SOFTWARE INGENIERITZA II

Errefaktorizazioa

TALDEA:
-AMETS CARRERA
-INTZA LARBURU
-IORITZ ARAMENDI

AURKIBIDEA

A. IORITZ ARAMENDI

1. "Write short units of code"

Honen arabera, kodea ulerterrazo egiteko, testak hobetzeko zein akatsak gutxiagotzeko, komenigarria da metodoek ez izatea 15 lerro baino gehiago. Hala balitz, metodo hauek txikiagoetan banatu beharko lirateke. Adibidez, addBet metodoa bera oso luzea dugu:

```
public void addBet(Aukera hautatutakoAukera, Apustua apustu) {
                Erregistratua erreg = db.find(domain.Erregistratua.class, apustu.getLog());
                db.getTransaction().begin();
                if(apustu.getErantzunKop()<=1) {
                        Date data=new Date();
                        double zenbatekoa=apustu.getZenbatekoa();
                        String deskribapena="Apustua egin. Dirua: -";
                        Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
                        db.persist(m);
                        erreg.addMovement(m);
                        erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
                        erreg.addApustua(apustu);
                        for(String jarraitzailea: erreg.getJarraitzaileak()) {
                                Erregistratua jarraitzaileaErreg = db.find(domain.Erregistratua.class,
jarraitzailea);
                                if(jarraitzaileaErreg.getKontuDirua()>=zenbatekoa &&
jarraitzaileaErreg.isBanned()==false) {
                                        jarraitzaileaErreg.addMovement(m);

jarraitzaileaErreg.setKontuDirua(jarraitzaileaErreg.getKontuDirua() - apustu.getZenbatekoa());
                                        Apustua jarraitzaileApustua = new
Apustua(jarraitzailea,apustu.getZenbatekoa());
                                        jarraitzaileApustua.setErantzunak(apustu.getErantzunak());
                                        jarraitzaileApustua.setErantzunKop(1);
                                        jarraitzaileaErreg.addApustua(jarraitzaileApustua);
                                }else {
                                        String deskribapena2=("Ezin izan da Apustua kopiatu. ");
                                        Mugimendua m2=new Mugimendua(data,0,deskribapena2);
                                        jarraitzaileaErreg.addMovement(m2);
                                }
                        }
                        db.persist(apustu);
                }else { //apustu Anitza da
                        if(hautatutakoAukera.isAzkenaDa() == true) {
                                Date data=new Date();
                                double zenbatekoa=apustu.getZenbatekoa();
                                String deskribapena="Apustu Anitza egin. Dirua: -";
                                Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
                                db.persist(m);
                                erreg.addMovement(m);
                                erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
                                erreg.addApustua(apustu);
                                for(String jarraitzailea: erreg.getJarraitzaileak()) {
                                        Erregistratua jarraitzaileaErreg =
db.find(domain.Erregistratua.class, jarraitzailea);
                                        if(jarraitzaileaErreg.getKontuDirua()>=zenbatekoa &&
jarraitzaileaErreg.isBanned()==false) {
                                        jarraitzaileaErreg.addMovement(m);

jarraitzaileaErreg.setKontuDirua(jarraitzaileaErreg.getKontuDirua() - apustu.getZenbatekoa());
                                        Apustua jarraitzaileApustua = new
Apustua(jarraitzailea,apustu.getZenbatekoa());
                                        jarraitzaileApustua.setErantzunak(apustu.getErantzunak());
                                        jarraitzaileApustua.setErantzunKop(apustu.getErantzunak().size());
                                        jarraitzaileaErreg.addApustua(jarraitzaileApustua);
                                        }else {
                                                String deskribapena2=("Ezin izan da Apustua kopiatu. ");
                                                Mugimendua m2=new Mugimendua(data,0,deskribapena2);
                                                jarraitzaileaErreg.addMovement(m2);
                                        }
                                }
```

```
                                db.persist(apustu);
                        }
                }
                db.getTransaction().commit(); }
```

Hori horrela izanik, errefaktorizazioa ibiliko dugu eta metodo honetatik beste metodo bat aterako dugu, honen tamaina txikitze aldera.

Honela gelditukoo litzateke:

```
public void addBet(Aukera hautatutakoAukera, Apustua apustu) {
                Erregistratua erreg = db.find(domain.Erregistratua.class, apustu.getLog());
                db.getTransaction().begin();
                if(apustu.getErantzunKop()<=1) {
                        Date data=new Date();
                        double zenbatekoa=apustu.getZenbatekoa();
                        String deskribapena="Apustua egin. Dirua: -";
                        Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
                        db.persist(m);
                        erreg.addMovement(m);
                        erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
                        erreg.addApustua(apustu);
                        for(String jarraitzailea: erreg.getJarraitzaileak()) {
                                jarraitzaileenApustuBakarra(apustu, data, zenbatekoa, m, jarraitzailea);
                        }
                        db.persist(apustu);
                }else { //apustu Anitza da
                        if(hautatutakoAukera.isAzkenaDa() == true) {
                                Date data=new Date();
                                double zenbatekoa=apustu.getZenbatekoa();
                                String deskribapena="Apustu Anitza egin. Dirua: -";
                                Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
                                db.persist(m);
                                erreg.addMovement(m);
                                erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
                                erreg.addApustua(apustu);
                                for(String jarraitzailea: erreg.getJarraitzaileak()) {
                                        Erregistratua jarraitzaileaErreg =
db.find(domain.Erregistratua.class, jarraitzailea);
                                        if(jarraitzaileaErreg.getKontuDirua()>=zenbatekoa &&
jarraitzaileaErreg.isBanned()==false) {
                                        jarraitzaileaErreg.addMovement(m);

jarraitzaileaErreg.setKontuDirua(jarraitzaileaErreg.getKontuDirua() - apustu.getZenbatekoa());
                                        Apustua jarraitzaileApustua = new
Apustua(jarraitzailea,apustu.getZenbatekoa());
                                        jarraitzaileApustua.setErantzunak(apustu.getErantzunak());
                                        jarraitzaileApustua.setErantzunKop(apustu.getErantzunak().size());
                                        jarraitzaileaErreg.addApustua(jarraitzaileApustua);
                                        }else {
                                                String deskribapena2=("Ezin izan da Apustua kopiatu. ");
                                                Mugimendua m2=new Mugimendua(data,0,deskribapena2);
                                                jarraitzaileaErreg.addMovement(m2);
                                        }
                                }
                                db.persist(apustu);
                        }
                }
                db.getTransaction().commit();
        }
        private void jarraitzaileenApustuBakarra(Apustua apustu, Date data, double zenbatekoa, Mugimendua
m,String jarraitzailea) {
                Erregistratua jarraitzaileaErreg = db.find(domain.Erregistratua.class, jarraitzailea);
                if(jarraitzaileaErreg.getKontuDirua()>=zenbatekoa && jarraitzaileaErreg.isBanned()==false) {
                        jarraitzaileaErreg.addMovement(m);
                        jarraitzaileaErreg.setKontuDirua(jarraitzaileaErreg.getKontuDirua() -
apustu.getZenbatekoa());
                        Apustua jarraitzaileApustua = new Apustua(jarraitzailea,apustu.getZenbatekoa());
                        jarraitzaileApustua.setErantzunak(apustu.getErantzunak());
                        jarraitzaileApustua.setErantzunKop(1);
                        jarraitzaileaErreg.addApustua(jarraitzaileApustua);
                }else {
                        String deskribapena2=("Ezin izan da Apustua kopiatu. ");
                        Mugimendua m2=new Mugimendua(data,0,deskribapena2);
                        jarraitzaileaErreg.addMovement(m2);
```

```
                }
        }
```

Ikus dezakegu, addBet metodoaren luzera txikiagotu dugula errefaktorizazioa erabili eta beste metodo bat sartuaz.

2."Write simple units of code"

Arau honen arabera, bifurkazioen kopurua 4 baino txikiagoa izatea komeni da. Hain zuzen, kodea nondik joan erabakiko duen bifurkazio kopurua 4 baino txikiagoa izan behar da. Hau da, if-for-while… kopurua 4 baino txikiagoa izatea komenigarria da. Beste era batera esanda, konplexutasun ziklomatikoa 5 baino txikiagoa. Horretarako addBet metodoa bera hartuko dugu.

```
public void addBet(Aukera hautatutakoAukera, Apustua apustu) {
                Erregistratua erreg = db.find(domain.Erregistratua.class, apustu.getLog());
                db.getTransaction().begin();
                if(apustu.getErantzunKop()<=1) {
                        Date data=new Date();
                        double zenbatekoa=apustu.getZenbatekoa();
                        String deskribapena="Apustua egin. Dirua: -";
                        Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
                        db.persist(m);
                        erreg.addMovement(m);
                        erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
                        erreg.addApustua(apustu);
                        for(String jarraitzailea: erreg.getJarraitzaileak()) {
                                jarraitzaileenApustuBakarra(apustu, data, zenbatekoa, m, jarraitzailea);
                        }
                        db.persist(apustu);
                }else { //apustu Anitza da
                        if(hautatutakoAukera.isAzkenaDa() == true) {
                                Date data=new Date();
                                double zenbatekoa=apustu.getZenbatekoa();
                                String deskribapena="Apustu Anitza egin. Dirua: -";
                                Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
                                db.persist(m);
                                erreg.addMovement(m);
                                erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
                                erreg.addApustua(apustu);
                                for(String jarraitzailea: erreg.getJarraitzaileak()) {
                                        Erregistratua jarraitzaileaErreg =
db.find(domain.Erregistratua.class, jarraitzailea);
                                        if(jarraitzaileaErreg.getKontuDirua()>=zenbatekoa &&
jarraitzaileaErreg.isBanned()==false) {
                                        jarraitzaileaErreg.addMovement(m);

jarraitzaileaErreg.setKontuDirua(jarraitzaileaErreg.getKontuDirua() - apustu.getZenbatekoa());
                                        Apustua jarraitzaileApustua = new
Apustua(jarraitzailea,apustu.getZenbatekoa());
                                        jarraitzaileApustua.setErantzunak(apustu.getErantzunak());
                                        jarraitzaileApustua.setErantzunKop(apustu.getErantzunak().size());
                                        jarraitzaileaErreg.addApustua(jarraitzaileApustua);
                                        }else {
                                                String deskribapena2=("Ezin izan da Apustua kopiatu. ");
                                                Mugimendua m2=new Mugimendua(data,0,deskribapena2);
                                                jarraitzaileaErreg.addMovement(m2);
                                        }
                                }
                                db.persist(apustu);
                        }
                }
                db.getTransaction().commit();
        }
```

Ikus dezekegunez, orain kode bifurkazio kopurua 5 ekoa da. Horretarako, beste metodo bat aterako dugu eta bifurkazio kopurua txikitu:

```
public void addBet(Aukera hautatutakoAukera, Apustua apustu) {
                Erregistratua erreg = db.find(domain.Erregistratua.class, apustu.getLog());
                db.getTransaction().begin();
                if(apustu.getErantzunKop()<=1) {
                        Date data=new Date();
                        double zenbatekoa=apustu.getZenbatekoa();
                        String deskribapena="Apustua egin. Dirua: -";
                        Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
                        db.persist(m);
                        erreg.addMovement(m);
                        erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
                        erreg.addApustua(apustu);
                        for(String jarraitzailea: erreg.getJarraitzaileak()) {
                                jarraitzaileenApustuBakarra(apustu, data, zenbatekoa, m, jarraitzailea);
                        }
                        db.persist(apustu);
                }else { //apustu Anitza da
                        if(hautatutakoAukera.isAzkenaDa() == true) {
                                Date data=new Date();
                                double zenbatekoa=apustu.getZenbatekoa();
                                String deskribapena="Apustu Anitza egin. Dirua: -";
                                Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
                                db.persist(m);
                                erreg.addMovement(m);
                                erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
                                erreg.addApustua(apustu);
                                for(String jarraitzailea: erreg.getJarraitzaileak()) {
                                        jarraitzaileApustuAnitz(apustu, data, zenbatekoa, m,
jarraitzailea);
                                }
                                db.persist(apustu);
                        }
                }
                db.getTransaction().commit();
        }
        private void jarraitzaileApustuAnitz(Apustua apustu, Date data, double zenbatekoa, Mugimendua m,
                        String jarraitzailea) {
                Erregistratua jarraitzaileaErreg = db.find(domain.Erregistratua.class, jarraitzailea);
                if(jarraitzaileaErreg.getKontuDirua()>=zenbatekoa && jarraitzaileaErreg.isBanned()==false) {
                jarraitzaileaErreg.addMovement(m);
                jarraitzaileaErreg.setKontuDirua(jarraitzaileaErreg.getKontuDirua() -
apustu.getZenbatekoa());
                Apustua jarraitzaileApustua = new Apustua(jarraitzailea,apustu.getZenbatekoa());
                jarraitzaileApustua.setErantzunak(apustu.getErantzunak());
                jarraitzaileApustua.setErantzunKop(apustu.getErantzunak().size());
                jarraitzaileaErreg.addApustua(jarraitzaileApustua);
                }else {
                        String deskribapena2=("Ezin izan da Apustua kopiatu. ");
                        Mugimendua m2=new Mugimendua(data,0,deskribapena2);
                        jarraitzaileaErreg.addMovement(m2);
                }
        }
```

Ikus dezakegu, jarraitzaileApustuAnitz metodoa aterata, 4-ko bifurkazio kopurua gelditzen zaiola addBet metodoari.

3. Duplicate code

Kode errepikatuaren zatia, sonarrekin konpondu dugunaren berdina da; hau da, kodea hainbatetan errepikatu beharrean, aldagai batean gordetzea egokiagoa dela. Horretarako, kode hau aldatu dugu:

```
String irabazlea = "¿Quién ganará el partido?";
                    String winner = "Who will win the match?";
                    if (Locale.getDefault().equals(new Locale("es"))) {
                          q1=ev1.addQuestion(irabazlea,1);
                          q2=ev1.addQuestion("¿Quién meterá el primer
gol?",2);
                          q3=ev11.addQuestion(irabazlea,1);
                          q4=ev11.addQuestion("¿Cuántos goles se
marcarán?",2);
                          q5=ev17.addQuestion(irabazlea,1);
                          q6=ev17.addQuestion("¿Habrá goles en la primera
parte?",2);
                    }
                    else if (Locale.getDefault().equals(new Locale("en"))) {
                          q1=ev1.addQuestion(winner,1);
                          q2=ev1.addQuestion(winner,2);
                          q3=ev11.addQuestion(winner,1);
                          q4=ev11.addQuestion("How many goals will be scored
in the match?",2);
                          q5=ev17.addQuestion("Who will win the match?",1);
                          q6=ev17.addQuestion("Will there be goals in the
first half?",2);
                    }
                    else {
                          q1=ev1.addQuestion("Zeinek irabaziko du
partidua?",1);
                          q2=ev1.addQuestion("Zeinek sartuko du lehenengo
gola?",2);
                          q3=ev11.addQuestion("Zeinek irabaziko du
partidua?",1);
                          q4=ev11.addQuestion("Zenbat gol sartuko dira?",2);
                          q5=ev17.addQuestion("Zeinek irabaziko du
partidua?",1);
                          q6=ev17.addQuestion("Golak sartuko dira lehenengo
zatian?",2);

                    }
```

Zati horretan, “Zeinek irabaziko du partidua?” galdera aldatuko dugu eta aldagai batean gorde:

```
String irabazlea = "¿Quién ganará el partido?";
                    String winner = "Who will win the match?";
                    String irabazlea2 = "Zeinek irabaziko du partidua?";
                    if (Locale.getDefault().equals(new Locale("es"))) {
                          q1=ev1.addQuestion(irabazlea,1);
                          q2=ev1.addQuestion("¿Quién meterá el primer
gol?",2);
                          q3=ev11.addQuestion(irabazlea,1);
                          q4=ev11.addQuestion("¿Cuántos goles se
marcarán?",2);
                          q5=ev17.addQuestion(irabazlea,1);
                          q6=ev17.addQuestion("¿Habrá goles en la primera
parte?",2);
                    }
                    else if (Locale.getDefault().equals(new Locale("en"))) {
                          q1=ev1.addQuestion(winner,1);
                          q2=ev1.addQuestion(winner,2);
                          q3=ev11.addQuestion(winner,1);
                          q4=ev11.addQuestion("How many goals will be scored
in the match?",2);
                          q5=ev17.addQuestion("Who will win the match?",1);
                          q6=ev17.addQuestion("Will there be goals in the
first half?",2);
                    }
                    else {
                          q1=ev1.addQuestion(irabazlea2,1);
                          q2=ev1.addQuestion("Zeinek sartuko du lehenengo
gola?",2);
                          q3=ev11.addQuestion(irabazlea2,1);
                          q4=ev11.addQuestion("Zenbat gol sartuko dira?",2);
                          q5=ev17.addQuestion("Zeinek irabaziko du
partidua?",1);
                          q6=ev17.addQuestion("Golak sartuko dira lehenengo
zatian?",2);

                    }
```

Modu honetan irabazlea2-n gordeko dugu galdera eta hau berrerabiliko dugu.

4. Keep unit interfaces small

Azken arau honen arabera, metodo baten parametro kopurua 4 baino txikiagoa izan behar du. Hori beteko ez balitz, objetu berri bat sortu beharko genuke.

```
private void jarraitzaileenApustuBakarra(Apustua apustu, Date data, double zenbatekoa, Mugimendua m,String
jarraitzailea) {
                Erregistratua jarraitzaileaErreg = db.find(domain.Erregistratua.class, jarraitzailea);
                if(jarraitzaileaErreg.getKontuDirua()>=zenbatekoa && jarraitzaileaErreg.isBanned()==false) {
                        jarraitzaileaErreg.addMovement(m);
                        jarraitzaileaErreg.setKontuDirua(jarraitzaileaErreg.getKontuDirua() -
apustu.getZenbatekoa());
                        Apustua jarraitzaileApustua = new Apustua(jarraitzailea,apustu.getZenbatekoa());
                        jarraitzaileApustua.setErantzunak(apustu.getErantzunak());
                        jarraitzaileApustua.setErantzunKop(1);
                        jarraitzaileaErreg.addApustua(jarraitzaileApustua);
                }else {
                        String deskribapena2=("Ezin izan da Apustua kopiatu. ");
                        Mugimendua m2=new Mugimendua(data,0,deskribapena2);
                        jarraitzaileaErreg.addMovement(m2);
                }
        }
```

Metodo honetan, 5 parametro jasotzen ditu. Beraz, apustuInfo sortu dugu:

```
private void jarraitzaileenApustuBakarra(ApustuaInfo a) {
                Erregistratua jarraitzaileaErreg = db.find(domain.Erregistratua.class, a.getJarraitzailea());
                if(jarraitzaileaErreg.getKontuDirua()>=a.getZenbatekoa() &&
jarraitzaileaErreg.isBanned()==false) {
                        jarraitzaileaErreg.addMovement(a.getMove());
                        jarraitzaileaErreg.setKontuDirua(jarraitzaileaErreg.getKontuDirua() -
a.getApustua().getZenbatekoa());
                        Apustua jarraitzaileApustua = new Apustua((String)
a.getJarraitzailea(),(double)a.getApustua().getZenbatekoa());
                        jarraitzaileApustua.setErantzunak(a.getApustua().getErantzunak());
                        jarraitzaileApustua.setErantzunKop(1);
                        jarraitzaileaErreg.addApustua(jarraitzaileApustua);
                }else {
                        String deskribapena2=("Ezin izan da Apustua kopiatu. ");
                        Mugimendua m2=new Mugimendua(a.getData(),0,deskribapena2);
                        jarraitzaileaErreg.addMovement(m2);
                }
        }
```

Modu honetan, paramtero bakarra jasoko luke. Horrez gain, addBet metodoko deia ere aldatu beharko genuke:

```
ApustuaInfo a = new ApustuaInfo(apustu, data, zenbatekoa, m, jarraitzailea);
                                jarraitzaileenApustuBakarra(a);
```

Modu honetara, arau hau ere beteko luke metodoak.

B. Intza Larburu

5. "Write short units of code"

Honen arabera, kodea ulerterrazo egiteko, testak hobetzeko zein akatsak gutxitzeko, komenigarria da metodoek ez izatea 15 lerro baino gehiago. Hala balitz, metodo hauek metodo txikiagotan banatu beharko liratezke . Adibidez, emaitzaGehitu metodoa oso luzea dugu:

```
public void emaitzaGehitu(Question q, String emaitza) {
                db.getTransaction().begin();
                boolean denakAsmatuak=true;
                float biderkatzaileTotala=0;
                System.out.println(">> DataAccess: getAllBets");
                TypedQuery<Apustua> query = db.createQuery("SELECT ap FROM Apustua ap",Apustua.class);
                List<Apustua> apustuLista = query.getResultList();
                for(Apustua ap:apustuLista) {
                        if(ap.isBukatua()==false) {
                                for(Aukera auk:ap.getErantzunak()) {
                                        for(Aukera auke:q.getAukerak()) {
                                                if(auke.getAukeraID()==auk.getAukeraID() &&  auk.getAukeraIzena().equals(emaitza)) {
                                                        auk.setEgoera("win");
                                                    if(ap.getErantzunKop()==1) {
                                                          ap.setAsmatua(true);
                                                    }
                                                }
                                            }
                                        }
                                }
                        }
                }
                System.out.println(">> DataAccess: getAllErreg");
                TypedQuery<Erregistratua> query2 = db.createQuery("SELECT erreg FROM Erregistratua erreg",Erregistratua.class);
                List<Erregistratua> ErregistratuLista = query2.getResultList();
                for(Erregistratua erreg:ErregistratuLista) {
                        for(Apustua apos:erreg.getApustuak()) {
                                if(apos.getErantzunKop()==1 && apos.isAsmatua()==true && apos.isBukatua()==false) {
                                        erreg.setKontuDirua(erreg.getKontuDirua()+ (apos.getZenbatekoa()*apos.getErantzunak().get(0).getKuota()));
                                        Date data=new Date();
                                        String deskribapena="Apustua irabazi. Dirua= +";
                                        double zenbatekoa=apos.getZenbatekoa()*apos.getErantzunak().get(0).getKuota();
                                        Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
                                        db.persist(m);
                                        erreg.addMovement(m);
                                        apos.setBukatua(true);
```

```
                                }else if(apos.getErantzunKop()>1 && apos.isBukatua()==false) { //apustu Anitza da
                                        for(Aukera aukera:apos.getErantzunak()) {
                                                if(aukera.getEgoera().equals("win")==false) {
                                                        denakAsmatuak=false;
                                                }
                                                biderkatzaileTotala = biderkatzaileTotala + aukera.getKuota();
                                         }
                                        if(denakAsmatuak==true) {
                                                apos.setAsmatua(true);
                                        }if(apos.isAsmatua()==true && apos.isBukatua()==false) {
                                                erreg.setKontuDirua(erreg.getKontuDirua() + (apos.getZenbatekoa()* biderkatzaileTotala));
                                                Date data=new Date();
                                                double zenbatekoa=apos.getZenbatekoa()* biderkatzaileTotala;
                                                String deskribapena="apustu Anitza irabazi. Dirua=+";
                                                Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
                                                db.persist(m);
                                                erreg.addMovement(m);
                                                apos.setBukatua(true);
                                        } //if
                                }//else if
                        }//for
                }//for
                db.getTransaction().commit();
        }
```

Hori horrela izanik, errefaktorizazioa ibiliko dugu eta metodo honetatik beste metodo bat aterako dugu, honen tamaina txikitze aldera.

Honela geldituko litzateke:

```
public void emaitzaGehitu(Question q, String emaitza) {
		db.getTransaction().begin();
		boolean denakAsmatuak=true;
		float biderkatzaileTotala=0;
		System.out.println(">> DataAccess: getAllBets");
		TypedQuery<Apustua> query = db.createQuery("SELECT ap FROM Apustua ap",Apustua.class);
		List<Apustua> apustuLista = query.getResultList();
		for(Apustua ap:apustuLista) {
			if(ap.isBukatua()==false) {
				for(Aukera auk:ap.getErantzunak()) {
					for(Aukera auke:q.getAukerak()) {
						if(auke.getAukeraID()==auk.getAukeraID() &&  auk.getAukeraIzena().equals(emaitza)) {
							auk.setEgoera("win");
						    if(ap.getErantzunKop()==1) {
						    	ap.setAsmatua(true);
						    }
					    }
				    }
			    }
		    }
		}
		System.out.println(">> DataAccess: getAllErreg");
		TypedQuery<Erregistratua> query2 = db.createQuery("SELECT erreg FROM Erregistratua erreg",Erregistratua.class);
		List<Erregistratua> ErregistratuLista = query2.getResultList();
		for(Erregistratua erreg:ErregistratuLista) {
			for(Apustua apos:erreg.getApustuak()) {
				if(apos.getErantzunKop()==1 && apos.isAsmatua()==true && apos.isBukatua()==false) {
					erreg.setKontuDirua(erreg.getKontuDirua()+ (apos.getZenbatekoa()*apos.getErantzunak().get(0).getKuota()));
					Date data=new Date();
					String deskribapena="Apustua irabazi. Dirua= +";
					double zenbatekoa=apos.getZenbatekoa()*apos.getErantzunak().get(0).getKuota();
					Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
					db.persist(m);
					erreg.addMovement(m);
					apos.setBukatua(true);
				}else if(apos.getErantzunKop()>1 && apos.isBukatua()==false) { //apustu Anitza da
					this.emaitzaGehituApustuAnitza(apos, erreg);
				}//else if
			}//for
		}//for
		db.getTransaction().commit();
```

```
	public void emaitzaGehituApustuAnitza(Apustua apos, Erregistratua erreg) {
		boolean denakAsmatuak=true;
		float biderkatzaileTotala=0;
		for(Aukera aukera:apos.getErantzunak()) {
			if(aukera.getEgoera().equals("win")==false) {
				denakAsmatuak=false;
			}
			biderkatzaileTotala = biderkatzaileTotala + aukera.getKuota();
		 }
		if(denakAsmatuak==true) {
			apos.setAsmatua(true);
		}if(apos.isAsmatua()==true && apos.isBukatua()==false) {
			erreg.setKontuDirua(erreg.getKontuDirua() + (apos.getZenbatekoa()* biderkatzaileTotala));
			Date data=new Date();
			double zenbatekoa=apos.getZenbatekoa()* biderkatzaileTotala;
			String deskribapena="apustu Anitza irabazi. Dirua=+";
			Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
			db.persist(m);
			erreg.addMovement(m);
			apos.setBukatua(true);
		} //if
	}
```

Ikus dezakegu, emaitzaGehitu metodoaren luzera txikitu dugula errefaktorizazioa erabili eta beste metodo bat sortuz.

6."Write simple units of code"

Arau honen arabera, bifurkazioen kopurua 4 baino txikiagoa izatea komeni da. Hain zuzen, kodea nondik joan erabakiko duen bifurkazio kopurua 4 baino txikiagoa izan behar da, if-for-while… kopurua. Beste era batera esanda, konplexutasun ziklomatikoa 5 baino txikiagoa. Horretarako emaitzaGehitu metodoa bera hartuko dugu.

Bifurkazio kopurua txikitzeko, beste metodo bat aterako dugu:

```
public void emaitzaGehitu(Question q, String emaitza) {
		db.getTransaction().begin();
		System.out.println(">> DataAccess: getAllBets");
		TypedQuery<Apustua> query = db.createQuery("SELECT ap FROM Apustua ap",Apustua.class);
		List<Apustua> apustuLista = query.getResultList();
		this.apustuakAztertu(apustuLista, q, emaitza);
		System.out.println(">> DataAccess: getAllErreg");
		TypedQuery<Erregistratua> query2 = db.createQuery("SELECT erreg FROM Erregistratua erreg",Erregistratua.class);
		List<Erregistratua> ErregistratuLista = query2.getResultList();
		for(Erregistratua erreg:ErregistratuLista) {
			for(Apustua apos:erreg.getApustuak()) {
				if(apos.getErantzunKop()==1 && apos.isAsmatua()==true && apos.isBukatua()==false) {
					erreg.setKontuDirua(erreg.getKontuDirua()+ (apos.getZenbatekoa()*apos.getErantzunak().get(0).getKuota()));
					Date data=new Date();
					String deskribapena="Apustua irabazi. Dirua= +";
					double zenbatekoa=apos.getZenbatekoa()*apos.getErantzunak().get(0).getKuota();
					Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
					db.persist(m);
					erreg.addMovement(m);
					apos.setBukatua(true);
				}else if(apos.getErantzunKop()>1 && apos.isBukatua()==false) { //apustu Anitza da
					this.emaitzaGehituApustuAnitza(apos, erreg);
				}//else if
			}//for
		}//for
		db.getTransaction().commit();
	}
```

```
	public void apustuakAztertu(List<Apustua> apustuLista, Question q, String emaitza) {
		for(Apustua ap:apustuLista) {
			if(ap.isBukatua()==false) {
				for(Aukera auk:ap.getErantzunak()) {
					for(Aukera auke:q.getAukerak()) {
						if(auke.getAukeraID()==auk.getAukeraID() &&  auk.getAukeraIzena().equals(emaitza)) {
							auk.setEgoera("win");
						    if(ap.getErantzunKop()==1) {
						    	ap.setAsmatua(true);
						    }
					    }
				    }
			    }
		    }
		}
	}
```

Ikus dezakegu, apustuakAztertu metodoa aterata, txikitu egin dugula konplexutasun ziklomatikoa.

7. Duplicate code

Kode errepikatuaren zatia, sonarrekin konpondu dugunaren berdina da; hau da, kodea hainbatetan errepikatu beharrean, aldagai batean gordetzea egokiagoa dela. Horretarako, kode hau aldatu dugu:

```
private JRadioButton getRdbtnEuskera() {
    if (rdbtnEuskera == null) {
        rdbtnEuskera = new JRadioButton(ResourceBundle.getBundle("Etiquetas").getString("ConsultMovements
        rdbtnEuskera.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Locale.setDefault(new Locale("eus"));
                System.out.println("Locale: "+Locale.getDefault());
                redibujar();                }
        });
        buttonGroup.add(rdbtnEuskera);
    }
    return rdbtnEuskera;
}
private JRadioButton getRdbtnCastellano() {
    if (rdbtnCastellano == null) {
        rdbtnCastellano = new JRadioButton(ResourceBundle.getBundle("Etiquetas").getString("ConsultMoveme
        rdbtnCastellano.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                Locale.setDefault(new Locale("es"));
                System.out.println("Locale: "+Locale.getDefault());
                redibujar();                }
        });
        buttonGroup.add(rdbtnCastellano);
    }
    return rdbtnCastellano;
}
private JRadioButton getRdbtnEnglish() {
    if (rdbtnEnglish == null) {
        rdbtnEnglish = new JRadioButton(ResourceBundle.getBundle("Etiquetas").getString("ConsultMovements
```

ConsultMovementsGUI klasearen zati horretan, “Locale: ” stringa hainbat aldiz errepikatzen denez, aldagai batean gordeko dugu:

```
String lokala="Locale: ";
private JRadioButton getRdbtnEuskera() {
    if (rdbtnEuskera == null) {
        rdbtnEuskera = new JRadioButton(ResourceBundle.getBundle("Etiquetas").getString("ConsultMovementsG
        rdbtnEuskera.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Locale.setDefault(new Locale("eus"));
                System.out.println(lokala+Locale.getDefault());
                redibujar();                }
        });
        buttonGroup.add(rdbtnEuskera);
    }
    return rdbtnEuskera;
}
private JRadioButton getRdbtnCastellano() {
    if (rdbtnCastellano == null) {
        rdbtnCastellano = new JRadioButton(ResourceBundle.getBundle("Etiquetas").getString("ConsultMovemen
        rdbtnCastellano.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                Locale.setDefault(new Locale("es"));
                System.out.println(lokala+Locale.getDefault());
                redibujar();                }
        });
        buttonGroup.add(rdbtnCastellano);
    }
    return rdbtnCastellano;
}
```

8. Keep unit interfaces small

Azken arau honen arabera, metodo baten parametro kopurua 4 baino txikiagoa izan behar du. Hori beteko ez balitz, objetu berri bat sortu beharko genuke.

Adibidez, metodo honetan, 4 parametro jasotzen ditu. Beraz, sortutako apustuaInfo erabiliko dugu:

```
private void jarraitzaileApustuAnitza(Apustua apustu, Date data, double zenbatekoa, Mugimendua m,
        String jarraitzailea) {
    Erregistratua jarraitzaileaErreg = db.find(domain.Erregistratua.class, jarraitzailea);
    if(jarraitzaileaErreg.getKontuDirua()>=zenbatekoa && jarraitzaileaErreg.isBanned()==false) {
    jarraitzaileaErreg.addMovement(m);
    jarraitzaileaErreg.setKontuDirua(jarraitzaileaErreg.getKontuDirua() - apustu.getZenbatekoa());
    Apustua jarraitzaileApustua = new Apustua(jarraitzailea,apustu.getZenbatekoa());
    jarraitzaileApustua.setErantzunak(apustu.getErantzunak());
    jarraitzaileApustua.setErantzunKop(apustu.getErantzunak().size());
    jarraitzaileaErreg.addApustua(jarraitzaileApustua);
    }else {
        String deskribapena2=("Ezin izan da Apustua kopiatu. ");
        Mugimendua m2=new Mugimendua(data,0,deskribapena2);
        jarraitzaileaErreg.addMovement(m2);
    }
}
```

```
private void jarraitzaileApustuAnitza(ApustuaInfo a) {
    Erregistratua jarraitzaileaErreg = db.find(domain.Erregistratua.class, a.getJarraitzailea());
    if(jarraitzaileaErreg.getKontuDirua()>=a.getZenbatekoa() && jarraitzaileaErreg.isBanned()==false) {
    jarraitzaileaErreg.addMovement(a.getMove());
    jarraitzaileaErreg.setKontuDirua(jarraitzaileaErreg.getKontuDirua() - a.getApustua().getZenbatekoa());
    Apustua jarraitzaileApustua = new Apustua((String)a.getJarraitzailea(),a.getApustua().getZenbatekoa());
    jarraitzaileApustua.setErantzunak(a.getApustua().getErantzunak());
    jarraitzaileApustua.setErantzunKop(a.getApustua().getErantzunak().size());
    jarraitzaileaErreg.addApustua(jarraitzaileApustua);
    }else {
        String deskribapena2=("Ezin izan da Apustua kopiatu. ");
        Mugimendua m2=new Mugimendua(a.getData(),0,deskribapena2);
        jarraitzaileaErreg.addMovement(m2);
    }
}
```

Modu honetan, paramtero bakarra jasoko luke. Horrez gain, addBet metodoko deia ere aldatu beharko genuke:

```
ApustuaInfo a2 = new ApustuaInfo(apustu, data, zenbatekoa, m, jarraitzailea);
jarraitzaileApustuAnitza(a2);
```

Modu honetara, arau hau ere beteko luke metodoak.

C. Amets Carrera

9. "Write short units of code"

Honen arabera, kodigo unitateak 15 lerrokoak izan behar dira gehienez. Hau lortzeko gomendatzen da 15 kode-lerro baina luzeagoak diren unitateak ez idaztea, eta iada idatzita dauden kasuan kode lerro muga gainditzen duten kode-unitate handiak kode-unitate txikiagotan banatzea kode-unitate txiki hauetako bakoitzak 15 lerro baino gutxiago dituztelarik. Horrek gure programaren mantenua errazten du kode-unitate txikiak berrerabiltzea, ulertzea eta testeatzea errazagoa delako.

Adibidez nire taldekideak errefaktorizatutako addBet() metodoa berrerabiliko dugu, izan ere,kode-unitate txikiagotan zatitu badugu ere oraindik 15 lerro baina luzeagoak diren kode-unitateak geratzen dira, hau da kodeak duen luzera:

```
441    public void addBet(Aukera hautatutakoAukera, Apustua apustu) {
442        Erregistratua erreg = db.find(domain.Erregistratua.class, apustu.getLog());
443        db.getTransaction().begin();
444        if(apustu.getErantzunKop()<=1) {
445            Date data=new Date();
446            double zenbatekoa=apustu.getZenbatekoa();
447            String deskribapena="Apustua egin. Dirua: -";
448            Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
449            db.persist(m);
450            erreg.addMovement(m);
451            erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
452            erreg.addApustua(apustu);
453            for(String jarraitzailea: erreg.getJarraitzaileak()) {
454                ApustuaInfo a = new ApustuaInfo(apustu, data, zenbatekoa, m, jarraitzailea);
455                jarraitzaileenApustuBakarra(a);
456            }
457            db.persist(apustu);
458        }else { //apustu Anitza da
459            if(hautatutakoAukera.isAzkenaDa() == true) {
460                Date data=new Date();
461                double zenbatekoa=apustu.getZenbatekoa();
462                String deskribapena="Apustu Anitza egin. Dirua: -";
463                Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
464                db.persist(m);
465                erreg.addMovement(m);
466                erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
467                erreg.addApustua(apustu);
468                for(String jarraitzailea: erreg.getJarraitzaileak()) {
469                    ApustuaInfo a2 = new ApustuaInfo(apustu, data, zenbatekoa, m, jarraitzailea);
470                    jarraitzaileApustuAnitza(a2);
471                }
472                db.persist(apustu);
473            }
474        }
475        db.getTransaction().commit();
476    }
477
```

Hori horrela izanik, errefaktorizazioa ibiliko dugu eta metodo honetatik beste metodo bat aterako dugu, honen tamaina txikitze aldera.

Honela gelditukoo litzateke:

```
440
441     public void addBet(Aukera hautatutakoAukera, Apustua apustu) {
442         Erregistratua erreg = db.find(domain.Erregistratua.class, apustu.getLog());
443         db.getTransaction().begin();
444         if(apustu.getErantzunKop()<=1) {
445             apustuBakarra(apustu, erreg);
446         }else { //apustu Anitza da
447             if(hautatutakoAukera.isAzkenaDa() == true) {
448                 apustuAnitza(apustu, erreg);
449             }
450         }
451         db.getTransaction().commit();
452     }
453
454     private void apustuAnitza(Apustua apustu, Erregistratua erreg) {
455         Date data=new Date();
456         double zenbatekoa=apustu.getZenbatekoa();
457         String deskribapena="Apustu Anitza egin. Dirua: -";
458         Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
459         db.persist(m);
460         erreg.addMovement(m);
461         erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
462         erreg.addApustua(apustu);
463         for(String jarraitzailea: erreg.getJarraitzaileak()) {
464             ApustuaInfo a2 = new ApustuaInfo(apustu, data, zenbatekoa, m, jarraitzailea);
465             jarraitzaileApustuAnitza(a2);
466         }
467         db.persist(apustu);
469
470     private void apustuBakarra(Apustua apustu, Erregistratua erreg) {
471         Date data=new Date();
472         double zenbatekoa=apustu.getZenbatekoa();
473         String deskribapena="Apustua egin. Dirua: -";
474         Mugimendua m=new Mugimendua(data,zenbatekoa,deskribapena);
475         db.persist(m);
476         erreg.addMovement(m);
477         erreg.setKontuDirua(erreg.getKontuDirua() - apustu.getZenbatekoa());
478         erreg.addApustua(apustu);
479         for(String jarraitzailea: erreg.getJarraitzaileak()) {
480             ApustuaInfo a = new ApustuaInfo(apustu, data, zenbatekoa, m, jarraitzailea);
481             jarraitzaileenApustuBakarra(a);
482         }
483         db.persist(apustu);
484     }
```

Ikus dezakegu, addBet metodoaren luzera txikiagotu dugula errefaktorizazioa erabili eta beste 2 metodo bat sartuaz.

10. "Write simple units of code"

Arau honen arabera, bifurkazioen kopurua 4 baino txikiagoa izatea komeni da. Hain zuzen, kodea nondik joan erabakiko duen bifurkazio kopurua 4 baino txikiagoa izan behar da, if-for-while… kopurua. Beste era batera esanda, konplexutasun ziklomatikoa 5 baino txikiagoa. Horretarako, nire taldekideak arazo berdin hau konpontzeko errefaktorizazio bidez sortu duen “apustuakAzteru()” metodoa erabiliko dugu.

```
552
553    public void apustuakAztertu(List<Apustua> apustuLista, Question q, String emaitza) {
554        for(Apustua ap:apustuLista) {
555            if(ap.isBukatua()==false) {
556                for(Aukera auk:ap.getErantzunak()) {
557                    for(Aukera auke:q.getAukerak()) {
558                        if(auke.getAukeraID()==auk.getAukeraID() &&  auk.getAukeraIzena().equals(emaitza)) {
559                            auk.setEgoera("win");
560                            if(ap.getErantzunKop()==1) {
561                              ap.setAsmatua(true);
562                            }
563                        }
564                    }
565                }
566            }
567        }
568    }
```

Bifurkazio kopurua txikitzeko, beste metodo bat aterako dugu:

```
552
553    public void apustuakAztertu(List<Apustua> apustuLista, Question q, String emaitza) {
554        for(Apustua ap:apustuLista) {
555            if(ap.isBukatua()==false) {
556                for(Aukera auk:ap.getErantzunak()) {
557                    apustukoAukerakAztertu(q, emaitza, ap, auk);
558                }
559            }
560        }
561    }
562
563    private void apustukoAukerakAztertu(Question q, String emaitza, Apustua ap, Aukera auk) {
564        for(Aukera auke:q.getAukerak()) {
565            if(auke.getAukeraID()==auk.getAukeraID() &&  auk.getAukeraIzena().equals(emaitza)) {
566                auk.setEgoera("win");
567                if(ap.getErantzunKop()==1) {
568                  ap.setAsmatua(true);
569                }
570            }
571          }
572    }
573
```

Ikus dezakegu, “apustukoAukerakAztertu()” metodoa aterata, 3-ko bifurkazio kopurua gelditzen zaiola hasieran geneukagun “apustuakAztertu()” metodoari.

11. Duplicate code

Kode errepikatuaren zatia, sonarrekin konpondu dugunaren berdina da; hau da, kodea hainbatetan errepikatu beharrean, aldagai batean gordetzea egokiagoa dela. Horretarako, kode hau aldatu dugu:

```
182
183       rdbtnEuskera.addActionListener(new ActionListener() {
184           public void actionPerformed(ActionEvent e) {
185               Locale.setDefault(new Locale("eus"));
186               System.out.println("Locale: "+Locale.getDefault());
187               rdbtnCastellano.setSelected(false);
188               rdbtnEnglish.setSelected(false);
189               nirePantaila.setTitle(ResourceBundle.getBundle("Etiquetas").getString("Ban"));
190               textPaneErabilIzena.setText(ResourceBundle.getBundle("Etiquetas").getString("textErabilIzena"));
191               textPaneErabilData.setText(ResourceBundle.getBundle("Etiquetas").getString("textData"));
192               btnBan.setText(ResourceBundle.getBundle("Etiquetas").getString("btnBan"));
193           }
194       });
195
196
197
198       rdbtnCastellano.addActionListener(new ActionListener() {
199           public void actionPerformed(ActionEvent e) {
200               Locale.setDefault(new Locale("es"));
201               System.out.println("Locale: "+Locale.getDefault());
202               rdbtnEuskera.setSelected(false);
203               rdbtnEnglish.setSelected(false);
204               nirePantaila.setTitle(ResourceBundle.getBundle("Etiquetas").getString("Ban"));
205               textPaneErabilIzena.setText(ResourceBundle.getBundle("Etiquetas").getString("textErabilIzena"));
206               textPaneErabilData.setText(ResourceBundle.getBundle("Etiquetas").getString("textData"));
207               btnBan.setText(ResourceBundle.getBundle("Etiquetas").getString("btnBan"));
208           }
209       });
210
```

“BanGUI” klasearen zati horretan, “Etiquetas” stringa hainbat aldiz errepikatzen denez, aldagai batean gordeko dugu:

```
181
182       String etiketak = "Etiquetas";
183       rdbtnEuskera.addActionListener(new ActionListener() {
184           public void actionPerformed(ActionEvent e) {
185               Locale.setDefault(new Locale("eus"));
186               System.out.println("Locale: "+Locale.getDefault());
187               rdbtnCastellano.setSelected(false);
188               rdbtnEnglish.setSelected(false);
189               nirePantaila.setTitle(ResourceBundle.getBundle(etiketak).getString("Ban"));
190               textPaneErabilIzena.setText(ResourceBundle.getBundle(etiketak).getString("textErabilIzena"));
191               textPaneErabilData.setText(ResourceBundle.getBundle(etiketak).getString("textData"));
192               btnBan.setText(ResourceBundle.getBundle(etiketak).getString("btnBan"));
193           }
194       });
195
196
197
198       rdbtnCastellano.addActionListener(new ActionListener() {
199           public void actionPerformed(ActionEvent e) {
200               Locale.setDefault(new Locale("es"));
201               System.out.println("Locale: "+Locale.getDefault());
202               rdbtnEuskera.setSelected(false);
203               rdbtnEnglish.setSelected(false);
204               nirePantaila.setTitle(ResourceBundle.getBundle(etiketak).getString("Ban"));
205               textPaneErabilIzena.setText(ResourceBundle.getBundle(etiketak).getString("textErabilIzena"));
206               textPaneErabilData.setText(ResourceBundle.getBundle(etiketak).getString("textData"));
207               btnBan.setText(ResourceBundle.getBundle(etiketak).getString("btnBan"));
208           }
209       });
210
```

Modu honetan etiketak aldagaian gordeko dugu “Etiquetas” eta hau berrerabiliko dugu.

12. Keep unit interfaces small

Azken arau honen arabera, metodo baten parametro kopurua 4 baino txikiagoa izan behar du. Hori beteko ez balitz, objetu berri bat sortu beharko genuke.

Adibidez, metodo honetan, 4 parametro jasotzen dira.

```
652
653    public void mezuaErantzunErregistratuari(Date data, Admin admin, String bidaltzailea, String mezua) {
654        db.getTransaction().begin();
655        System.out.println(bidaltzailea);
656        Erregistratua erreg=db.find(domain.Erregistratua.class, bidaltzailea);
657        Mezua m=new Mezua(data,admin.getUser(),mezua);
658        erreg.addMezua(m);
659        db.persist(m);
660        db.getTransaction().commit();
661    }
```

Beraz, sortutako “mezuaInfo” klasea erabiliko dugu:

```
654
655
656    public void mezuaErantzunErregistratuari(MezuaInfo m) {
657        db.getTransaction().begin();
658        System.out.println(m.getBidaltzailea());
659        Erregistratua erreg=db.find(domain.Erregistratua.class, m.getBidaltzailea());
660        Mezua mezua=new Mezua(m.getData(),m.getAdmin().getUser(),m.getMezua());
661        erreg.addMezua(mezua);
662        db.persist(m);
663        db.getTransaction().commit();
664    }
```

Modu honetan, paramtero bakarra jasoko luke. Horrez gain, MezuakAdminGUI-klaseko deia ere aldatu beharko genuke:

```
262                MezuaInfo m = new MezuaInfo(data, a, bidaltzaileIzena, mezua);
263                negozioLogika.mezuaErantzunErregistratuari(m);
```

Modu honetara, arau hau ere beteko luke metodoak.