {
                                    irabaziMugimendua(bet);
                              }
                              System.out.println(bet.getPronostikoak() + "  berrize");
                        }
                  }
            }
            db.remove(e);
            db.getTransaction().commit();
            Apustua a = db.find(Apustua.class, 53);
            System.out.println(a);
      }
```

-ORAIN:

```
public void ezabatuGertaera(Event event) {
            Bezeroa bezeroa;
            int x;
            Event e = db.find(Event.class, event.getEventNumber());
            db.getTransaction().begin();
            for (Question question : e.getQuestions()) {
                  for (Pronostikoa pronostic : question.getPronostics()) {
                        for (Apustua bet : pronostic.getApustuak()) {
                              ezabatuPronostikoa(pronostic, bet);
                        }
                  }
            }
            db.remove(e);
            db.getTransaction().commit();
            Apustua a = db.find(Apustua.class, 53);
```

```
		System.out.println(a);
	}

	private void ezabatuPronostikoa(Pronostikoa pronostic, Apustua bet) {
		int x;
		x = bet.removePronostikoa(pronostic);
		if (x == 0) {
			itzuliMugimendua(bet);
		} else if (bet.getAsmatutakoKop().equals(bet.getPronostikoKop())) {
			irabaziMugimendua(bet);
		}
		System.out.println(bet.getPronostikoak() + "  berrize");
	}
```

-AZALPENA:
Lehen bost erabaki nodo zituen ezabatuGertaera() metodoa bi metodoetan banatua izan da, horrela, ezabatuGertaera() metodoak hiru erabaki nodo ditu, eta honek deitzen duen ezabatuPronostikoa() metodoak bi erabaki nodo.

3- Duplicate code

-LEHEN:

```
(...)

rate.setText("");
jakinarazpenak.setText("");
textArea.setText("");
textArea.setVisible(false);
nork.setText("");
mezua.setText("");
baldintzak.setText("");
euroko.setText("");
hilabetean.setText("");
komisioa.setText("");
onartuGaldera.setVisible(false);
bai.setVisible(false);
ez.setVisible(false);
getSelectedMezua();
bai.setVisible(false);
ez.setVisible(false);

(...)

rate.setText("");
jakinarazpenak.setText("");
textArea.setText("");
textArea.setVisible(false);
nork.setText("");
mezua.setText("");
baldintzak.setText("");
euroko.setText("");
hilabetean.setText("");
komisioa.setText("");
onartuGaldera.setVisible(false);
bai.setVisible(false);
ez.setVisible(false);
getSelectedMezua();
bai.setVisible(false);
ez.setVisible(false);

(...)
```

-ORAIN:

```
(...)

setRemoved();
(...)

setRemoved();

(...)

private void setRemoved() {
	rate.setText("");
	jakinarazpenak.setText("");
	textArea.setText("");
	textArea.setVisible(false);
	nork.setText("");
	mezua.setText("");
	baldintzak.setText("");
	euroko.setText("");
	hilabetean.setText("");
	komisioa.setText("");
	onartuGaldera.setVisible(false);
	bai.setVisible(false);
	ez.setVisible(false);
}
```

-AZALPENA:

PostontziaGUI-ko aurreko kodea errepikatua agertzen zen, eta hau ekiditeko metodo bat sortu dut, horrela metodoari dei egin eta kodea ez errepikatu behar izateko.

4- Long parameter list

-LEHEN:

DataAccess:

```
public Bezeroa bidaliMezua(Bezeroa nork, Bezeroa nori, String mezua, String gaia,
String mota, double zenbatApostatu, double hilabeteanZenbat, double
zenbatErrepikatuarentzat) {
            Bezeroa igorlea = db.find(Bezeroa.class, nork.getErabiltzaileIzena());
            Bezeroa hartzailea = db.find(Bezeroa.class, nori.getErabiltzaileIzena());
            db.getTransaction().begin();
            BezeroartekoMezua mezuBerria = igorlea.addBidalitakoBezeroMezua(nori,
mezua, gaia, mota, zenbatApostatu, hilabeteanZenbat, zenbatErrepikatuarentzat);
            hartzailea.addJasotakoBezeroMezua(mezuBerria);
            db.persist(mezuBerria);
            db.getTransaction().commit();
            return igorlea;
    }
```

BL:

```
public Bezeroa bidaliMezua(Bezeroa nork, Bezeroa nori, String mezua, String gaia,
String mota, double zenbatApostatu, double hilabeteanZenbat, double
zenbatErrepikatuarentzat) {
      dbManager.open(false);
      Bezeroa bezeroa = dbManager.bidaliMezua(nork, nori, mezua, gaia, mota,
zenbatApostatu, hilabeteanZenbat, zenbatErrepikatuarentzat);
      dbManager.close();
      return bezeroa;
    }
```

-ORAIN:

DataAccess:

```
public Bezeroa bidaliMezua(Bezeroa nork, Bezeroa nori, String mezua, List<Double>
zenbakiParametroak) {

            String[] mezuaParts = mezua.split(" ");

            Bezeroa igorlea = db.find(Bezeroa.class, nork.getErabiltzaileIzena());
            Bezeroa hartzailea = db.find(Bezeroa.class, nori.getErabiltzaileIzena());
```

```
            db.getTransaction().begin();
            BezeroartekoMezua mezuBerria = igorlea.addBidalitakoBezeroMezua(nori,
mezuaParts[0], mezuaParts[1], mezuaParts[2], zenbakiParametroak.get(0),
zenbakiParametroak.get(1), zenbakiParametroak.get(2));
            hartzailea.addJasotakoBezeroMezua(mezuBerria);
            db.persist(mezuBerria);
            db.getTransaction().commit();
            return igorlea;
    }
```

BL:

```
    public Bezeroa bidaliMezua(Bezeroa nork, Bezeroa nori, String mezua, String gaia,
String mota, double zenbatApostatu, double hilabeteanZenbat, double
zenbatErrepikatuarentzat) {
      dbManager.open(false);
      String mezuOsoa = mezua+" "+gaia+" "+mota;
      List<Double> zenbakiParametroak = new ArrayList<>();
      zenbakiParametroak.add(zenbatApostatu);
      zenbakiParametroak.add(hilabeteanZenbat);
      zenbakiParametroak.add(zenbatErrepikatuarentzat);
      Bezeroa bezeroa = dbManager.bidaliMezua(nork, nori, mezuOsoa, zenbakiParametroak);
      dbManager.close();
      return bezeroa;
    }
```

-AZALPENA:

Lehen zortzi parametro jasotzen zituen bidaliMezua() metodoaren parametro kopurua 4-ra jaitsi dut, alde batetik String-ak String bakar batean pasaz eta ondoren hauek hutsunearekin(“ ”) bereiztuz, eta double balioak ArrayList batean pasaz. DataAccess-ean aldaketa hau egiteak, argi dagoenez, BLImplementation klasean zenbait aldaketa ere eskatzen ditu, goiko kodean ikusi daitekeen moduan.

Aitor Paredes:

1- Write short units of code and keep unit interfaces small.

-LEHEN:

```
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 {
                        berria = new Bezeroa(izena, abizena1, abizena2,
                                 erabiltzaileIzena, pasahitza,
                                 telefonoZbkia, emaila,
                                 jaiotzeData);
                  }
                  db.getTransaction().begin();
                  db.persist(berria);
                  db.getTransaction().commit();
                  return berria;
            }
      }
```

-ORAIN:

```
public Pertsona register(String pertsonaDatuak,
                         String erabiltzailea,
                         String kontaktua,
                         Date jaiotzeData) throws UserAlreadyExist {
            String[] erabiltzaileaParts = erabiltzailea.split(" ");
            TypedQuery<Pertsona> query = db.createQuery("SELECT p FROM
            Pertsona p WHERE p.erabiltzaileIzena=?1", Pertsona.class);
            query.setParameter(1, erabiltzaileaParts[0]);
            List<Pertsona> pertsona = query.getResultList();
            if (!pertsona.isEmpty()) {
                  throw new UserAlreadyExist();
            } else {
                  Pertsona berria = pertsonaSortu(pertsonaDatuak,
                                    erabiltzailea, kontaktua, jaiotzeData);
                  db.getTransaction().begin();
                  db.persist(berria);
                  db.getTransaction().commit();
                  return berria;
            }
}

public Pertsona pertsonaSortu(String pertsonaDatuak,
                              String erabiltzailea,
                              String kontaktua,
                              Date jaiotzeData) {
            String[] erabiltzaileaParts = erabiltzailea.split(" ");
            if (erabiltzaileaParts[2].equals("admin")) {
                  return new Admin(pertsonaDatuak, erabiltzailea,
                                   kontaktua, jaiotzeData);
            } else if (erabiltzaileaParts[2].equals("langilea")) {
                  return new Langilea(pertsonaDatuak, erabiltzailea,
                                      kontaktua, jaiotzeData);
            } else {
                  return new Bezeroa(pertsonaDatuak, erabiltzailea,
                                     kontaktua, jaiotzeData);
            }
}
```

-AZALPENA:

Metodoaren parametro kopurua murrizteri dagokionez, hainabat gauza eduiki behar izan dira kontuan. Metodoa DA-ean dagoenez, bai GUIan bai BL-ean metodoa modifikatu behar izan da:

-GUIan lehen:

```
Pertsona pertsona = facade.register(izena, abizena1, abizena2,
                    erabiltzaileIzena, pasahitza, telefonoa, emaila,
                    UtilDate.newDate(Integer.valueOf(year.getText()),
                    hilabeteak.getIndexOf(hilabeteak.getSelectedItem()),
                    Integer.valueOf(day.getText())), "bezeroa");
```

-GUIan orain:

```
String pertsonaDatuak= name.getText() + " " +
                       abizena_1.getText() + " " +
                       abizena_2.getText();
String erabiltzailea = userName.getText() + " " +
                       new String(pas_1.getPassword()) +
                       " bezeroa";
String kontaktua = e_mail.getText() + " " +
                   phoneNumber.getText();
Date jaiotzeData = UtilDate.newDate(Integer.valueOf(year.getText()),
                   hilabeteak.getIndexOf(hilabeteak.getSelectedItem()),
                   Integer.valueOf(day.getText()));
BLFacade facade = MainGUI.getBusinessLogic();
try {
    Pertsona pertsona = facade.register(pertsonaDatuak, erabiltzailea,
                                        kontaktua, jaiotzeData);
```

Horrela GUIak BL-ko metodoari soilik 4 parametrorekin deitzen dio. Horretarako BLeko metodoa horrela gelditzen da (Intefazearen deklarazioa ere aldatu ondoren):

-BLean lehen:

```
public Pertsona register(String izena,
                         String abizena1,
                         String abizena2,
                         String erabiltzaileIzena,
                         String pasahitza,
                         String telefonoZbkia,
                         String email,
                         Date jaiotzeData,
                         String mota) throws UserAlreadyExist{
            dbManager.open(false);
            Pertsona emaitza = dbManager.register(izena, abizena1,
                                abizena2, erabiltzaileIzena, pasahitza,
                                telefonoZbkia, email, jaiotzeData, mota);
            dbManager.close();
            return emaitza;
}
```

-BLean orain:

```
public Pertsona register(String pertsonaDatuak,
                         String erabiltzailea,
                         String kontaktua,
                         Date jaiotzeData) throws UserAlreadyExist{
            dbManager.open(false);
            Pertsona emaitza = dbManager.register(pertsonaDatuak,
                                erabiltzailea, kontaktua, jaiotzeData);
            dbManager.close();
            return emaitza;
}
```

Bukatzeko parametroak murriztean azkeneko arazo bat topatzen dugu; erregistreratzerakoan Pertsona mota abstraktua heredatzen duen Admin, Langile edo Bezero klaseko instantzia bat sortzen da. Parametroak aldatu ditugunez, hauen eriakitzeileak ere aldatu beharko ditugu. Hiru klaseetan eraikitzeileak Pertsona klaseko eraikitzailea deitzen du atributu hauetarako beraz klase hauetan soilik super eraikitzaileri deia aldatu beharko da parametro berriekin egin dadin. Aldaketa handiena Pertsona klaseko eraikitzailean egin beharko da:

-Pertsona lehen:

```
public Pertsona (String izena, String abizena1, String abizena2, String
                 erabiltzaileIzena, String pasahitza, String telefonoZbkia,
                 String email, Date jaiotzeData) {

            this.izena = izena;
            this.abizena1 = abizena1;
            this.abizena2 = abizena2;
            this.erabiltzaileIzena = erabiltzaileIzena;
            this.pasahitza = pasahitza;
            this.telefonoZbkia = telefonoZbkia;
            this.email=email;
            this.jaiotzeData=jaiotzeData;
}
```

-Pertsona orain:

```
public Pertsona (String pertsonaDatuak, Date jaiotzeData, String
                 erabiltzailea,String kontaktua) {

            String[] izenaParts = pertsonaDatuak.split(" ");
            String[] erabiltzaileaParts = erabiltzailea.split(" ");
            String[] kontaktuaParts = kontaktua.split(" ");
            this.izena = izenaParts[0];
            this.abizena1 = izenaParts[1];
            this.abizena2 = izenaParts[2];
            this.erabiltzaileIzena = erabiltzaileaParts[0];
            this.pasahitza = erabiltzaileaParts[1];
            this.telefonoZbkia = kontaktuaParts[0];
            this.email=kontaktuaParts[1];
            this.jaiotzeData= jaiotzeData;
}
```

Parametroak murriztearen ondaorioz Pertsona klaseko eraikitzailea konplexuagoa da, baina atributuak murriz badaitezke ere, horrek ondoriak ekar litzake programako beste alde askotan, eta egitura osorik ez aldatzearren, atributuak mantentzea erabaki dut.

Beste aldetik metodoaren luzera murrizteko refaktorizazioa dugu. Parametroak murriztu ondoren, kodea txukun uzteko honen luzera murriztea erabaki dut. Horretarako erabiltzeile motaren arabera instantzia bat edo bestea sortzen denez, konprobaketa egiteko *sortuPertsona* metodoa sortu dut.

2- Write simple units of code

-LEHEN:

```
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>();
            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;
}
```

-ORAIN:

```
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>();
            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);
            errepApostuaEgin(pronostikoSorta, erabiltzaile, a, apus);
            erabiltzaile.addMugimendua("Apustua egin ", -a, "jokatu");
            db.getTransaction().commit();
            return erabiltzaile;
      }
```

-AZALPENA:

apostuEgin metodoaren konplexutasuna murrizteko hainbeste nested branching edukita bi gauza egin behar izan ditut. Lehendabizi kodea bitan banatu dut, honek bi gauza argi egiten baitzituen metodo berean. Hasierako zatia metdoan bertan utzi dut apostu normalaz arduratzen baita. Bigarren zatia *errepApostuaEgin*metodora mugitu dut, errepikapenetaz arduratzen baita. *errepApostuaEgin* metodoa *apostuaEgin* metodoak deitzen du.

```
public void errepApostuaEgin(ArrayList<Pronostikoa> pronostikoSorta,
               Bezeroa erabiltzaile, double a, Apustua apus) {
            Vector<Errepikapena> jarraitzaile =
               erabiltzaile.getErrepikatzaileak();
            for(Errepikapena er: jarraitzaile) {
                  double apustudiru=0;
                  if (er.getHilabeteHonetanGeratzenDena()>0) {
                        apustudiru =
                           Math.min(er.getHilabeteHonetanGeratzenDena(),
                           er.getApustatukoDena()*a);
                        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);
                        }
                  }
            }
}
```

Beste aldetik, hasierako metodoan bostgarren brenchingean konprobaketa bat egiten zen *er.getHilabeteHonetanGeratzenDena()* eta *er.getApostatukoDena()*a* artean, non balio minimoa *apustudiru*-ri esleitzen zitzaion. *if* hori ekiditeko *Math.min()* funtzioa erabili dut balio minimoa esleitzeko.

3- Duplicate code

-LEHEN:

```
(...)

}else {

      tableMezua.setVisible(false);
      scrollPaneMezuak.setVisible(false);
      send.setVisible(false);
      send.setEnabled(false);
      erantzun.setVisible(false);
      erantzun.setEnabled(false);
      mezuraItzuli.setVisible(false);
      mezuraItzuli.setEnabled(false);
      textArea.setVisible(false);
      gaia.setText("");
      aldatuTamaina(630,108);
      back.setBounds(10, 15, 89, 23);

(...)

if(elkarrizketaZerrenda.isEmpty()) {

      tableMezua.setVisible(false);
      scrollPaneMezuak.setVisible(false);
      send.setVisible(false);
      send.setEnabled(false);
      erantzun.setVisible(false);
      erantzun.setEnabled(false);
      mezuraItzuli.setVisible(false);
      mezuraItzuli.setEnabled(false);
      textArea.setVisible(false);
      gaia.setText("");
      aldatuTamaina(630,108);
      back.setBounds(10, 15, 89, 23);

(...)
```

-ORAIN:

```
(...)

}else {

	setPaneElkarrizketan();

(...)

if(elkarrizketaZerrenda.isEmpty()) {

	setPaneElkarrizketan();

(...)

public void setPaneElkarrizketan() {

	tableMezua.setVisible(false);
	scrollPaneMezuak.setVisible(false);
	send.setVisible(false);
	send.setEnabled(false);
	erantzun.setVisible(false);
	erantzun.setEnabled(false);
	mezuraItzuli.setVisible(false);
	mezuraItzuli.setEnabled(false);
	textArea.setVisible(false);
	gaia.setText("");
	aldatuTamaina(630,108);
	back.setBounds(10, 15, 89, 23);

}
```

-AZALPENA:

Kodean bi ekintza ezberdinetarako JFrame-aren elementuen atributuak horrela konfiguratzen ziren, kodea errepikatuz. Hori ekiditeko, kodea metodo batera mugitu eta erabili behar denean deia egitea erabaki dut.

Zdravko Todorov

1- Write short units of code

-LEHEN:

```
public void ezabatuGertaera(Event event) {
	Bezeroa bezeroa;
	int x;
	Event e = db.find(Event.class, event.getEventNumber());
	db.getTransaction().begin();
	for (Question question : e.getQuestions()) {
		for (Pronostikoa pronostic : question.getPronostics()) {
			for (Apustua bet : pronostic.getApustuak()) {
				x = bet.removePronostikoa(pronostic);
				if(x == 0) {
					bezeroa=bet.getBezeroa();
					if (bet.getErrepikatua()!= null) {
						Bezeroa erre = bet.getErrepikatua();
						Errepikapena er =
bezeroa.getErrepikapena(erre);

er.eguneratuHilHonetanGeratzenDena(bet.getKopurua());
					}
					bezeroa.addMugimendua("Apustuaren dirua
itzuli ("+bet.getIdentifikadorea()+")", bet.getKopurua(),"bueltatu");
					bezeroa.removeApustua(bet);
					db.remove(bet);
				}else
if(bet.getAsmatutakoKop().equals(bet.getPronostikoKop())) {
					double komisioa = 0;
					if (bet.getErrepikatua()!= null) {
						Bezeroa bez = bet.getErrepikatua();
						bezeroa = bet.getBezeroa();
						Errepikapena errepikapen =
bezeroa.getErrepikapena(bez);
						Komisioa =
(bet.getKopurua()*bet.getKuotaTotala()-bet.getKopurua())*errepikapen.getKom
isioa();
						bez.addMugimendua("Apustu
errepikatuaren komisioa ("+bezeroa+")", komisioa,"irabazi");
					}
					bezeroa = bet.getBezeroa();
```

```
                                double irabazia =
bet.getKopurua()*bet.getKuotaTotala()-komisioa;
                                bezeroa.addMugimendua("Apustua irabazi
("+bet.getIdentifikadorea()+")", irabazia, "irabazi");
                            }
                            System.out.println(bet.getPronostikoak()+"
berrize");
                    }
                }
            }
            db.remove(e);
            db.getTransaction().commit();
            Apustua a = db.find(Apustua.class, 53);
            System.out.println(a);
}
```

-ORAIN:

```
public void ezabatuGertaera(Event event) {
    Bezeroa bezeroa;
    int x;
    Event e = db.find(Event.class, event.getEventNumber());
    db.getTransaction().begin();
    for (Question question: e.getQuestions()) {
        for (Pronostikoa pronostic: question.getPronostics()) {
            for (Apustua bet: pronostic.getApustuak()) {
                x = bet.removePronostikoa(pronostic);
                if (x == 0) {
                    itzuliMugimendua(bet);
                } else if
(bet.getAsmatutakoKop().equals(bet.getPronostikoKop())) {
                    irabaziMugimendua(bet);
                }
                System.out.println(bet.getPronostikoak() + "  berrize");
            }
        }
    }
    db.remove(e);
    db.getTransaction().commit();
    Apustua a = db.find(Apustua.class, 53);
    System.out.println(a);
}
```

```
private void irabaziMugimendua(Apustua bet) {
    Bezeroa bezeroa;
    double komisioa = 0;
    if (bet.getErrepikatua() != null) {
        Bezeroa bez = bet.getErrepikatua();
        bezeroa = bet.getBezeroa();
        Errepikapena errepikapen = bezeroa.getErrepikapena(bez);
        komisioa = (bet.getKopurua() * bet.getKuotaTotala() -
bet.getKopurua()) * errepikapen.getKomisioa();
        bez.addMugimendua("Apustu errepikatuaren komisioa (" + bezeroa +
")", komisioa, "irabazi");
    }
    bezeroa = bet.getBezeroa();
    double irabazia = bet.getKopurua() * bet.getKuotaTotala() - komisioa;
    bezeroa.addMugimendua("Apustua irabazi (" + bet.getIdentifikadorea() +
")", irabazia, "irabazi");
}

private void itzuliMugimendua(Apustua bet) {
    Bezeroa bezeroa;
    bezeroa = bet.getBezeroa();
    if (bet.getErrepikatua() != null) {
        Bezeroa erre = bet.getErrepikatua();
        Errepikapena er = bezeroa.getErrepikapena(erre);
        er.eguneratuHilHonetanGeratzenDena(bet.getKopurua());
    }
    bezeroa.addMugimendua("Apustuaren dirua itzuli (" +
bet.getIdentifikadorea() + ")", bet.getKopurua(),
        "bueltatu");
    bezeroa.removeApustua(bet);
    db.remove(bet);
}
```

-AZALPENA:

Metodoa motzagoa bihurtze aldera, beste bi metodo atera dira bertatik. Batean, egin behar den mugimendua itzuli behar bada, metodo aparte batean egiten da. Eta beste kasuan, irabazteko bi kasuak, beste metodo batean. Horrela bi metodo nagusiko bi IF garrantzitsuen barnean bakarrik metodora deia geratuko da, askoz ere motzago.

2- Write simple units of code

-LEHEN:

```
public Bezeroa deleteApustua(Apustua apustua) throws EventFinished {
    db.getTransaction().begin();
    Apustua a = db.find(Apustua.class, apustua.getIdentifikadorea());
    ArrayList < Pronostikoa > pronostikoak = a.getPronostikoak();
    Date today = new Date();
    for (Pronostikoa p: pronostikoak) {
        Date eventDate = p.getQuestion().getEvent().getEventDate();
        if (!eventDate.after(today)) {
            throw new EventFinished();
        }
    }
    Bezeroa bezeroa = a.getBezeroa();
    bezeroa.removeApustua(a);
    if (a.getErrepikatua() != null) {
        Errepikapena errepikapen =
bezeroa.getErrepikapena(a.getErrepikatua());
        errepikapen.eguneratuHilHonetanGeratzenDena(a.getKopurua());
    }
    bezeroa.addMugimendua("Apustua ezabatu (" + a.getIdentifikadorea() +
")", a.getKopurua(), "bueltatu");
    for (Pronostikoa p: pronostikoak) {
        p.removeApustua(a);
    }
    Vector < Errepikapena > errepikatzaileak =
bezeroa.getErrepikatzaileak();
    for (Errepikapena er: errepikatzaileak) {
        Bezeroa bez = er.getNork();
        Apustua apusErr = bez.baduApustua(a);
        if (apusErr != null) {
            bez.removeApustua(apusErr);
            bez.addMugimendua("Apustu errepikatua ezabatu (" + bezeroa +
")", apusErr.getKopurua(), "bueltatu");
            for (Pronostikoa p: apusErr.getPronostikoak()) {
                p.removeApustua(apusErr);
            }
            er.eguneratuHilHonetanGeratzenDena(apusErr.getKopurua());
            db.remove(apusErr);
        }
    }
    db.remove(a);
    db.getTransaction().commit();
    return bezeroa;
}
```

-ORAIN:

```
public Bezeroa deleteApustua(Apustua apustua) throws EventFinished {
          db.getTransaction().begin();
          Apustua a = db.find(Apustua.class, apustua.getIdentifikadorea());
          ArrayList < Pronostikoa > pronostikoak = a.getPronostikoak();
          Date today = new Date();
          for (Pronostikoa p: pronostikoak) {
              Date eventDate = p.getQuestion().getEvent().getEventDate();
              if (!eventDate.after(today)) {
                  throw new EventFinished();
              }
          }
          Bezeroa bezeroa = a.getBezeroa();
          bezeroa.removeApustua(a);
          ezabatuApustua(a, pronostikoak, bezeroa);
          ezabatuErrepikatuak(a, bezeroa);
          db.remove(a);
          db.getTransaction().commit();
          return bezeroa;
      }



private void ezabatuErrepikatuak(Apustua a, Bezeroa bezeroa) {
          Vector < Errepikapena > errepikatzaileak =
bezeroa.getErrepikatzaileak();
          for (Errepikapena er: errepikatzaileak) {
              Bezeroa bez = er.getNork();
              Apustua apusErr = bez.baduApustua(a);
              if (apusErr != null) {
                  bez.removeApustua(apusErr);
                  bez.addMugimendua("Apustu errepikatua ezabatu (" +
bezeroa + ")", apusErr.getKopurua(), "bueltatu");
                  for (Pronostikoa p: apusErr.getPronostikoak()) {
                      p.removeApustua(apusErr);
                  }
                  er.eguneratuHilHonetanGeratzenDena(apusErr.getKopurua());
                  db.remove(apusErr);
              }
          }
      }
```

```
private void ezabatuApustua(Apustua a, ArrayList < Pronostikoa >
pronostikoak, Bezeroa bezeroa) {
          if (a.getErrepikatua() != null) {
              Errepikapena errepikapen =
bezeroa.getErrepikapena(a.getErrepikatua());
              errepikapen.eguneratuHilHonetanGeratzenDena(a.getKopurua());
          }
          bezeroa.addMugimendua("Apustua ezabatu (" +
a.getIdentifikadorea() + ")", a.getKopurua(), "bueltatu");
          for (Pronostikoa p: pronostikoak) {
              p.removeApustua(a);
          }
      }
```

-AZALPENA:

Metodoa simpleagoa bihurtzeko, bere komplexutasun ziklomatikoa jeitsi beharra dago. Kasu egokiena, metodo orok, 4 baino komplexutasun bajuagoa izatea da. Beraz, beste bi metodoetan banatua izan da nagusia, eta bere komplexutasuna guztiz aldatu da. Bi metodoetatik, lehenak apustua ezabatzen du eta besteak apustu errepikatuak bakarrik. Horrela, ez dago metodorik 4ko komplexutasun ziklomatikoa baino gehio duenik, programaren irakurketa eta errore aurkitzea nabarmenki erraztuz.

3- Duplicate code (PostontziaGUI.java)

-LEHEN:

```
(...) else if (mota.equals("errepikatuak eskaera onartu")) {

nork.setText(ResourceBundle.getBundle("Etiquetas").getString("Who")+":
"+selectedBezeroartekoMezua.getIgorlea());

mezua.setText(ResourceBundle.getBundle("Etiquetas").getString("Message")+"
"+selectedBezeroartekoMezua.getMezua());

baldintzak.setText(ResourceBundle.getBundle("Etiquetas").getString("Conditi
ons"));

euroko.setText(ResourceBundle.getBundle("Etiquetas").getString("EuroBet")+"
"+selectedBezeroartekoMezua.getZenbatApostatu()+"'");

hilabetean.setText("-"+ResourceBundle.getBundle("Etiquetas").getString("Max
Month")+" "+selectedBezeroartekoMezua.getHilabeteanZenbat()+"'");
(...)
```

```
(...) if(mota.equals("eskaera")) {

nork.setText(ResourceBundle.getBundle("Etiquetas").getString("Who")+":
"+selectedBezeroartekoMezua.getIgorlea());

mezua.setText(ResourceBundle.getBundle("Etiquetas").getString("Message")+"
"+selectedBezeroartekoMezua.getMezua());

baldintzak.setText(ResourceBundle.getBundle("Etiquetas").getString("Conditi
ons"));

euroko.setText(ResourceBundle.getBundle("Etiquetas").getString("EuroBet")+"
"+selectedBezeroartekoMezua.getZenbatApostatu()+"'");

hilabetean.setText("-"+ResourceBundle.getBundle("Etiquetas").getString("Max
Month")+" "+selectedBezeroartekoMezua.getHilabeteanZenbat()+"'");
(...)
```

-ORAIN:

```
(...) else if (mota.equals("errepikatuak eskaera onartu")) {
eskaeraOnartu();
(...)
```

```
(...) if(mota.equals("eskaera")) {
eskaeraOnartu();
(...)
```

-AZALPENA:

Kode errepikatua ekiditzeko, nahikoa da errepikatzen det zatia hartu, eta metodo bat sortzea horrekin. Ondorioz, kode hori exekutatzeko leku askotatik, nahikoa da metodoari deitzea leku guztietatik. Horrela, akats asko ekidin daitezke.

4- Long parameter list (Errepikapena.java)

-LEHEN:

```
public Errepikapena(Bezeroa nork, Bezeroa nori, double apustatukoDena,
double hilabetekoMax, double komisioa) {}
```

-ORAIN:

```
public Errepikapena(BezeroParea bezPare, double apustatukoDena, double
hilabetekoMax, double komisioa) {}
```

```
public BezeroParea(Bezeroa nork, Bezeroa nori) {
    super();
    this.nork = nork;
    this.nori = nori;
}

public Bezeroa getNork() {
    return nork;
}

public Bezeroa getNori() {
    return nori;
}
```

-AZALPENA:

Kasu honetan, hasiera batean genuen metodoaren parametro lista 5 elementuz osatuta zegoen eta honek desegokia egiten zuen gure printzipioetan oinarrituz. Hau kompontzeko, “nori” eta “nork” motako bezero desberdinak izan partez, “BezeroParea” motako objetua sortu da, non pare hori gordetzen den. Horrela, mota horretako objetua pasa dakioke “Errepikapena” eraikitzaileari, ondorioz, 4 parametro bakarrik uzten.