Lab: errefaktorizazioa eclipsen

Josu Aguinaga Bengoetxea[1]

E-mail: [1]aginagajosu@gmail.com

2023.eko urriaren 22

Gaien Aurkibidea

Kapitulua 1

Sarrera

Kapituluaren edukia

Laboratorio honetan egindako errefaktorizazioak Josu Aguinaga Bengoetxea-ek egin ditu, eta git-en bitartez errepositorio lokalera igo dira.

GitHub errepositorioa: https://github.com/Josu-A/Lab-Errefaktorizazioa

Kapitulua 2

Errefaktorizazioak

Kapituluaren edukia

ATALA 1 Write short units of code

AZPIATALA 1 Hasierako kodea

```
1  public void open(boolean initializeMode) {
2
3      System.out.println("Opening DataAccess instance =>
       ↪  isDatabaseLocal: " + c.isDatabaseLocal() + "
       ↪  getDatabBaseOpenMode: " +
       ↪  c.getDataBaseOpenMode());
4
5      String fileName = c.getDbFilename();
6      if (initializeMode) {
7          fileName = fileName + ";drop";
8          System.out.println("Deleting the DataBase");
9      }
10
11     if (c.isDatabaseLocal()) {
12         emf = Persistence.createEntityManagerFactory("o
           ↪  bjectdb:" +
           ↪  fileName);
13         db = emf.createEntityManager();
14     } else {
15         Map<String, String> properties = new
           ↪  HashMap<String, String>();
16         properties.put("javax.persistence.jdbc.user",
           ↪  c.getUser());
17         properties.put("javax.persistence.jdbc.password
           ↪  ",
           ↪  c.getPassword());
18
19         emf = Persistence.createEntityManagerFactory("o
           ↪  bjectdb://" + c.getDatabaseNode() + ":" +
           ↪  c.getDatabasePort() + "/" + fileName,
           ↪  properties);
20
21         db = emf.createEntityManager();
22     }
23 }
```

AZPIATALA 2 **Errefaktorizatutako kodea**

```
1  public void open(boolean initializeMode) {
2      System.out.println("Opening DataAccess instance =>
       ↪  isDatabaseLocal: " + c.isDatabaseLocal()+ "
       ↪  getDatabBaseOpenMode: " +
       ↪  c.getDataBaseOpenMode());
```

```
3
4       String fileName =
         ↪  getDBInitializationName(initializeMode);
5
6       if (c.isDatabaseLocal()) {
7           emf = Persistence.createEntityManagerFactory("o
             ↪  bjectdb:" +
             ↪  fileName);
8           db = emf.createEntityManager();
9       } else {
10          Map<String, String> properties = new
             ↪  HashMap<String, String>();
11          properties.put("javax.persistence.jdbc.user",
             ↪  c.getUser());
12          properties.put("javax.persistence.jdbc.password
             ↪  ",
             ↪  c.getPassword());
13
14          emf = Persistence.createEntityManagerFactory("o
             ↪  bjectdb://" + c.getDatabaseNode() + ":" +
             ↪  c.getDatabasePort() + "/" + fileName,
             ↪  properties);
15
16          db = emf.createEntityManager();
17      }
18  }
19
20  public String getDBInitializationName(boolean
     ↪  initializeMode) {
21      String fileName = c.getDbFilename();
22      if (initializeMode) {
23          fileName = fileName + ";drop";
24          System.out.println("Deleting the DataBase");
25      }
26      return fileName;
27  }
```

AZPIATALA 3 **Deskribapena**

`open` metodoan, datu basean ireki baino gehiago egiten da. Baita ere, datu basea ezabatu behar al den ireki baino lehenago aztertzen du. Beraz, kode zati hori, `getDBInitializationName` metodora atera dezakegu, kode lerroak gutxitzeko.

ATALA 2 Write simple units of code

AZPIATALA 1 Hasierako kodea

```
1  public boolean returnMoney(User user, Event event) {
2      boolean ok = false;
3      double dirua = 0;
4      try {
5          User u = db.find(User.class, user);
6          db.getTransaction().begin();
7          for (Mugimendua m : u.getMugimenduak()) {
8              if (m.getGertaera() != null) {
9                  if (m.getGertaera().getEventNumber().eq
                       uals(event.getEventNumber()))
                       {
10                     dirua = m.getDiruKop();
11                 }
12             }
13         }
14         u.setDirua(u.getDirua() + dirua);
15         db.getTransaction().commit();
16         ok = true;
17     } catch (Exception e) {
18         e.printStackTrace();
19     }
20     return ok;
21 }
```

AZPIATALA 2 Errefaktorizatutako kodea

```
1  public boolean returnMoney(User user, Event event) {
2      boolean ok = false;
3      double dirua = 0;
4      try {
5          User u = db.find(User.class, user);
6          db.getTransaction().begin();
7          dirua = getBetMoney(u, event);
8          u.setDirua(u.getDirua() + dirua);
9          db.getTransaction().commit();
10         ok = true;
```

```
11      } catch (Exception e) {
12          e.printStackTrace();
13      }
14      return ok;
15  }
16
17  public double getBetMoney(User user, Event event) {
18      for (Mugimendua m : user.getMugimenduak()) {
19          if (m.getGertaera() != null) {
20              if (m.getGertaera().getEventNumber().equals
                ↪  (event.getEventNumber()))
                ↪  {
21                  return m.getDiruKop();
22              }
23          }
24      }
25      return 0;
26  }
```

AZPIATALA 3 Deskribapena

Metodoaren hasierako konplexutasun ziklomatikoa 5-ekoa da. Hau konpontzeko, erabiltzaileari itzuli behar zaion dirua lortzen duen kode zatia metodo berri batean sartu dezakegu, eta metodo honi deitu hasierako funtzioan. Honela, hasierako metodoaren konplexutasun ziklomatikoa 2 izango da, eta metodo berriarena 4.

ATALA 3 Duplicate code

AZPIATALA 1 Hasierako kodea

```
1  User user1 = new User("user", "izena", "abizena",
    ↪  "user", 26, "asd@gmail.com", 0, lista);
2  Admin user2 = new Admin("admin", "izena", "abizena",
    ↪  "admin", 26, "asd@gmail.com");
3  Langile user3 = new Langile("langile", "izena",
    ↪  "abizena", "langile", 26, "asd@gmail.com");
```

AZPIATALA 2 Errefaktorizatutako kodea

```
String defaultEmail = "asd@gmail.com";

User user1 = new User("user", "izena", "abizena",
 ↪  "user", 26, defaultEmail, 0, lista);
Admin user2 = new Admin("admin", "izena", "abizena",
 ↪  "admin", 26, defaultEmail);
Langile user3 = new Langile("langile", "izena",
 ↪  "abizena", "langile", 26, defaultEmail);
```

AZPIATALA 3 Deskribapena

Default emaila hiru aldiz definitu ordez, defaultEmail aldagai bat sortu dezakegu emailaren balioarekin. Gero, balio aldagai hau berrerabili dezakegu defaul email bat nahi dugun bakoitzean.

ATALA 4 Keep unit interfaces small

AZPIATALA 1 Hasierako kodea

```
public Question createQuestion(Event event, String
 ↪  question, float betMinimum) throws
 ↪  QuestionAlreadyExist {
    System.out.println(">> DataAccess: createQuestion=>
     ↪  event= " + event + " question= " + question + "
     ↪  betMinimum="
        + betMinimum);

    Event ev = db.find(Event.class,
     ↪  event.getEventNumber());

    if (ev.DoesQuestionExists(question))
        throw new QuestionAlreadyExist(ResourceBundle.g
         ↪  etBundle("Etiquetas").getString("ErrorQuery
         ↪  AlreadyExist"));

    db.getTransaction().begin();
```

```
11      Question q = ev.addQuestion(question, betMinimum);
12      db.persist(ev); // db.persist(q) not required when
         ↪  CascadeType.PERSIST is added in questions
13      db.getTransaction().commit();
14      return q;
15  }
```

AZPIATALA 2 Errefaktorizatutako kodea

```
1   public Question createQuestion(Event event, QuestionBet
     ↪  questionBet) throws QuestionAlreadyExist {
2       System.out.println(">> DataAccess: createQuestion=>
         ↪  event= " + event + " question= " +
         ↪  questionBet.getQuestion() + " betMinimum="
3           + questionBet.getBetMinimum());
4
5       Event ev = db.find(Event.class,
         ↪  event.getEventNumber());
6
7       if (ev.DoesQuestionExists(questionBet.getQuestion()
         ↪  ))
8           throw new QuestionAlreadyExist(ResourceBundle.g
             ↪  etBundle("Etiquetas").getString("ErrorQuery
             ↪  AlreadyExist"));
9
10      db.getTransaction().begin();
11      Question q =
         ↪  ev.addQuestion(questionBet.getQuestion(),
         ↪  questionBet.getBetMinimum());
12      db.persist(ev);
13      db.getTransaction().commit();
14      return q;
15  }
```

QuestionBet objektu berria.

```
1   package dataAccess;
2
3   public class QuestionBet {
4       private String question;
```

```
5
6       private float betMinimum;
7
8       public QuestionBet(String question, float
         ↪  betMinimum) {
9           this.question = question;
10          this.betMinimum = betMinimum;
11      }
12
13      public String getQuestion() {
14          return this.question;
15      }
16
17      public void setQuestion(String question) {
18          this.question = question;
19      }
20
21      public float getBetMinimum() {
22          return this.betMinimum;
23      }
24
25      public void setBetMinimum(float betMinimum) {
26          this.betMinimum = betMinimum;
27      }
28  }
```

AZPIATALA 3 **Deskribapena**

Parametro kopurua gutxitzeko, hainbat parametro objektu batean gordetzen ditugu eta objektu berri hori parametro gisa erabiltzen dugu. Errefaktorizazioan, hori egiteaz gain, metodoa erabiltzen den leku guztietan parametroak `QuestionBet` batean gordeta pasa beharko diogu (`BLFacadeImplementation` eta testak).