eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

ERREFAKTORIZAZIOA

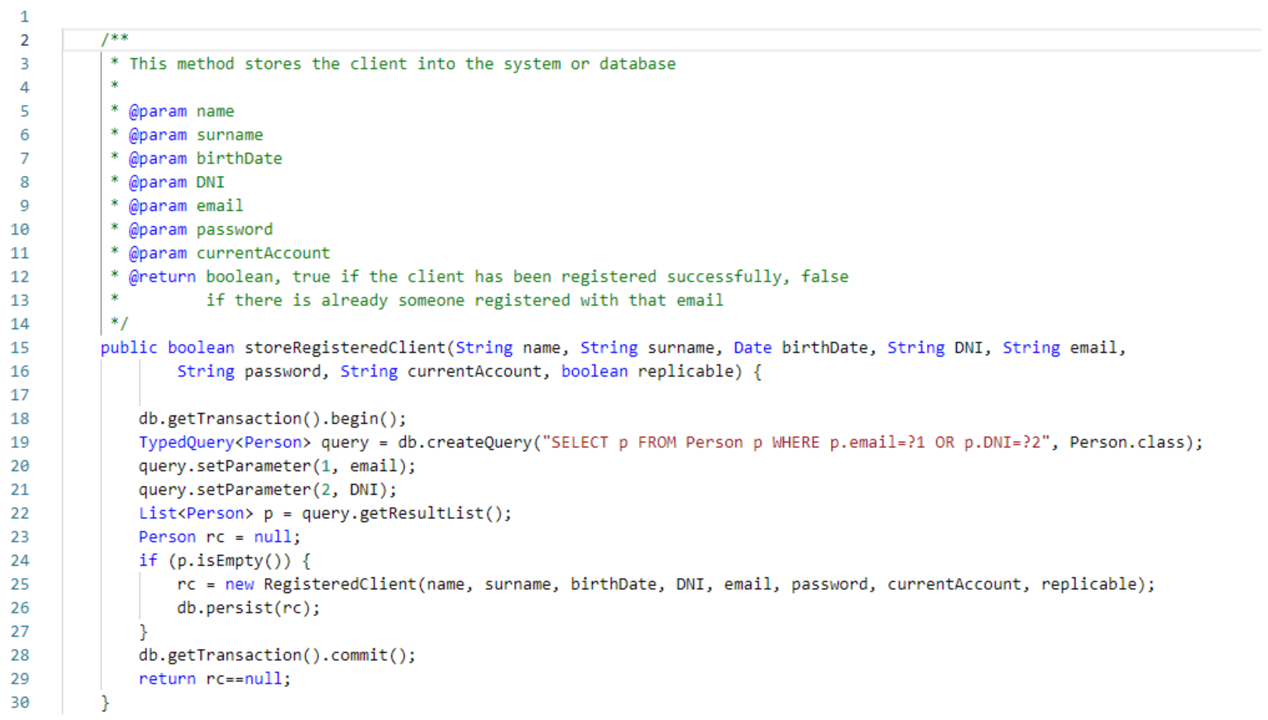
Software Ingeniaritza II

Itsaso Yan Garcia eta Oihane Zaldua

Itsaso:

"Keep unit interfaces small" edo Long paramater List (5. kapituloa).

1. Hasierako kodea:

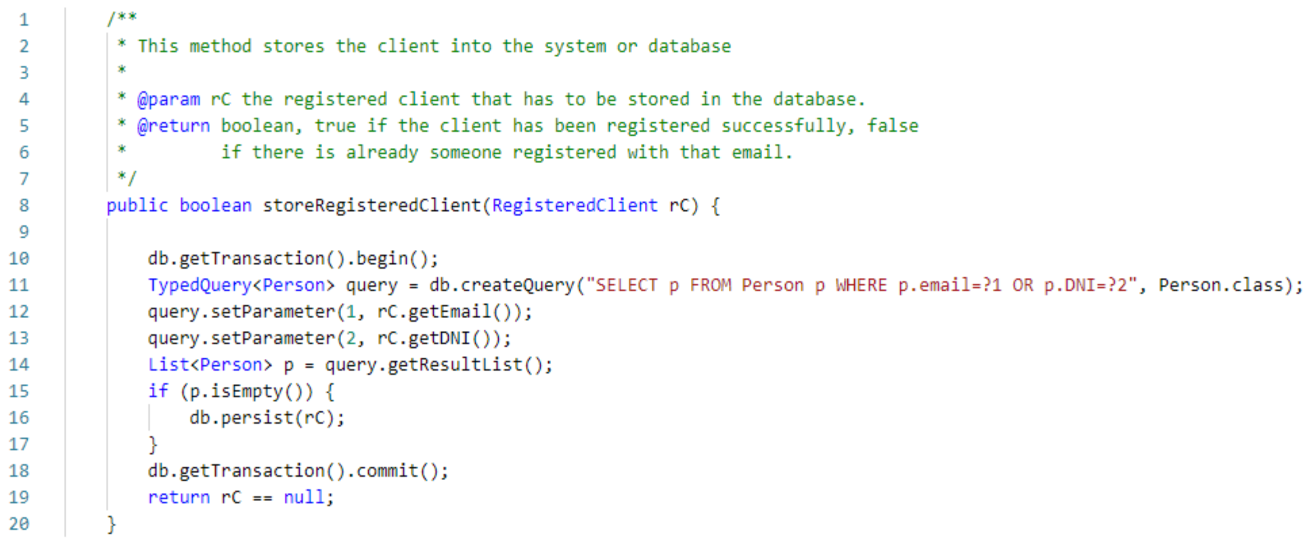


```java

/**
 * This method stores the client into the system or database
 *
 * @param name
 * @param surname
 * @param birthDate
 * @param DNI
 * @param email
 * @param password
 * @param currentAccount
 * @return boolean, true if the client has been registered successfully, false
 *         if there is already someone registered with that email
 */
public boolean storeRegisteredClient(String name, String surname, Date birthDate, String DNI, String email,
        String password, String currentAccount, boolean replicable) {

    db.getTransaction().begin();
    TypedQuery<Person> query = db.createQuery("SELECT p FROM Person p WHERE p.email=?1 OR p.DNI=?2", Person.class);
    query.setParameter(1, email);
    query.setParameter(2, DNI);
    List<Person> p = query.getResultList();
    Person rc = null;
    if (p.isEmpty()) {
        rc = new RegisteredClient(name, surname, birthDate, DNI, email, password, currentAccount, replicable);
        db.persist(rc);
    }
    db.getTransaction().commit();
    return rc==null;
}
```

2. Errefaktorizatuko kodea:



```java
/**
 * This method stores the client into the system or database
 *
 * @param rC the registered client that has to be stored in the database.
 * @return boolean, true if the client has been registered successfully, false
 *         if there is already someone registered with that email.
 */
public boolean storeRegisteredClient(RegisteredClient rC) {

    db.getTransaction().begin();
    TypedQuery<Person> query = db.createQuery("SELECT p FROM Person p WHERE p.email=?1 OR p.DNI=?2", Person.class);
    query.setParameter(1, rC.getEmail());
    query.setParameter(2, rC.getDNI());
    List<Person> p = query.getResultList();
    if (p.isEmpty()) {
        db.persist(rC);
    }
    db.getTransaction().commit();
    return rC == null;
}
```

3. Egindako errefaktorizazioren deskribapena:

`storeRegisteredClient` metodoa errefaktorizatu dut. Errefaktorizazio honen helburua, metodo honek zituen sarrerako parametro kopurua ahalik eta gehien murriztea zen. Izan ere, metodo batek 4 parametro baino gehiago dituenean, kodea ulertzeko eta mantentzeko zailagoa izateaz gain, *bad smell* bilakatzen da. Hortaz, lehen metodo honek zituen 8 parametroak `RegisteredClient` batek dituen datuak zirenez, zuzenean `RegisteredClient` motako objektu bat pasa diot, parametro kopurua minimoa izatea lortuz. Beraz, `new RegisteredClient` egin beharrean sortutako bezero erregistratu hori datu-basean gordetzeko, zuzenean parametroan pasatako bezero erregistratua (`RegisteredClient`) gordetzen da datu-basean (*db.persist(rC))*. Horrela, metodo honen sarrerako parametro kopurua murrizteaz gain, kodea laburragoa izatea lortu dut, mantentzeko eta ulertzeko errazago bilakatuz.

Oihane:

"Write short units of code" edo Long Method (2. kapituloa)

1. Hasierako kodea:

```java
/**
* This method puts the result (the right prediction) of a question
*
* @param prediction
*/
public void putResult(Question question, Prediction prediction) {
db.getTransaction().begin();
Question q = db.find(Question.class, question.getQuestionNumber());
q.setResult(prediction.getPrediction());
Vector<Bet> bets = prediction.getBets();
for (int i = 0; i < bets.size(); i++) { //predikzioko apustu bakoitzeko
Bet b = db.find(Bet.class, bets.get(i).getBetNumber());
RegisteredClient rc = db.find(RegisteredClient.class, b.getClient().getEmail());
float money = b.getMoney(); //apustatutako ditu kantitatea
Vector<Prediction> ps = b.getPredictions();
RegisteredClient replicatedClient=null;
if(b.getReplicatedClient()!=null) {//apustua erreplikatutako bezero baten erreplika bada
replicatedClient = b.getReplicatedClient();
}
if(b.getMultiple()) { //apustu anitza bada
boolean multipleWin = true;
float totalFee=1;
for(int j=0; j<ps.size(); j++) {
if(ps.get(j).getQuestion().getResult()==null || !ps.get(j).getQuestion().getResult().equals(ps.get(j).getPrediction())) multipleWin=f
totalFee=totalFee*ps.get(j).getFee();
}
if (multipleWin) { //apustu anitza irabazi badu
if(replicatedClient!=null) { //apustua erreplikatua bazen, bezero originalari portzentai bat eman
replicatedClient.setBalance(replicatedClient.getBalance() + money*totalFee*(float)(0.1));
rc.setBalance(rc.getBalance() + money*totalFee*(float)(0.9));
rc.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(), rc.getBalance() + money*totalFee*(float)(0.9), new Date(), "x");
replicatedClient.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(), replicatedClient.getBalance() + money*totalFee*(float)(0.1), new Da
"x");
}else{
rc.setBalance(rc.getBalance() + money*totalFee);
rc.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(), rc.getBalance() + money*totalFee, new Date(), "x");
}
b.setWin(true);
}
} else { //apustu sinplea
float new_balance = money * prediction.getFee() * b.getFeeMultiplier();
if(replicatedClient!=null) { //apustua erreplikatua bazen, bezero originalari portzentai bat eman
replicatedClient.setBalance(replicatedClient.getBalance() + new_balance*(float)(0.1));
rc.setBalance(rc.getBalance() + new_balance*(float)(0.9));
rc.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(),rc.getBalance() + new_balance*(float)(0.9), new Date(), "-");
replicatedClient.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(), replicatedClient.getBalance() + money*(float)(0.1), new Date(), "-"
}else{
rc.setBalance(rc.getBalance() + new_balance);
rc.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(),rc.getBalance() + new_balance, new Date(), "-");

}
b.setWin(true);
}
db.persist(rc);
db.persist(b);
}
db.persist(q);
db.getTransaction().commit();
}
```

2. Errefaktorizatuko kodea:

```
/**
* This method puts the result (the right prediction) of a question
*
* @param prediction
*/
public void putResult(Question question, Prediction prediction) {
        db.getTransaction().begin();
        Question q = db.find(Question.class, question.getQuestionNumber());
        q.setResult(prediction.getPrediction());
        Vector<Bet> bets = prediction.getBets();
        for (int i = 0; i < bets.size(); i++) { //predikzioko apustu bakoitzeko
                Bet b = db.find(Bet.class, bets.get(i).getBetNumber());
                RegisteredClient rc = db.find(RegisteredClient.class, b.getClient().getEmail());

                if(b.getMultiple()) { //apustu anitza bada
                        putResultMultipleBet(b, rc);
                } else { //apustu sinplea
                        putResultSimpleBet(prediction, b, rc);
                }
                db.persist(rc);
                db.persist(b);
        }
        db.persist(q);
        db.getTransaction().commit();
}

/**
* This method sets as won the not-multiple bet and calls winMoney so bet winners' gain their money
* @param prediction
* @param b
* @param rc
*/
private void putResultSimpleBet(Prediction prediction, Bet b, RegisteredClient rc) {
        float new_balance = b.getMoney() * prediction.getFee() * b.getFeeMultiplier();
        winMoney(b, new_balance, "-", rc);
        b.setWin(true);
}

/**
 * This method checks if the multiple bet has been won and calls winMoney so bet winners' gain their money
 * @param b
 * @param rc
 */
private void putResultMultipleBet(Bet b, RegisteredClient rc) {
        Vector<Prediction> ps = b.getPredictions();
        boolean multipleWin = true;
        float totalFee=1;
        for(int j=0; j<ps.size(); j++) {
                if(ps.get(j).getQuestion().getResult()==null ||
!ps.get(j).getQuestion().getResult().equals(ps.get(j).getPrediction())) multipleWin=false;
                totalFee=totalFee*ps.get(j).getFee();
        }
        if (multipleWin) { //apustu anitza irabazi badu
                float new_balance = b.getMoney()*totalFee;
                winMoney(b, new_balance, "x", rc);
                b.setWin(true);
        }
}

/**
 * This method gives the clients the corresponded money for winning a bet
 * @param b
 * @param new_balance
 * @param multipleorsimple
 * @param rc
 */
private void winMoney(Bet b, float new_balance, String multipleorsimple, RegisteredClient rc){
        RegisteredClient replicatedClient = b.getReplicatedClient();
        if(replicatedClient!=null) {//apustua erreplikatua bada, bezero originalari portzentai bat eman
                replicatedClient.setBalance(replicatedClient.getBalance() + new_balance*(float)(0.1));
                rc.setBalance(rc.getBalance() + new_balance*(float)(0.9));
                rc.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(), rc.getBalance() + new_balance*(float)(0.9), new
Date(), multipleorsimple);
                replicatedClient.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(), replicatedClient.getBalance() +
new_balance*(float)(0.1), new Date(), multipleorsimple);
        }else {
                rc.setBalance(rc.getBalance() + new_balance);
                rc.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(),rc.getBalance() + new_balance, new Date(),
multipleorsimple);
        }
}
```

3. Egindako errefaktorizazioren deskribapena:

Zergatik egin dugu errefaktorizazioa?

`putResult` metodoak 53 lerro zituen hasieran, eta 26ko konplexutasun ziklomatikoa. Honek, kodea beste programatzaileek ulertzea zailtzen du, baita etorkizunean kodea aldatzea edo testeatzea ere. Honengatik errefaktorizatu dugu, 15 lerro inguru edo gutxiagoko kodeetara, kode nagusia metodo gehiagotan banatuz. Honek ez du, zentzu batean, konplexutasun ziklomatikoa gutxitzen, baina konplexutasun ziklomatiko oso handia izatearen desabantailak modu batean arintzen ditu, mantenua errazten eta kodea testeatzeko eta ulertzeko errazagoa eginez.

Nola planteatu dugu?

Refactor>Extract Method erabili daiteke errefaktorizazioa modu errazean egiteko, `putResult`-eko baldintzetako kodeak metodo desberdinetan banatuz. Hala ere, kodea nahiko errepikakorra da, eta metodo berriek 4 parametro baino gehiago (Bad Smell dena), 15 lerro baino gehiago edo kode errepikakorra izango lukete. Horregatik, nire kabuz metodo errepikakor horiek elkartzea erabaki dut eta, errefaktorizatzean, Bad Smell berriak sahiestu.

Egindako errefaktorizazaioa:

Hasteko, Extract Method egin dut 23-31 lerroetan, eta hau moldatu dut 36-43 lerroetarako ere erabilgarri izateko. Honela, `winMoney` metodoa sortu dut, result/emaitza bat jartzerakon bezeroei dagokien dirua ordaintzeko.

Jarraian, kodea aztertu eta 11-14 lerroak ezbeharrezkoak zirela ikusi dut. Hauek lerro bakar batean idatzi ahal ziren: *RegisteredClient replicatedClient=b.getReplicatedClient()*. Honela kodea laburtu eta ulerterrazagoa da. Gainera, *replicatedClient* hori parametro bezala pasatzen zen `winMoney` metodo berrira, baina 4 parametro baino gehiago izanda Bad Smell berria sortu eta, horregatik, `winMoney`-n sartu dut lerro berri hori, parametro bat aurreztuz. (Aldaketa hauen ondorioz geratutako kodea Eranskina I-n ikus daiteke)

`putResult` metodoa oraindik luzeegia zen. Ondorioz, Refactor>Extract Method erabili dut *if(b.getMultiple())* baldintzaren barneko kodea eta *else*-aren kodea bi metodo berrietan izateko. Honek ere kodea ulerterrazagoa egin duela ikus daiteke ("Errefaktorizatutako kodea" atalean).

Eranskina I:

```
/**
* This method puts the result (the right prediction) of a question
*
* @param prediction
*/
public void putResult(Question question, Prediction prediction) {
     db.getTransaction().begin();
     Question q = db.find(Question.class, question.getQuestionNumber());
     q.setResult(prediction.getPrediction());
     Vector<Bet> bets = prediction.getBets();
     for (int i = 0; i < bets.size(); i++) { //predikzioko apustu bakoitzeko
     Bet b = db.find(Bet.class, bets.get(i).getBetNumber());
     RegisteredClient rc = db.find(RegisteredClient.class, b.getClient().getEmail());
     float money = b.getMoney(); //apustatutako ditu kantitatea
     Vector<Prediction> ps = b.getPredictions();
     RegisteredClient replicatedClient=b.getReplicatedClient();

     if(b.getMultiple()) { //apustu anitza bada
     boolean multipleWin = true;
     float totalFee=1;
     for(int j=0; j<ps.size(); j++) {
     if(ps.get(j).getQuestion().getResult()==null      ||      !ps.get(j).getQuestion().getResult().equals(ps.get(j).getPrediction()))
multipleWin=false;
     totalFee=totalFee*ps.get(j).getFee();
     }
     if (multipleWin) { //apustu anitza irabazi badu
     float new_balance = money*totalFee;
     winMoney(b, new_balance, "x", rc);
     b.setWin(true);
     }
     } else { //apustu sinplea
     float new_balance = money * prediction.getFee() * b.getFeeMultiplier();
     winMoney(b, new_balance, "-", rc);
     b.setWin(true);
     }
     db.persist(rc);
     db.persist(b);
     }
     db.persist(q);
     db.getTransaction().commit();
}

/**
* This method gives the clients the corresponded money for winning a bet
* @param b
* @param new_balance
* @param multipleorsimple
* @param rc
*/
public void winMoney(Bet b, float new_balance, String multipleorsimple, RegisteredClient rc){
     RegisteredClient replicatedClient = b.getReplicatedClient();
     if(replicatedClient!=null) {//apustua erreplikatua bada, bezero originalari portzentai bat eman
     replicatedClient.setBalance(replicatedClient.getBalance() + new_balance*(float)(0.1));
     rc.setBalance(rc.getBalance() + new_balance*(float)(0.9));
     rc.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(),  rc.getBalance()  +  new_balance*(float)(0.9),  new  Date(),
multipleorsimple);
     replicatedClient.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(), replicatedClient.getBalance() + new_balance*(float)(0.1), new
Date(), multipleorsimple);
     }else {
     rc.setBalance(rc.getBalance() + new_balance);
     rc.addMovements2(ResourceBundle.getBundle("Etiquetas").getString("BetWon")+":
"+b.getPredictions().get(0).getQuestion().getEvent().getDescription(),rc.getBalance() + new_balance, new Date(), multipleorsimple);
     }
}
```