apustuaEgin metodoan aldaketa:

Egilea: Unax Labaka

LEHEN

apustuaEgin metodoan arazo nagusi bat zegoen kodean, honek 16 ko komplexutasun ziklomatikoa zuen edo kodea oso luzea eta komplexua zen hainbat if eta for begizta berean edukiz.

Arazo nagusia apustua egitean jarraitzaileak egiteko zatian zegoen, non hainbat gauza uksi behar ziren apustuak egin baino lehenago, hau dena for baten barruan, honek denak programa nahiko komplexua sortzen zuen.

:

```
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>();

        +1
     1 for(Pronostikoa p : pronostikoak) {
        pronos = db.find(Pronostikoa.class, p.getIdentifikadorea());
        pronostikoSorta.add(pronos);
    }

    db.getTransaction().begin();
    Apustua apus = erabiltzaile.addApustua(pronostikoSorta, a, null);
        +1
     2 for(Pronostikoa p : pronostikoSorta) {
        p.addApustua(apus);
    }

    db.persist(apus);
    Vector<Errepikapena> jarraitzaile=erabiltzaile.getErrepikatzaileak();
        +1
     3 for(Errepikapena er: jarraitzaile) {
        double apustudiru=0;
           +2 (incl 1 for nesting)
         4 if (er.getHilabeteHonetanGeratzenDena()>0) {

              +3 (incl 2 for nesting)
             5 if (er.getHilabeteHonetanGeratzenDena()>=er.getApustatukoDena()*a) {
                apustudiru=er.getApustatukoDena()*a;
            +1
            } 6 else {
                apustudiru=er.getHilabeteHonetanGeratzenDena();
            }

              +3 (incl 2 for nesting)
             7 if (er.getNork().getDirua() >= apustudiru) {
                Apustua apustu = er.getNork().addApustua(pronostikoSorta, apustudiru, erabiltzaile);
                   +4 (incl 3 for nesting)
                 8 for (Pronostikoa p : pronostikoSorta) {
                    p.addApustua(apustu);
                }
                er.getNork().addMugimendua("Apustu errepikatua egin ("+erabiltzaile+")", -apustudiru, "jokatu");
```

ORAIN

Hau konpontzeko refraktorren **extract method** aukera erabili dugu, honekin lehen adierazi dugun zatia beste metodo berri batera mugitu dugu (**jarraitzaileeiApustuaEgin**). Honek kodea gehiago simplifikatzen du jarraitzaileei apustua egin behar zaien zatia metodo desberdin bat bezala tratatzen delako, kodearen mantenua erraztuz.

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

ezabatuGertaera metodoan aldaketa:

Egilea: Unax Labaka

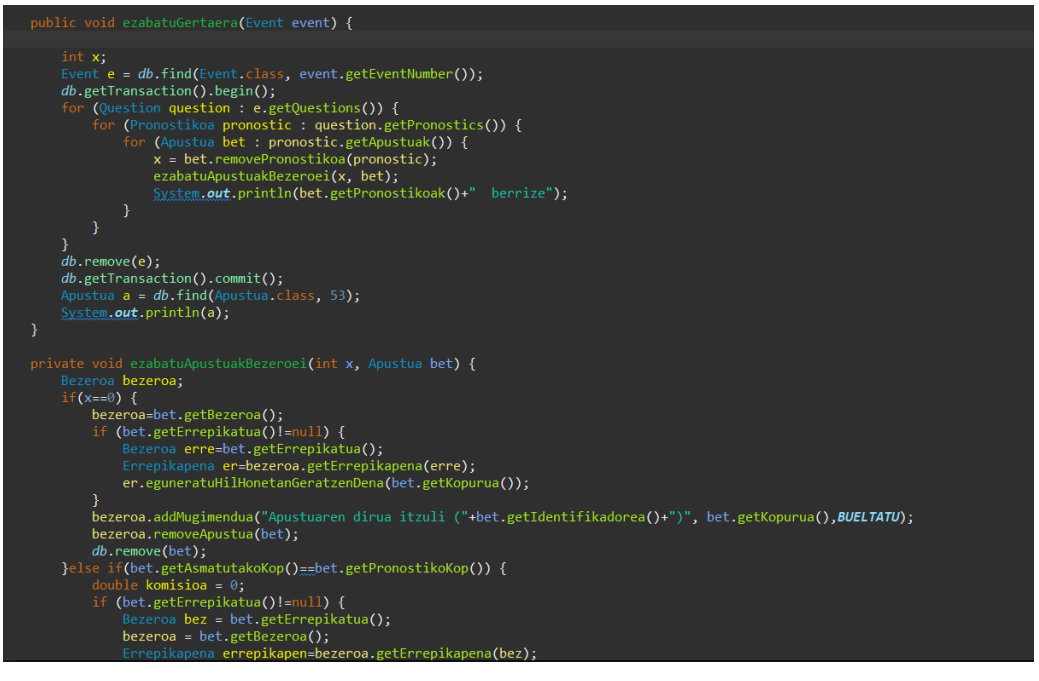
LEHEN

apustuaEgin metodoan bezala ezabatuGertaera oso luzea eta oso komplexua da dituen begizta guztiengatik. Arazoa gertaera osoa ezabatu behar denez honen apustuak + apustuak egiten dituzten bezeroak + bezero hauen jarraitzaileen apustuak ezabatu behar direlako da. Ikus daitekeenez honek metodoa oso komplexua egiten du begizta askorekin.

```
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()==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.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);
                }
                System.out.println(bet.getPronostikoak()+"  berrize");
            }
        }
    }
```

ORAIN

Hau konpontzeko refractor **extractMethod** erabili dugu lehenengo if-etik aurrera beste metodo bat sortzeko (**ezabatuApustuakBezeroei**), gertaeraren apustuak bezeroei eta beraien jarraitzaileei ezabatzen dizkien metodo bat izateko. Horrela kodea simpleagoa egiten dugu eta errazagoa mantenurako.



```
public void ezabatuGertaera(Event event) {

    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);
                ezabatuApustuakBezeroei(x, 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 ezabatuApustuakBezeroei(int x, Apustua bet) {
    Bezeroa bezeroa;
    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()==bet.getPronostikoKop()) {
        double komisioa = 0;
        if (bet.getErrepikatua()!=null) {
            Bezeroa bez = bet.getErrepikatua();
            bezeroa = bet.getBezeroa();
            Errepikapena errepikapen=bezeroa.getErrepikapena(bez);
```

Aldaketa txikiak DataAccess-en:

Egilea: Unax Labaka

Smell “handiak” aldatzeaz aparte beste smell txiki asko ezabatu dira. Adibidez kodea piskat simpleagoa egiteko aldagai pare bat ezabatu dira return bat errazago egin ahal izateko. Horretaz aparte aldagai batzuen karaktereak aldatu dira “araudi estandar” bat jarrai dezaten, hau da parametroen izenak letra txikiz izan behar dira.

LEHEN

```
public Bezeroa getBezeroa(String ErabiltzaileIzena) {
    Bezeroa erabiltzaile = db.find(Bezeroa.class, ErabiltzaileIzena);
    return erabiltzaile;
}

public Langilea getLangilea(String ErabiltzaileIzena) {
    Langilea erabiltzaile = db.find(Langilea.class, ErabiltzaileIzena);
    return erabiltzaile;
}
```

ORAIN

```
public Bezeroa getBezeroa(String erabiltzaileIzena) {
    return db.find(Bezeroa.class, erabiltzaileIzena);
}

public Langilea getLangilea(String erabiltzaileIzena) {
    return db.find(Langilea.class, erabiltzaileIzena);
}
```

BLFacadeImplementation-en aldaketa txikiak:

Egilea: Unax Labaka

LEHEN

Hainbat parametro ez zeuden karaktere egokiekin (aldagaiak txikiz)

```
@WebMethod
public Bezeroa getBezeroa(String ErabiltzaileIzena) {
    dbManager.open(false);
    Bezeroa b = dbManager.getBezeroa(ErabiltzaileIzena);
    dbManager.close();
    return b;
}

@WebMethod
public Langilea getLangilea(String ErabiltzaileIzena) {
    dbManager.open(false);
    Langilea l = dbManager.getLangilea(ErabiltzaileIzena);
    dbManager.close();
    return l;
}
```

ORAIN

Parametroen karaktereak zuzendu egin dira araudia errepetatzeko. ErabiltzaileIzena parametroak erabiltzaileIzena deitzera pasa dira. Horrela kodea ordenatuagoa da, bai klaseak metodoak edo adlagaiek izen sistema konkretu bat errespetatzen dutelako.

```
@WebMethod
public Bezeroa getBezeroa(String erabiltzaileIzena) {
    dbManager.open(false);
    Bezeroa b = dbManager.getBezeroa(erabiltzaileIzena);
    dbManager.close();
    return b;
}

@WebMethod
public Langilea getLangilea(String erabiltzaileIzena) {
    dbManager.open(false);
    Langilea l = dbManager.getLangilea(erabiltzaileIzena);
    dbManager.close();
    return l;
}
```

register metodoan aldaketa:

Egilea: Leire Etxarri

LEHEN

Metodo honen arazoa parametro gehiegi jasotzen dituela da, izan ere, gehienez 7 jasotzea izango litzateke ideala, eta metodo honek 9 jasotzen ditu.

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

ORAIN

Metodo honek jasotzen dituen 9 parametroetatik 8 pertsonaren atributuak dira, hori dela eta, zuzenean pertsonaren parametroa pasatuko diogu, mota String-arekin.

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

public void createEvent(String description, Date eventData) throws EventAlreadyExist{
```

String-ak bikoiztuta:

Egilea: Aitor Castaño

LEHEN

String asko errepikatuta daude, eta testu hori aldatu nahi denean errepikatuta dagoen toki guztietan aldatu beharrko da. Kode hori ez da erosoa mantentzeko.

```
Duplication
m2 = b2.addMugimendua("Apustua egin", -2, 1 "jokatu", UtilDate.newDate(2021, 2, 16));
m3 = b2.addMugimendua("Bankuko diru-sarrera", 30, "bankua", UtilDate.newDate(2021, 2, 15));
Duplication
m4 = b5.addMugimendua("Apustu errepikatua egin ("+b2+")", -4, 2 "jokatu", UtilDate.newDate(2021, 2, 16));
```

ORAIN

Refractor erabilita konpondu daiteke. Testuan sakatu eta Refractor > Extract Constant egin. Horrela konstante bat sortzen da “jokatu” testuarekin eta ez daude errepikapenak. Horrela testua aldatzeko bakarrik aldi batean aldatu behar da.

```
private static final String BUELTATU = "bueltatu";
private static final String AAAAAAAA = "aaaaaaaa";
private static final String IRABAZI = "irabazi";
private static final String JOKATU = "jokatu";
private static final String BANKUA = "bankua";
```

```
Mugimendua m1,m2,m3,m4;
m1 = b2.addMugimendua("Bankuko diru-sarrera", 52, BANKUA, UtilDate.newDate(2021, 2, 15));
m2 = b2.addMugimendua("Apustua egin", -2, JOKATU, UtilDate.newDate(2021, 2, 16));
m3 = b2.addMugimendua("Bankuko diru-sarrera", 30, BANKUA, UtilDate.newDate(2021, 2, 15));
m4 = b5.addMugimendua("Apustu errepikatua egin ("+b2+")", -4, JOKATU, UtilDate.newDate(2021, 2, 16));
```

Vector -> ArrayList:

Egilea: Aitor Castaño

LEHEN

Sonar lit vector jarri beharrean ArrayList jartzea gomendatzen du.

```
public Vector<Langilea> getLangileak() {
    Vector<Langilea> langileak = new Vector<Langilea>();
    TypedQuery<Langilea> query = db.createQuery("SELECT l FROM Langilea l", Langilea.class);
    List<Langilea> list = query.getResultList();
    for (Langilea l : list) {
        langileak.add(l);
    }
    return langileak;
}
```

ORAIN

Refractor erabilita konpondu daiteke. Metodoan sakatuta eta Refractor > Change Method Signature… eginda metodoa bueltatzen duen lista Vector etik ArrayList era aldatu daiteke. Ondoren BLFacadeImplementation eta beste klase batzuetan dauden metodoen definizioak aldatu behar dira ArrayList erabiltzeko.

```
public ArrayList<Langilea> getLangileak() {
    ArrayList<Langilea> langileak = new ArrayList<Langilea>();
    TypedQuery<Langilea> query = db.createQuery("SELECT l FROM Langilea l", Langilea.class);
    List<Langilea> list = query.getResultList();
    for (Langilea l : list) {
        langileak.add(l);
    }
    return langileak;
}
```

BidaliMezua metodoa:

Egilea: Leire Etxarri

LEHEN

8 parametro erabiltzen ditu. Kodearen ulergarritasuna oztopatzen du horrek, gehienez 7 parametro izatea gomendatzen da.

```
public BezeroartekoMezua 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 mezuBerria;
}
```

ORAIN

Gomendioa betetzea ez da zaila, refactor -> Change Method Signature rehabilita parametro ugari kendu eta horien balioak BezeroartekoMezua klasea erabiliz pasatzeko, beste parametro bat gehitu.

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