ERREFAKTORIZAZIOA

Proiektua:https://github.com/LurLopez/proiektu

Lur Lopez:

"Keep unit interfaces small"

Kodigo honetan Driver bati kotxe gat gehitzeko, parametro gisa ematen dizkiogu, eta parametro horiekin, dataAccessean bertan sortzen dugu kotxea eta hori datu basean gordetzen dugu. Arazoa da, 5 parametro pasatzen diogula, eta honek nahasmena eta erroreak ekar ditzake

```
public boolean KotxeaGehitu(String Marka,String Modeloa,String Matrikula,int Eserlekuak,Driver d)throws Exception {

    if(!Marka.isEmpty()&&!Modeloa.isEmpty()&&!Matrikula.isEmpty()) {
            Kotxe k=db.find(Kotxe.class, Matrikula);
            if(k==null) {
            Kotxe kotxe=new Kotxe(Eserlekuak,Matrikula,Marka,Modeloa);
            db.getTransaction().begin();
            db.persist(kotxe);
            d.KotxeaGehitu(kotxe);
            db.merge(d);
            Driver dr=db.find(Driver.class, d.getName());
            db.getTransaction().commit();
            return true;
            }
            else {
                throw new Exception(ResourceBundle.getBundle("Etiquetas").getString("Errore.MatrikulaDagoenekoExistitu"));
            }
    }
    else {
        throw new Exception(ResourceBundle.getBundle("Etiquetas").getString("Errore.BalioaFalta"));
    }
}
```

Hau zuzentzeko, zuzenean kotxea parametroa pasatzea erabaki dugu. Kotxe hori GUIean sortuko da. Sortutako kotxe berri horien parametroa null diren ala ez jakiteko, kotxearen eraikitzailean konprobatzea erabaki dugu. Horrela, inoiz ezingo da Kotxe bat null balioekin sortu.

DataAcceseko KotxeaGehitu metodoa:

```
public boolean KotxeaGehitu(Kotxe kotxea,Driver d)throws Exception {
    //String Marka,String Modeloa,String Matrikula,int Eserlekuak,
            Kotxe k=db.find(Kotxe.class, kotxea.getMatrikula());
            if(k==null) {
            db.getTransaction().begin();
            db.persist(kotxea);
            d.KotxeaGehitu(kotxea);
            db.merge(d);
            Driver dr=db.find(Driver.class, d.getName());
            db.getTransaction().commit();
            return true;
            }
            else {
                throw new Exception(ResourceBundle.getBundle("Etiquetas").getString("Errore.MatrikulaDagoenekoExistitu"));
            }
```

Kotxea klasea:

```
public Kotxe(int e,String mat,String mar,String mode) throws Exception {
    if(!mar.isEmpty()&&!mode.isEmpty()&&!mat.isEmpty()) {
    this.eserlekukop=e;
    this.matrikula=mat;
    this.marka=mar;
    this.modeloa=mode;
}
else { throw new Exception(ResourceBundle.getBundle("Etiquetas").getString("Errore.BalioaFalta"));
}

}
```

Write short units of code:

```java
public List<Ride> GetAlertaBatenBidaiak(Alerta a) {

    List<Ride> badago = new ArrayList<Ride>();
    TypedQuery<Ride> query = db.createQuery("SELECT r FROM Ride r", Ride.class);
    List<Ride> results = query.getResultList();
    String nora = a.getNora();
    String noraTxikiz = nora.toLowerCase();
    String nondik = a.getNondik();
    String nondikTxikiz = nondik.toLowerCase();
    String bnora;
    String bnoraTxikiz;
    String bnondik;
    String bnondikTxikiz;
    Date adata = a.getData();
    for(Ride r : results) {
        bnora = r.getTo();
        bnoraTxikiz = bnora.toLowerCase();
        bnondik = r.getFrom();
        bnondikTxikiz = bnondik.toLowerCase();
        if(bnoraTxikiz.equals(noraTxikiz) && bnondikTxikiz.equals(nondikTxikiz) && adata.equals(r.getDate())) {
            badago.add(r);
        }
    }
```

Kasu honetan, oso luzea den metodo bat dago. Bertan, Soberan dauden hainbat lerro daude, eta horrez gain, bi gauza egiten ditu: alerta bateko elementuak hartu eta bere atributuak letra txikira pasatzen ditu. Ondoren, datu baseko alerta guztietan begiratzen da, eta alerta hori dagoeneko existitzen bada, lista batean gehitzen da

```java
public List<Ride> GetAlertaBatenBidaiak(Alerta a) {

    List<Ride> badago = new ArrayList<Ride>();
    TypedQuery<Ride> query = db.createQuery("SELECT r FROM Ride r", Ride.class);
    List<Ride> results = query.getResultList();
    String noraTxikiz = a.getNora().toLowerCase();
    String nondikTxikiz =a.getNondik().toLowerCase();
    Date adata = a.getData();
    return GetAlertaBatenBidaiakFor(noraTxikiz,nondikTxikiz,adata,results);
}

public List<Ride> GetAlertaBatenBidaiakFor(String noraTxikiz,String nondikTxikiz,Date adata,List<Ride> results){
    List<Ride> itzuli = new ArrayList<Ride>();
    for(Ride r : results) {
        String bnoraTxikiz = r.getTo().toLowerCase();
        String bnondikTxikiz = r.getFrom().toLowerCase();
        if(bnoraTxikiz.equals(noraTxikiz) && bnondikTxikiz.equals(nondikTxikiz) && adata.equals(r.getDate())) {
            itzuli.add(r);
        }
    }
    return itzuli;
}
```

Gure soluzioa bi ataletan banatzen da. Alde batetik, lehenengo metodoan datu baseko alerta guztiak eskuratuko ditugu, eta guk parametro gisa pasa diogun alertaren nora eta nondik atributuak letra txikiz gordeko ditugu aldagai batean.

Ondoren, beste metodo batean, guk pasatako atributu horiekin for hedatu bat rekorrituko du, eta alerta bakoitzari nora eta nori atributuak gordeko ditu aldagai batean (hauek ere txikiz egongo dira). Ondoren, pasa diogun alertaren atributuekin konparatuko du, eta berdinak badira lista batean itzuliko du.

Write simple units of code:

Nahiz eta kasu honetan 4 if eta for izan (mugan dago), etorkizunean Driver entitatean gauzak ezabatu nahi baditugu agian beharrezkoa izango da beste bat gehitzea, eta hortaz, orain bi metodoetan banatuko dugu:

```
public void erabiltzaileaEzabatu(User d) {
    db.getTransaction().begin();
    User erabiltzaile = db.find(User.class, d.getErabiltzaileIzena());
    if(erabiltzaile instanceof Traveler) {
        List<Alerta> alertaList = GetTravelerBatenAlertak((Traveler) d);
        for(Alerta a : alertaList) {
            db.remove(a);
        }
        List<Erreserba> erreserbaList = GetTravelerBatenErreserbak((Traveler) d);
        db.getTransaction().commit();
        for(Erreserba e : erreserbaList) {
            ErreserbaEzabatu(e.getId());
        }
        db.getTransaction().begin();
    }



    if (erabiltzaile != null) {
        db.remove(erabiltzaile);
    }
    db.getTransaction().commit();

}
```

Alde batetik, travelerra ezabatu metodoa dago, eta beste alde batetik, erabiltzailea ezabatu. Hortaz, erabiltzaileaEzabatu metodoari deitzen denean, pasatako erabiltzailea travelerra den konprobatuko da. Hala bada, travelerraEzabatu metodoari deituko zaio, eta bere atributo propioak ezabatuko dira datu basetik. Amaitzeko, erabiltzailea datu basetik ezabatuko da.

```
public void erabiltzaileaEzabatu(User d) {
    db.getTransaction().begin();
    User erabiltzaile = db.find(User.class, d.getErabiltzaileIzena());
    db.getTransaction().commit();
    if(erabiltzaile instanceof Traveler) travelerraEzabatu((Traveler) d);

    db.getTransaction().begin();
    if (erabiltzaile != null) {
        db.remove(erabiltzaile);
    }
    db.getTransaction().commit();
}

public void travelerraEzabatu(Traveler t) {
    db.getTransaction().begin();
    List<Alerta> alertaList = GetTravelerBatenAlertak(t);
    for(Alerta a : alertaList) {
        db.remove(a);
    }
    List<Erreserba> erreserbaList = GetTravelerBatenErreserbak(t);
    for(Erreserba e : erreserbaList) {
        ErreserbaEzabatu(e.getId());
    }
    db.getTransaction().commit();
}
```

Duplicate code:

Kasu honetan bi metodo ia antzeko ditugu. Batean Tbaloratua aldagaia aldatzen da, eta bestean Dbaloratua. Bi metodo hauek elkartu egingo ditugu. eta bi aldagaietako balioa aldatu behar duen adierazteko boolean bat pasako diogu atributu gisa.

```
        public void travelerBaloratu(int erreserbakode) {
            db.getTransaction().begin();
            Erreserba e=db.find(Erreserba.class, erreserbakode);
            e.setTbaloratua(true);
            db.getTransaction().commit();
        }

        public void driverBaloratu(int erreserbakode) {
            db.getTransaction().begin();
            Erreserba e=db.find(Erreserba.class, erreserbakode);
            e.setDbaloratua(true);
            db.getTransaction().commit();
        }
```

Zein atributu aldatu nahi dugun jakiteko, nork atributua gehitu diogu.

```
public void userraBaloratu(String nork,int erreserbakode) {
    db.getTransaction().begin();
    Erreserba e=db.find(Erreserba.class, erreserbakode);

    if(nork.equals("traveler")) e.setTbaloratua(true);
    else if(nork.equals("driver")) e.setDbaloratua(true);

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

Ander Untzetabarrenetxea:

"Keep unit interfaces small":

```
public Erreklamazio ErreklamazioaSortu(String gaia,String testua,User nork, User nori, Ride r) {
    ArrayList<Erreklamazio> ErreklamazioakNork=nork.getErreklamazioList();
    ArrayList<Erreklamazio> ErreklamazioakNori=nori.getErreklamazioList();
    Erreklamazio erreklamazio=new Erreklamazio(gaia,testua,nork,nori,r);
    ErreklamazioakNork.add(erreklamazio);
    ErreklamazioakNori.add(erreklamazio);

    db.getTransaction().begin();

    db.persist(erreklamazio);
    nork.setErreklamazioList(ErreklamazioakNork);
    nori.setErreklamazioList(ErreklamazioakNori);

    db.getTransaction().commit();


    return erreklamazio;
}
```

Funtzio honi 5 parametro pasatzen zaizkio, honek nahasmendua ekar dezake.

```
public Erreklamazio ErreklamazioaSortu(Erreklamazio erreklamazioa) {
    User nork = erreklamazioa.getNork();
    User nori = erreklamazioa.getNori();
    ArrayList<Erreklamazio> ErreklamazioakNork=nork.getErreklamazioList();
    ArrayList<Erreklamazio> ErreklamazioakNori=nori.getErreklamazioList();
    ErreklamazioakNork.add(erreklamazioa);
    ErreklamazioakNori.add(erreklamazioa);

    db.getTransaction().begin();

    db.persist(erreklamazioa);
    nork.setErreklamazioList(ErreklamazioakNork);
    nori.setErreklamazioList(ErreklamazioakNori);

    db.getTransaction().commit();


    return erreklamazioa;
}
```

Erreklamazioa jada sortuta pasa diogu eta funtzio honek datu basean gordeko ditu gorde beharreko aldaketak.

Erreklamazioa GUI-an sortuko da:

```
if(!ErreklamazioField.getText().equals("") && !GaiaField.getText().equals("")) {
    facade.setErreklamatuta(erreserba.getId());
    String gaia=GaiaField.getText();
    String testua = ErreklamazioField.getText();
    Erreklamazio erreklamazioa=new Erreklamazio(gaia,testua,nork,nori,r);
    facade.ErreklamazioaSortu(erreklamazioa);
    itxi_lehioa(arg0);
    if(nork instanceof Traveler) {
        Traveler t= (Traveler) nork;
        BidaiaBaieztatuGUI lehioa=new BidaiaBaieztatuGUI(t);
        lehioa.setVisible(true);
    }
    else if(nork instanceof Driver) {
        Driver d=(Driver) nork;
        DriverMenuaGUI lehioa=new DriverMenuaGUI(d);
        lehioa.setVisible(true);
    }


}
else {
    System.out.println("Balio bat falta da");
}
```

"Write short units of code":

```
public void ErreserbaOnartu(int kodeeskaera) {
    db.getTransaction().begin();
    Erreserba erreserba = db.find(Erreserba.class, kodeeskaera);
    if(erreserba!=null) {
        Traveler bidaiari=erreserba.getTraveler();
        int kodebidaia=erreserba.getDriverKodea();
        Float prezioa = db.find(Ride.class, kodebidaia).getPrice();
        List<Erreserba> erreserbak=this.Bidaia_Baten_Erreserbak_Eta_Eskaerak(kodebidaia);
        Erreserba libre_erreserba=this.Bidaia_Baten_Erreserba_Libreak(erreserbak);
        if(libre_erreserba!=null) {
            db.getTransaction().commit();
            Driver gidari=Gidaria_Bilatu(kodebidaia);



            libre_erreserba.setOnartuta(true);
            libre_erreserba.setTraveler(bidaiari);
            db.merge(libre_erreserba);

            int librekop=LibreDaudenEserlekuKopurua(kodebidaia);
            db.getTransaction().begin();
            db.remove(erreserba);
            db.getTransaction().commit();
            db.getTransaction().begin();
            if(librekop==0) {

                List<Erreserba> eskaerak= Bidaia_Baten_Erreserbak_Eta_Eskaerak(kodebidaia);
                for(Erreserba erreserbak_kendu:eskaerak) {
                    if(erreserbak_kendu.getErreserbatuta()==true) {
                    db.getTransaction().commit();
                    EskaeraDeuseztatu(erreserbak_kendu.getId(),0);
                    db.getTransaction().begin();
                }
            }
            }
        }
    }
    else System.out.println("Ez dago eserleku librerik");
    db.getTransaction().commit();
}
```

ErreserbaOnartu metodoa-k kode lerro asko ditu, bertan egindako erreserba bat onartzen da baldin eta eserleku librerik badaude bidaia horretan, horretarako egiaztapen ugari egiten dira. Lehenik eginiko erreserbaren eskaera billatzen da datu basean eta aurkituz gero eskaeraren datuak hartzen dira, gero, erreserba eskaera ezabatzen da eta eserleku librerik badagoen edo ez egiaztatzen da eta posible bada erreserba gordetzen da.

```
public void ErreserbaOnartu(int kodeeskaera) {
    db.getTransaction().begin();
    Erreserba erreserba = db.find(Erreserba.class, kodeeskaera);
    if(erreserba!=null) {
        Traveler bidaiari=erreserba.getTraveler();
        int kodebidaia=erreserba.getDriverKodea();
        Float prezioa = db.find(Ride.class, kodebidaia).getPrice();
        List<Erreserba> erreserbak=this.Bidaia_Baten_Erreserbak_Eta_Eskaerak(kodebidaia);
        Erreserba libre_erreserba=this.Bidaia_Baten_Erreserba_Libreak(erreserbak);
        if(libre_erreserba!=null) {
            erreserbaGorde(libre_erreserba, kodebidaia, bidaiari, erreserba);
        }else {
            System.out.println("Ez dago eserleku librerik");
        }
    }
    else System.out.println("Ez da erreserba aurkitu");
    db.getTransaction().commit();
}

public void erreserbaGorde(Erreserba libre_erreserba,int kodebidaia,Traveler bidaiari,Erreserba erreserba) {
        db.getTransaction().commit();
        Driver gidari=Gidaria_Bilatu(kodebidaia);
        libre_erreserba.setOnartuta(true);
        libre_erreserba.setTraveler(bidaiari);
        db.merge(libre_erreserba);
        int librekop=LibreDaudenEserlekuKopurua(kodebidaia);
        db.getTransaction().begin();
        db.remove(erreserba);
        db.getTransaction().commit();
        db.getTransaction().begin();
        if(librekop==0) {

            List<Erreserba> eskaerak= Bidaia_Baten_Erreserbak_Eta_Eskaerak(kodebidaia);
            for(Erreserba erreserbak_kendu:eskaerak) {
                if(erreserbak_kendu.getErreserbatuta()==true) {
                db.getTransaction().commit();
                EskaeraDeuseztatu(erreserbak_kendu.getId(),0);
                db.getTransaction().begin();
            }
        }
        }
}
```

Funtzioa bi funtziotan banatu dugu, lehen funtzioan eskaera badagoen edo ez konprobatzen da eta erreserbaren datuak lortzen ditugu, erreserba egiteko aukerarik badago erreserbaGorde funtzioari deitzen zaio. Bigarren zatian erreserba egiten da eta eskaera ezabatzen da.

"Write simple units of code":

Jada ez dugu 4-ko konplexutasun zinematikoa duen kode zatirik, ErreserbaOnartu funtzioan egindako aldaketak aplikatuz “bad smell” hau ere konpondu dugu.

"Duplicate code":

Issue hau aurkitzeko gui paketeko BaloratuGUI klasera joan behar izan gara, ez baitugu aurkitu dataAccess-en.

```
    if(nork instanceof Traveler) {
        Traveler t= (Traveler) nork;
        BidaiaBaieztatuGUI lehioa=new BidaiaBaieztatuGUI(t);
        lehioa.setVisible(true);
        itxi_lehioa(arg0);
    }
    else if(nork instanceof Driver) {
        Driver d=(Driver) nork;
        DriverMenuaGUI lehioa=new DriverMenuaGUI(d);
        lehioa.setVisible(true);
        itxi_lehioa(arg0);
    }
```

BaloratuGUI klasean kode zati hau hainbat aldiz agertzen da errepikatuta.

```
private void hurrengoGUI(User nork, ActionEvent arg0) {
    if(nork instanceof Traveler) {
        Traveler t= (Traveler) nork;
        BidaiaBaieztatuGUI lehioa=new BidaiaBaieztatuGUI(t);
        lehioa.setVisible(true);
        itxi_lehioa(arg0);
    }
    else if(nork instanceof Driver) {
        Driver d=(Driver) nork;
        DriverMenuaGUI lehioa=new DriverMenuaGUI(d);
        lehioa.setVisible(true);
        itxi_lehioa(arg0);
    }
}
```

Metodo hau sortu dugu kodea errepikatu beharrean funtzio honi deitzeko eta kode zatia soilik behin idatzia izateko.