S O F T W A R E I N G E N I A R I T Z A II

2 0 2 5 K O U R R I A R E N 11 A N

PROIEKTUA 1.2

BEÑAT ERCIBENGOA CALVO
URTZI ETXEGARAI TABERNA
EKAITZ PINEDO ALVAREZ

Github-eko link-a:https://github.com/UrtziE/SI2Proiektua

"Write short units of code"

Jarraibideak:
• Mugatu kode-unitateen luzera 15 lerrotara.
• Hori lortzeko, ez idatzi hasieratik 15 lerro baino luzeagoak diren unitateak, edo zatitu unitate luzeak hainbat unitate txikiagotan, unitate bakoitzak gehienez 15 lerro izan arte.
• Honek mantengarritasuna hobetzen du, izan ere, unitate txikiak errazago ulertzen, probatzen eta berrerabiltzen dira.

Gure kodean agertu diren metodoak ez dutenak erregela hau betetzen:

- erreklamazioaOnartuEdoDeuseztatu (Beñat Ercibengoa Calvo)

Arazoa:
20 lerroko metodoa, izenean bi gauza desberdin egin ditzakeela jartzen du eta nahasgarria da.
Hasierako kodea:

```
public void erreklamazioaOnartuEdoDeuseztatu(Erreklamazioa erreklamazioa, float kantitatea, boolean onartuta) {

    db.getTransaction().begin();
    Erreklamazioa e = db.find(Erreklamazioa.class, erreklamazioa.getId());
    e.setEgoera(EgoeraErreklamazioa.BUKATUA);
    Profile nork = db.find(Profile.class, e.getNork().getUser());
    Profile nori = db.find(Profile.class, e.getNori().getUser());
    Profile admin = e.getAdmin();
    RideRequest request = e.getErreserba();
    Ride r = request.getRide();
    if (onartuta) {
        nork.gehituDirua(kantitatea);
        nori.kenduDirua(kantitatea);
        nork.addErreklamazioMezu(0, r, erreklamazioa);
        nork.gehituMezuaTransaction(9, kantitatea, request);
        nori.gehituMezuaTransaction(11, kantitatea, request);
        admin.addErreklamazioMezu(2, r, erreklamazioa);
    } else {
        nork.addErreklamazioMezu(1, r, erreklamazioa);
        admin.addErreklamazioMezu(3, r, erreklamazioa);
    }

    e.setWhenDecided(new Date());
    db.getTransaction().commit();
}
```

Konponketa:
Errefaktorizazioaren ondoren 14 lerro ditu. Orain izen ulergarria du eta gainera gainetik irakurtzerakoan argi eta garbi ulertzen da erreklamazioak onartu edo deuseztatzen dituela parametroz pasatako balioaren arabera.

Kode errefaktorizatua:

```
public void erreklamazioaProzesatu(Erreklamazioa erreklamazioa, float kantitatea, boolean onartuta) {
    db.getTransaction().begin();
    Erreklamazioa e = db.find(Erreklamazioa.class, erreklamazioa.getId());
    e.setEgoera(EgoeraErreklamazioa.BUKATUA);
    Profile nork = db.find(Profile.class, e.getNork().getUser());
    Profile nori = db.find(Profile.class, e.getNori().getUser());
    Profile admin = e.getAdmin();
    RideRequest request = e.getErreserba();
    Ride r = request.getRide();
    if (onartuta) {
        erreklamazioaOnartu(nork, nori, admin, kantitatea, erreklamazioa);
    } else {
        erreklamazioaDeuseztatu(nork, admin, kantitatea, erreklamazioa);
    }
    e.setWhenDecided(new Date());
    db.getTransaction().commit();
}
```

```
private void erreklamazioaOnartu(Profile nork, Profile nori, Profile admin, float kantitatea, Erreklamazioa e) {
    RideRequest request = e.getErreserba();
    Ride r = request.getRide();

    nork.gehituDirua(kantitatea);
    nori.kenduDirua(kantitatea);
    nork.addErreklamazioMezu(0, r, e);
    nork.gehituMezuaTransaction(9, kantitatea, request);
    nori.gehituMezuaTransaction(11, kantitatea, request);
    admin.addErreklamazioMezu(2, r, e);
}

private void erreklamazioaDeuseztatu(Profile nork, Profile admin, float kantitatea, Erreklamazioa e) {
    RideRequest request = e.getErreserba();
    Ride r = request.getRide();

    nork.addErreklamazioMezu(1, r, e);
    admin.addErreklamazioMezu(3, r, e);
}
```

- erreserbatu(Ekaitz Pinedo)

Arazoa:

21 lerroko metodoa.

Hasierako kodea:

```
public RideRequest erreserbatu(RideRequest rq) {
    Ride ride = rq.getRide();
    String requestFrom = rq.getFromRequested();
    String requestTo = rq.getToRequested();
    int seats = rq.getSeats();
    Traveller traveller = rq.getTraveller();
    Date time = rq.getWhenRequested();

    RideRequest request=null;
    db.getTransaction().begin();
    Ride rd = db.find(Ride.class, ride.getRideNumber());
    if(rd.lortuEserlekuKopMin(requestFrom, requestTo)>=seats) {
    Traveller t = db.find(Traveller.class, traveller.getUser());

    t.kenduDirua(rd.lortuBidaiarenPrezioa(requestFrom, requestTo) * seats);
    request = t.addRequest(time, rd, seats, requestFrom, requestTo);
    rd.addRequest(request);

    t.gehituMezuaTransaction(0, rd.lortuBidaiarenPrezioa(requestFrom, requestTo) * seats, request);

    db.getTransaction().commit();
    }else {
        System.out.println();
        db.getTransaction().commit();
        return null;
    }

    return request;

}
```

Konponketa:

Errefaktorizazioaren ondoren 13 lerro ditu. Bi metodo privatu sortu ditut. Lehenengoa eserleku kopurua egokia baden konprobatzen du, eskatutako eserlekuak gehiegi badira ez da erreserbatzen eta null itzultzen du; bestela, bigarren metodora sartzen da erreserba eskaera sortuz eta bidaiariari gehitu. Azkenik erreserba hori bidaian gordetzen da eta datu baseko egoera berria gordetzen da.

Kode errefaktorizatua:

```
public RideRequest erreserbatu(RideRequest request) {
    Ride ride = request.getRide();
    Ride rd = db.find(Ride.class, ride.getRideNumber());
    db.getTransaction().begin();


    if(!eserlekuKopEgokia(rd,request)) {
        db.getTransaction().commit();
        return null;
    }

    RideRequest rq=createRideRequest(rd,request);
    rd.addRequest(request);
    db.getTransaction().commit();

    return rq;

}

private boolean eserlekuKopEgokia(Ride rd,RideRequest request) {
    String requestFrom = request.getFromRequested();
    String requestTo = request.getToRequested();
    int seats = request.getSeats();

    return (rd.lortuEserlekuKopMin(requestFrom, requestTo)>=seats);
}
private RideRequest createRideRequest(Ride ride,RideRequest request) {
    Traveller traveller = request.getTraveller();
    String requestFrom = request.getFromRequested();
    String requestTo = request.getToRequested();
    int seats = request.getSeats();
    Date time = request.getWhenRequested();
    Traveller t = db.find(Traveller.class, traveller.getUser());
    t.kenduDirua(ride.lortuBidaiarenPrezioa(requestFrom, requestTo) * seats);
    request = t.addRequest(time, ride, seats, requestFrom, requestTo);

    t.gehituMezuaTransaction(0, ride.lortuBidaiarenPrezioa(requestFrom, requestTo) * seats, request);

    return request;
}
```

- DataAccess/onartuEdoDeuseztatu(Urtzi Etxegarai)

Arazoa:

- 16 lerroko metodoa, badago metodo bat antzeko izenarekin, beraz ere aldatuko dugu

Hasierako kodea:

```
public void onartuEdoDeuseztatu(RideRequest request, boolean onartuta) {

    db.getTransaction().begin();

    RideRequest r = db.find(RideRequest.class, request.getId());
    Ride ride = r.getRide();

    if (onartuta) {
        // Aldatu
        ride.kenduSeatGeltokiei(r.getSeats(), r.getFromRequested(), r.getToRequested());
        // ride.setBetMinimum((ride.getnPlaces() - r.getSeats()));
        r.setState(EgoeraRideRequest.ACCEPTED);
        ride.deuseztatuSeatKopuruBainaHandiagoa(r);
        if (ride.getnPlaces() == 0) {
            ride.setEgoera(EgoeraRide.TOKIRIK_GABE);
        }

    } else {
        Traveller t = r.getTraveller();
        t.gehituDirua(r.getPrezioa());
        r.setState(EgoeraRideRequest.REJECTED);
        t.gehituMezuaTransaction(1, r.getPrezioa(), r); // Dirua itzuli
    }

    r.setWhenDecided(new Date());
    db.getTransaction().commit();
}
```

Konponketa:

Ikusi daitekenez, lehen metodo nagusiak 2 ekintza egiten zituen: Erreserba onartu eta erreserba deuseztatu. Orain 2 ekintza horiek funtzio laguntzaile batzuk gauzatzen dituzte. Beraz, kode nagusia askoz ere motzagoa geratu da eta orain 9 lerro ditu, metodo laguntzaileak 5 eta 4 lerro dituzte hurrenez hurren, beraz, oso kode ulergarria geratu da, noski, funtzioei izen egokiak jarriz.

Kode errefaktorizatua:

```
public void onartuEdoDeuseztatuErreserba(RideRequest request, boolean onartuta) {

    db.getTransaction().begin();

    RideRequest r = db.find(RideRequest.class, request.getId());
    Ride ride = r.getRide();

    if (onartuta) {
        onartuErreserba(ride,r);
    } else {
        deuseztatuErreserba(r);
    }
    r.setWhenDecided(new Date());
    db.getTransaction().commit();
}
```

```
private void onartuErreserba(Ride ride, RideRequest r) {
    ride.kenduSeatGeltokiei(r.getSeats(), r.getFromRequested(), r.getToRequested());
    r.setState(EgoeraRideRequest.ACCEPTED);
    ride.deuseztatuSeatKopuruBainaHandiagoa(r);
    if (ride.getnPlaces() == 0) {
        ride.setEgoera(EgoeraRide.TOKIRIK_GABE);
    }
}
private void deuseztatuErreserba(RideRequest r) {
    Traveller t = r.getTraveller();
    t.gehituDirua(r.getPrezioa());
    r.setState(EgoeraRideRequest.REJECTED);
    t.gehituMezuaTransaction(1, r.getPrezioa(), r); // Dirua itzuli
}
```

"Write simple units of code"

Jarraibideak:

• Mugatu adarkatze-puntu kopurua unitateko gehienez 4ra.

• Hori lortzeko, zatitu unitate konplexuak sinpleagoetan eta saihestu unitate konplexuak oro har.

• Honek mantengarritasuna hobetzen du, izan ere, adarkatze-puntu gutxiago edukitzeak unitateak errazago aldatzeko eta probatzeko modukoak egiten ditu.

Gure kodean agertu diren metodoak ez dutenak erregela hau betetzen:

- konprobatuBidaienEgunak (Beñat Ercibengoa Calvo)

Arazoa:

Metodo honek 7 bidegurutze ditu. Ez da batere sinplea eta irakurtgaitza ere baita.

Hasierako kodea:

```
private void konprobatuBidaienEgunak() {
    TypedQuery<Ride> query = db.createQuery("SELECT r FROM Ride r", Ride.class);
    List<Ride> rideP = query.getResultList();
    db.getTransaction().begin();
    Date gaur = new Date();
    for (Ride ride : rideP) {
        if ((ride.getEgoera().equals(EgoeraRide.MARTXAN) || ride.getEgoera().equals(EgoeraRide.TOKIRIK_GABE))
                & ride.getDate().before(new Date())) {
            Ride ri = db.find(Ride.class, ride.getRideNumber());
            ri.setEgoera(EgoeraRide.PASATUA);
        } else if ((gaur.getTime() - ride.getDate().getTime()) / (1000 * 60 * 60 * 24) > 3) {
            if (ride.getEgoera().equals(EgoeraRide.PASATUA)) {
                for (RideRequest r : ride.getEskakizunak()) {
                    if(r.getState().equals(EgoeraRideRequest.ACCEPTED)) {
                        r.setState(EgoeraRideRequest.DONE);

                    }
                    r.setBaloratuaDriver(true);
                    r.setBaloratuaTraveller(true);
                    r.setErreklamatuaDriver(true);
                    r.setErreklamatuaTraveller(true);
                }
            } else if (ride.getEgoera().equals(EgoeraRide.DONE) || ride.getEgoera().equals(EgoeraRide.NOT_DONE)) {

            }
        }

    }
    db.getTransaction().commit();

}
```

Konponketa:

Errefaktorizatu ondoren, bidegurutze bakarra du metodo nagusiak. Metodo nagusian egiten ziren gauza desberdin guztiak hainbat metodo laguntzaile txikiagoetan banatu ditugu, bakoitza izen egoki batez definituz denak ulertu dadin.

Kode errefaktorizatua:

```
private void konprobatuBidaienEgunak() {
    TypedQuery<Ride> query = db.createQuery("SELECT r FROM Ride r", Ride.class);
    List<Ride> rideP = query.getResultList();
    db.getTransaction().begin();
    Date gaur = new Date();
    for (Ride ride : rideP) {
        konprobatuBidaia(gaur, ride);
    }
    db.getTransaction().commit();
}

private void konprobatuBidaia(Date gaur, Ride ride) {
    if ((ride.getEgoera().equals(EgoeraRide.MARTXAN) || ride.getEgoera().equals(EgoeraRide.TOKIRIK_GABE))
        & ride.getDate().before(new Date())) {
        bidaiaItxi(ride);
    } else if ((gaur.getTime() - ride.getDate().getTime()) / (1000 * 60 * 60 * 24) > 3) {
        bidaiEskaerakProzesatu(ride);
    }
}

private void bidaiaItxi(Ride ride) {
    Ride ri = db.find(Ride.class, ride.getRideNumber());
    ri.setEgoera(EgoeraRide.PASATUA);
}

private void bidaiEskaerakProzesatu(Ride ride) {
    if (ride.getEgoera().equals(EgoeraRide.PASATUA)) {
        for (RideRequest rr : ride.getEskakizunak()) {
            pasatutakoBidaiEskaerakProzesatu(rr);
        }
    }
}

private void pasatutakoBidaiEskaerakProzesatu(RideRequest rr) {
    if(rr.getState().equals(EgoeraRideRequest.ACCEPTED)) {
        rr.setState(EgoeraRideRequest.DONE);
    }
    rr.setBaloratuaDriver(true);
    rr.setBaloratuaTraveller(true);
    rr.setErreklamatuaDriver(true);
    rr.setErreklamatuaTraveller(true);
}
```

- Domain/Ride/kenduSeatGeltokiei metodoa(Urtzi Etxegarai)

Arazoa:

Metodo honek 5 bidegurutze ditu Konplexutasun ziklomatikoa 6 da. Ez da batere sinplea eta ulertzea zaila egin daiteke

Hasierako kodea:

```
public void kenduSeatGeltokiei(int seats, String from, String to) {
    boolean hasi=false;
    boolean daude= geltokiListContains(from,to);
    if(daude) {
    for(int i=0;i<geltokiList.size();i++) {
        if(geltokiList.get(i).getTokiIzen().equals(from)) {
            hasi=true;
        }
        if(geltokiList.get(i).getTokiIzen().equals(to)) {
            hasi=false;
        }
        if(hasi) {
            geltokiList.get(i).kenduSeatKop(seats);
        }

    }
    }

}
```

Konponketa:

Orain metodo nagusian 3 bidegurutze daude(Konplexutasun ziklomatikoa 4) metodo nagusian. Metodo nagusiak egiten duena da geltokiLista errekorritu eta from-etik to-ra dauden geltokiei eserlekuak kendu egiten dizkio. Metodo laguntzaileak egiten duena da begizta-ren i posizioa ea from-etik to arteko geltokietan dagoen konprobatu. Beraz, bidegurutzeen arazoa konpontzeko, funtzio laguntzaile bat sortu dugu funtzio nagusiak egiten dituen lanak banatzeko.(Aldagai batzuen balioak aldatu ditut ulergarriagoa izateko hasi⇒fromToBarruan).

Kode errefaktorizatua:

```
public void kenduSeatGeltokiei(int seats, String from, String to) {
    boolean fromToBarruan=false;
    boolean daude= geltokiListContains(from,to);
    if(daude) {
    for(Geltoki g: geltokiList) {
        fromToBarruan=frometikToraDagoIGeltokia(g,from,to,fromToBarruan);
        if(fromToBarruan) {
            g.kenduSeatKop(seats);
        }

    }
    }

}
private boolean frometikToraDagoIGeltokia(Geltoki g,String from,String to,boolean fromToBarruan) {
    if(g.getTokiIzen().equals(from)) {
        return true;
    }
    if(g.getTokiIzen().equals(to)) {
        return false;
    }
    return fromToBarruan;
}
```

- lortuEserlekuKopMin(Ekaitz Pinedo)

Arazoa:

Metodo honen adarkatze-puntu kopurua 5ko da.

Hasierako kodea:

```
public int lortuEserlekuKopMin(String from, String to) {
    boolean hasi=false;
    int min=1000;
    for(int i=0;i<geltokiList.size();i++) {
        if(geltokiList.get(i).getTokiIzen().equals(from)) {
            hasi=true;
            min=geltokiList.get(i).getEserleku();
        }
        if(geltokiList.get(i).getTokiIzen().equals(to)) {
            hasi=false;
            //PENTSATU EA JARRI HEMEN RETURN
        }
        if(hasi) {
            if(min>geltokiList.get(i).getEserleku()) {
                min=geltokiList.get(i).getEserleku();
            }
        }
    }
    return min;
}
```

Konponketa:

Errefaktorizazioaren ondoren metodo nagusia asko sinplifikatu da bakarrik 2 adarkatze puntu izanez. Hau lortzeko metodo pribatu berri bat sortu dut eserleku minimo berria dagoen edo ez konprobatzeko (metodo nagusiaren barruan egon zitekeen baina horrela irakurgarritasuna hobea da), eta lehen sortutako metodo pribatu berrerabili dut jakiteko geltoki horren eserlekuak konprobatu behar diren edo ez jakiteko.

Kode errefaktorizatua:

```
public int lortuEserlekuKopMin(String from, String to) {
    boolean hasi=false;
    int min = Integer.MAX_VALUE;
    for(Geltoki g: geltokiList) {
        hasi=frometikToraDagoIGeltokia(g,from,to,hasi);
        if(hasi) {
            min=eserlekuKopMinKonprobatu(g,min);
        }
    }
    return min;
}
private int eserlekuKopMinKonprobatu(Geltoki geltoki,int min) {
    if(min>geltoki.getEserleku()) {
        min=geltoki.getEserleku();
    }
    return min;
}

private boolean frometikToraDagoIGeltokia(Geltoki g,String from,String to,boolean fromToBarruan) {
    if(g.getTokiIzen().equals(from)) {
        return true;
    }
    if(g.getTokiIzen().equals(to)) {
        return false;
    }
    return fromToBarruan;
}
```

“Duplicate code”

Jarraibideak:
• Ez kopiatu kodea.
• Horren ordez, idatzi kode berrerabilgarria eta orokorra, edo erabili lehendik dauden metodoak.
• Honek mantengarritasuna hobetzen du, izan ere, kodea kopiatzen denean, akatsak hainbat tokitan zuzendu behar izaten dira, eta hori eraginkortasunik gabea eta akatsak eragiteko arriskutsua da.

Gure kodean agertu diren metodoak ez dutenak erregela hau betetzen:

- ikusitakoAlerta (Beñat Ercibengoa Calvo)

Arazoa:

Bi metodo hauek identikoak dira. Desberdintasun bakarra da batean query-ko parametro bat true dela eta bestea false. Honek kode errepikatu asko sortzen du.

Hasierako kodea:

```
public List<Mezua> ikusitakoAlerta(Traveller traveller) {
    TypedQuery<Mezua> query = db
            .createQuery("SELECT m FROM Mezua m WHERE  m.type=?2 AND m.irakurrita=?3 AND m.p=?4", Mezua.class);
    query.setParameter(2, 2);
    query.setParameter(3, true);
    query.setParameter(4, traveller);
    List<Mezua> alertaMezuak = query.getResultList();
    return alertaMezuak;
}

public List<Mezua> getIkusiGabeAlerta(Traveller traveller) {
    TypedQuery<Mezua> query = db
            .createQuery("SELECT m FROM Mezua m WHERE  m.type=?2 AND m.irakurrita=?3 AND m.p=?4", Mezua.class);
    query.setParameter(2, 2);
    query.setParameter(3, false);
    query.setParameter(4, traveller);
    List<Mezua> alertaMezuak = query.getResultList();

    return alertaMezuak;
}
```

Konponketa:

Metodo berri orokor bat sortu dugu, zeinetan balio boolear batez irakurritako edo irakurri gabeko mezuak lortu nahi ditugun adierazten diogun. Aurreko bi metodo originalek metodo honi deituko diote true/false parametroekin. Bi metodo originalak ezabatu ditzakegu eta getAlertarekin geratu. Hala eta guztiz ere, honek ekarriko lituzke aldaketak BLfacaden eta GUI-ko hainbat klaseetan. Horregatik, soluzio hau eraginkorra da, DataAccess-en barne egitura oso gutxi aldatuz kode errepikatua ezabatzea lortzen degulako kanpoko egiturari eraginik egin gabe(bestela, domino eran erroreak aterako lirateke: DataAccess → BLFacadeImplementation → BLFacade → GUIKlaseak).

Kode errefaktorizatua:

```java
public List<Mezua> getAlerta(Traveller traveller, boolean irakurrita) {
    TypedQuery<Mezua> query = db
            .createQuery("SELECT m FROM Mezua m WHERE  m.type=?2 AND m.irakurrita=?3 AND m.p=?4", Mezua.class);
    query.setParameter(2, 2);
    query.setParameter(3, irakurrita);
    query.setParameter(4, traveller);
    List<Mezua> alertaMezuak = query.getResultList();
    return alertaMezuak;
}

public List<Mezua> ikusitakoAlerta(Traveller traveller) {
    return getAlerta(traveller, true);
}

public List<Mezua> getIkusiGabeAlerta(Traveller traveller) {
    return getAlerta(traveller, false);
}
```

- getMezuak eta getErreklamazioMezuak(EkaitzPinedo)

Arazoa:

Bi metodo hauek inplementazio oso antzeko bat dute.

Hasierako kodea:

```java
/**
 * Metodo honek erabiltzaile batek dituen mezuen lista itzultzen du
 *
 * @param p erabiltzaile horren profila
 * @return mezuen lista
 */
public List<Mezua> getMezuak(Profile p) {
    db.getTransaction().begin();
    Profile profile = db.find(Profile.class, p.getUser());
    List<Mezua> mList1 = profile.getMezuList();
    db.getTransaction().commit();
    List<Mezua> mList = new LinkedList<Mezua>();
    for (Mezua mezu : mList1) {
        if (mezu.getType() == 1) {
            mList.add(mezu);
        }
    }
    return mList;
}

public List<Mezua> getErreklamazioMezuak(Profile p) {
    db.getTransaction().begin();
    Profile profile = db.find(Profile.class, p.getUser());
    List<Mezua> mList1 = profile.getMezuList();
    db.getTransaction().commit();
    List<Mezua> mList = new LinkedList<Mezua>();
    for (Mezua mezu : mList1) {
        if (mezu.getType() == 3) {
            mList.add(mezu);
        }
    }
    return mList;
}
```

Konponketa:

Errefaktorizatu dut metodo pribatu orokor bat sortuz eta horrela kodea ez da errepikatzen.

Kode errefaktorizatua:

```
/**
 * Metodo honek erabiltzaile batek dituen mezuen lista itzultzen du
 *
 * @param p erabiltzaile horren profila
 * @return mezuen lista
 */
public List<Mezua> getMezuak(Profile p) {
    return lortuMezuak(p,1);
}

public List<Mezua> getErreklamazioMezuak(Profile p) {
    return lortuMezuak(p,3);
}
private List<Mezua> lortuMezuak(Profile p,int type){
    db.getTransaction().begin();
    Profile profile = db.find(Profile.class, p.getUser());
    List<Mezua> mList1 = profile.getMezuList();
    db.getTransaction().commit();
    List<Mezua> mList = new LinkedList<Mezua>();
    for (Mezua mezu : mList1) {
        if (mezu.getType() == type) {
            mList.add(mezu);
        }
    }
    return mList;
}
```

- Domain/Mezua(Urtzi Etxegarai)

Arazoa:

Ikusi egiten den bezala Rides klasean kode duplikatu asko dago, etiketen sorrerarekin dago lotuta gehienbat.

Hasierako kodea:

```
public void zeinTransaction(int i) {
    switch (i) {
    // Traveller
    case 0:

        typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.Requested") + ":  "
                + erreserba.mezua() + "__" + ride;
        datamezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.DataRequest") + " " + when;
        diruMezu = "-" + kantitatea + "€";

        break;
    case 1:

        typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.Rejected") + ":  "
                + erreserba.mezua() + "__" + ride.mezua();
        datamezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.DataRejected") + " " + when;
        diruMezu = "+" + kantitatea + "€";

        break;
    case 2:

        typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.Canceled") + ":  " + ride.mezua();
        datamezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.DataCanceled") + " " + when;
        diruMezu = "+" + kantitatea + "€";

        break;
    case 3:
        typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.Deposite") + ": ";
        datamezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.DataDeposite") + " " + when;
        diruMezu = "+" + kantitatea + "€";
        break;
    // Driver
    case 4:
        typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.NotDone") + ":    " + ride.mezua();
        datamezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.DataNotDone") + " " + when;
        diruMezu = "+" + kantitatea + "€";

        break;
    case 5:
        typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.Canceled") + ":   " + ride.mezua();
        datamezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.DataCanceled") + " " + when;
        diruMezu = "+" + kantitatea + "€";
        break;
```

```
        case 6:
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.Withdraw") + ":";
            datamezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.DataWithdraw") + " " + when;
            diruMezu = "-" + kantitatea + "€";
            break;
        case 7:
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.Done") + ":    " + ride.mezua();
            datamezua = ResourceBundle.getBundle("Etiquetas").getString(DATADONE) + " " + when;
            diruMezu = "+" + kantitatea + "€";
            break;
        case 8:
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.NewErreklamazioa") + ":    "
                    + ride.mezua();
            datamezua = ResourceBundle.getBundle("Etiquetas").getString(DATADONE) + " " + when;
            diruMezu = "+" + kantitatea + "€";
            break;
        case 9:
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.ErreklamazioaAccepted") + ":    "
                    + ride.mezua();
            datamezua = ResourceBundle.getBundle("Etiquetas").getString(DATADONE) + " " + when;
            diruMezu = "+" + kantitatea + "€";
            break;
        case 10:
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.ErreklamazioaRejected") + ":    "
                    + ride.mezua();
            datamezua = ResourceBundle.getBundle("Etiquetas").getString(DATADONE) + " " + when;
            break;
        case 11:
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.Erreklamatuta") + ":    "
                    + ride.mezua();
            datamezua = ResourceBundle.getBundle("Etiquetas").getString(DATADONE) + " " + when;
            diruMezu = "-" + kantitatea + "€";
            break;
        }

    }

    public void zeinErreklamazio(int i) {
        switch (i) {
        case 0:
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.ErreklamazioaAccepted") + ":    "
                    + ride.mezua();
            datamezua = ResourceBundle.getBundle("Etiquetas").getString(DATADONE) + " " + when;
            diruMezu = "+" + kantitatea + "€";
            break;
        case 1:
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.ErreklamazioaRejected") + ":    "
                    + ride.mezua();
            datamezua = ResourceBundle.getBundle("Etiquetas").getString(DATADONE) + " " + when;
            break;
        case 2:
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.ErreklamazioBukatuta") + ":    "
                    + ride.mezua()+"  Accepted";
            datamezua = ResourceBundle.getBundle("Etiquetas").getString(DATADONE) + " " + when;
            break;
        case 3:
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString("Mezuak.ErreklamazioBukatuta") + ":    "
                    + ride.mezua() +"  Rejected";
            datamezua = ResourceBundle.getBundle("Etiquetas").getString(DATADONE) + " " + when;
            break;
        }
    }
```

SI2RidesProiektua > src > main/java > domain > Mezua.java

Duplicated Lines (%) 15.5%

Konponketa:

Kode duplikatu hori saihesteko, metodo bat sortu egin dut “sortuMezua” izenekoa, honi, behar dituen etiketen izenak aldagai moduan pasatzen zaizkio, eta bi aldagai gehiago jakiteko mezuan “-” edo “+” agertu behar den edo ezer. Baita ere mezu espezial batzuk

kontuan hartzen dira if batzuen bidez. Hau guztiagatik ikusi daiteke kodea oso txukun eta editatzeko erraza geratu dela eta kode duplikatu asko kendu egin du

Kode errefaktorizatua:

```
private void sortuMezua(String mezua, String data, String gehituEdoKendu, boolean erreserbaDa) {
    if (erreserbaDa) {
        typerenMezua = ResourceBundle.getBundle("Etiquetas").getString(mezua) + ":  "+ erreserba.mezua() + "__" + ride.mezua();
    } else {
        if (ride != null) {
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString(mezua) + ":   " + ride.mezua();
        } else {
            typerenMezua = ResourceBundle.getBundle("Etiquetas").getString(mezua) + ": ";
        }
        datamezua = ResourceBundle.getBundle("Etiquetas").getString(data) + " " + when;
        diruMezu = gehituEdoKendu + kantitatea + "€";
    }
}
```

```
public void zeinTransaction(int i) {
    switch (i) {
    // Traveller
    case 0:
        sortuMezua("Mezuak.Requested","Mezuak.DataRequest","-",true);


        break;
    case 1:
        sortuMezua("Mezuak.Rejected","Mezuak.DataRejected","+",true);
        break;
    case 2:
        sortuMezua("Mezuak.Canceled","Mezuak.DataCanceled","+",EZDAERRESERBA);
        break;
    case 3:
        sortuMezua("Mezuak.Deposite","Mezuak.DataDeposite","+",EZDAERRESERBA);
        break;
    // Driver
    case 4:
        sortuMezua("Mezuak.NotDone","Mezuak.DataNotDone","+",EZDAERRESERBA);
        break;
    case 5:
        sortuMezua("Mezuak.Canceled","Mezuak.DataCanceled","+",EZDAERRESERBA);
        break;
    case 6:
        sortuMezua("Mezuak.Withdraw","Mezuak.DataWithdraw","-",EZDAERRESERBA);
        break;
    case 7:
        sortuMezua("Mezuak.Done",DATADONE,"+",EZDAERRESERBA);
        break;
    case 8:
        sortuMezua("Mezuak.NewErreklamazioa",DATADONE,"+",EZDAERRESERBA);
        break;
    case 9:
        sortuMezua("Mezuak.ErreklamazioaAccepted",DATADONE,"+",EZDAERRESERBA);
        break;
    case 10:
        sortuMezua("Mezuak.ErreklamazioaRejected",DATADONE,"",EZDAERRESERBA);
        break;
    case 11:
        sortuMezua("Mezuak.Erreklamatuta",DATADONE,"-",EZDAERRESERBA);
        break;
    }

}
```

```java
public void zeinErreklamazio(int i) {
    switch (i) {
    case 0:
        sortuMezua("Mezuak.ErreklamazioaAccepted",DATADONE,"+",EZDAERRESERBA);
        break;
    case 1:
        sortuMezua("Mezuak.ErreklamazioaRejected",DATADONE,"",EZDAERRESERBA);
        break;
    case 2:
        sortuMezua("Mezuak.ErreklamazioBukatuta",DATADONE,"",EZDAERRESERBA);
        typerenMezua = typerenMezua+"  Accepted";
        break;
    case 3:
        sortuMezua("Mezuak.ErreklamazioBukatuta",DATADONE,"",EZDAERRESERBA);
        typerenMezua =typerenMezua +"  Rejected";

        break;
    }
}

public void zeinAlerta() {
    sortuMezua("Mezuak.EskatutakoAlerta",DATADONE,"",EZDAERRESERBA);
    typerenMezua = typerenMezua + alerta.toString();
}
```

SI2RidesProiektua > src > main/java > domain > Mezua.java

Duplicated Lines (%) 0.0%

"Keep unit interfaces small"

Jarraibideak:
• Mugatu unitateko parametro kopurua gehienez 4ra.
• Horretarako, parametroak objektuetan bildu.
• Honek mantengarritasuna hobetzen du, izan ere, parametro gutxi izateak unitateak errazago ulertzeko eta berrerabiltzeko aukera ematen du.

Gure kodean agertu diren metodoak ez dutenak erregela hau betetzen:

- erreserbatu (Beñat Ercibengoa Calvo)

Arazoa:
6 parametroko metodoa da erreserbatu.
Hasierako kodea:

```
public RideRequest erreserbatu(Date time, Ride ride, Traveller traveller, int seats, String requestFrom, String requestTo) {
    RideRequest request=null;
    db.getTransaction().begin();
    Ride rd = db.find(Ride.class, ride.getRideNumber());
    if(rd.lortuEserlekuKopMin(requestFrom, requestTo)>=seats) {
    Traveller t = db.find(Traveller.class, traveller.getUser());

    t.kenduDirua(rd.lortuBidaiarenPrezioa(requestFrom, requestTo) * seats);
    request = t.addRequest(time, rd, seats, requestFrom, requestTo);
    rd.addRequest(request);
    // HOBETZEKOA RIDEK IZATEA GEHITU MEZUA RIDE METODOA

    // hobetzekoa, travellerek izatea gehitu mezua metodoa,
    t.gehituMezuaTransaction(0, rd.lortuBidaiarenPrezioa(requestFrom, requestTo) * seats, request);// Traveller ride
                                                                                                    // requested

    //System.out.println("Eta" + t + " has been updated");
    db.getTransaction().commit();
    }else {
        System.out.println();
        db.getTransaction().commit();
        return null;
    }

    return request;

}
```

Konponketa:
ErreserbaEskaera sortu dugu; hau da, parametro horiek guztiak biltzen dituen klasea. Klase honi esker, erreserba bat egiteko garaian klase honen instantzia bat parametroz pasatzearekin erreserba egin daiteke. Metodo honetan sarrerako parametroko aldagaiak pila bat erabiltzen zirenez erabaki dugu 5 lerro gastatzea ErreserbaEskaera klasearen aldagaiak gordetzen aldagai lokaletan ondorengo kode lerroak irakurterrazagoak izan daitezen.

Kode errefaktorizatua:

```
public RideRequest erreserbatu(RideRequest rq) {
    Ride ride = rq.getRide();
    String requestFrom = rq.getFromRequested();
    String requestTo = rq.getToRequested();
    int seats = rq.getSeats();
    Traveller traveller = rq.getTraveller();
    Date time = rq.getWhenRequested();

    RideRequest request=null;
    db.getTransaction().begin();
    Ride rd = db.find(Ride.class, ride.getRideNumber());
    if(rd.lortuEserlekuKopMin(requestFrom, requestTo)>=seats) {
    Traveller t = db.find(Traveller.class, traveller.getUser());

    t.kenduDirua(rd.lortuBidaiarenPrezioa(requestFrom, requestTo) * seats);
    request = t.addRequest(time, rd, seats, requestFrom, requestTo);
    rd.addRequest(request);

    t.gehituMezuaTransaction(0, rd.lortuBidaiarenPrezioa(requestFrom, requestTo) * seats, request);

    db.getTransaction().commit();
    }else {
        System.out.println();
        db.getTransaction().commit();
        return null;
    }

    return request;

}
```

- register(Ekaitz Pinedo)

Arazoa:

7 parametroko metodoa.

Hasierako kodea:

```
public Profile register(String email, String name, String surname, String username, String password, String telf,
        String type) {

    Profile u = db.find(Profile.class, username);
    Profile user;
    if (u != null) {
        return null;
    } else {

        if (type.equals("Traveller")) {
            user = new Traveller(email, name, surname, username, password, telf);

        } else {
            user = new Driver(email, name, surname, username, password, telf);
        }
        db.getTransaction().begin();
        db.persist(user);
        db.getTransaction().commit();
        System.out.println("Ondo gorde da");
        return user;
    }

}
```

Konponketa:

Orain bi parametro bakarrik hartzen ditu, profile bat eta mota. Gainera metodo lagungarri bat sortu dut irakurgarritasuna hobetzeko eta “Write short units of code” errespetatzeko.

Kode errefaktorizatua:

```
public Profile register(Profile p,String type) {

    Profile u = db.find(Profile.class, p.getUser());
    Profile user;
    if (u != null) {
        return null;
    } else {

        user=createTravellerOrDriver(p,type);

        db.getTransaction().begin();
        db.persist(user);
        db.getTransaction().commit();
        System.out.println("Ondo gorde da");
        return user;
    }

}
private Profile createTravellerOrDriver(Profile p,String type) {
    String email=p.getEmail();
    String name=p.getName();
    String surname= p.getSurname();
    String username=p.getUser();
    String password=p.getPassword();
    String telf=p.getTelf();

    if (type.equals("Traveller")) {
        return new Traveller(email, name, surname, username, password, telf);

    } else {
        return new Driver(email, name, surname, username, password, telf);
    }

}
```

- DataAccess/gehituErreklamazioa metodoa(Urtzi Etxegarai)

Arazoa:

5 parametroko metodoa da gehituErreklamazioa().

Hasierako kodea:

```
public void gehituErreklamazioa(Profile p, Profile nori, String deskripzioa, float prezioa, RideRequest r)
    db.getTransaction().begin();
    Profile profile = db.find(Profile.class, p.getUser());
    RideRequest request = db.find(RideRequest.class, r.getId());
    if (profile instanceof Traveller) {
        request.setErreklamatuaDriver(true);
    } else {
        request.setErreklamatuaTraveller(true);
    }
    profile.gehituErreklamazioa(nori, deskripzioa, prezioa, r);
    db.getTransaction().commit();
}
```

Konponketa:

Arazoa konpontzeko, 5 parametroak sartu beharrean Erreklamazio klasea sartuko dugu, “Erreklamazio” klase honek erreklamazio bat sortzeko beharrezko informazioa gordetzen du. Erreklamazio klase barruan gordeta daude lehen sartutako 5 parametroak. Hori egitea pentsatu dugu azken finean 5 parametro asko direlako eta kodea zikindu egiten duelako. Gainera baliatu egiten dugu Erreklamazioa klasea, jadanik existitzen zena, beraz kodea txukundu dugu klase berririk sortu gabe.

Kode errefaktorizatua:

```
public void gehituErreklamazioa( Erreklamazioa erreklam) {
    Profile nork= erreklam.getNork();
    RideRequest r= erreklam.getErreserba();
    db.getTransaction().begin();
    Profile profile = db.find(Profile.class, nork.getUser());
    RideRequest request = db.find(RideRequest.class, r.getId());
    if (profile instanceof Traveller) {
        request.setErreklamatuaDriver(true);
    } else {
        request.setErreklamatuaTraveller(true);
    }
    profile.gehituErreklamazioa(erreklam.getNori(), erreklam.getDeskripzioa(), erreklam.getPrezioa(), request);
    db.getTransaction().commit();
}
```

Github-eko link-a

https://github.com/UrtziE/SI2Proiektua