

SOFTWARE INGENIARITZA 2

**PROIEKTUAREN BIGARREN FASEA:
ERREFAKTORIZAZIOA**

Ander Rubio
Ainhitze Ituarte
Aimar Mancisidor

Aurkibidea

"Write short units of code" motako errefaktorizazioak	3 - 7
"Write simple units of code" motako errefaktorizazioak	8 - 12
“Duplicate code” motako errefaktorizazioak	13 - 15
"Keep unit interfaces small" motako errefaktorizazioak	16 - 23

"Write short units of code" motako errefaktorizazioak

- *gertaeraEzabatu* metodoa (34 lerro (soilik ”” duten lerroak kontatu gabe))

1. Hasierako kodea

```
1136@ public boolean gertaeraEzabatu(Event ev) {  
1137    Event event = db.find(Event.class, ev);  
1138    boolean resultB = true;  
1139    List<Question> listQ = event.getQuestions();  
1140  
1141    for(Question q : listQ) {  
1142        if(q.getResult() == null) {  
1143            resultB = false;  
1144        }  
1145    }  
1146    if(resultB == false) {  
1147        return false;  
1148    }else if(new Date().compareTo(event.getEventDate())<0) {  
1149        TypedQuery<Quote> Qquery = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEventNumber() =?1", Quote.class);  
1150        Qquery.setParameter(1, event.getEventNumber());  
1151        List<Quote> listQUO = Qquery.getResultList();  
1152        for(int j=0; j<listQUO.size(); j++) {  
1153            Quote quo = db.find(Quote.class, listQUO.get(j));  
1154            for(int i=0; i<quo.getApustuak().size(); i++) {  
1155                ApustuaAnitza apustuaAnitza = quo.getApustuak().get(i).getApustuaAnitza();  
1156                ApustuaAnitza api = db.find(ApustuaAnitza.class, apustuaAnitza.getApustuaAnitzaNumber());  
1157                db.beginTransaction().begin();  
1158                api.removeApustua(quo.getApustuak().get(i));  
1159                db.getTransaction().commit();  
1160                if(api.getApustuak().isEmpty() && !api.getEgoera().equals("galdua")) {  
1161                    this.apustuaEzabatu(api.getUser(), api);  
1162                }else if(!api.getApustuak().isEmpty() && api.irabazitaMarkatu()){  
1163                    this.apustuaIrabazi(api);  
1164                }  
1165                db.beginTransaction().begin();  
1166                Sport spo = quo.getQuestion().getEvent().getSport();  
1167                spo.setApustukantitatea(spo.getApustukantitatea()-1);  
1168                Kirolestatistikak ke=api.getUser().kirolestatistikakLortu(spo);  
1169                ke.setKont(ke.getKont()-1);  
1170                db.beginTransaction().commit();  
1171            }  
1172        }  
1173    }  
1174    db.beginTransaction().begin();  
1175    db.remove(event);  
1176    db.getTransaction().commit();  
1177    return true;  
1178}  
1179 }
```

2. Errefaktorizatutako kodea

```
1136@ public boolean gertaeraEzabatu(Event ev) {  
1137    Event event = db.find(Event.class, ev);  
1138    boolean resultB = gertaeraEzabatuLag3(event);  
1139    if(resultB == false) {  
1140        return false;  
1141    }else if(new Date().compareTo(event.getEventDate())<0) {  
1142        TypedQuery<Quote> Qquery = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEventNumber() =?1", Quote.class);  
1143        Qquery.setParameter(1, event.getEventNumber());  
1144        List<Quote> listQUO = Qquery.getResultList();  
1145        for(int j=0; j<listQUO.size(); j++) {  
1146            Quote quo = db.find(Quote.class, listQUO.get(j));  
1147            gertaeraEzabatuLag2(quo);  
1148        }  
1149    }  
1150    db.beginTransaction().begin();  
1151    db.remove(event);  
1152    db.getTransaction().commit();  
1153    return true;  
1154}  
1155 }
```

3. Egindako errefaktorizazioaren azalpena

Errefaktorizazio hau egiteko, metodoaren hainbat atal metodo ezberdin bihurtu ditut. Konkretuki 3 metodo berri sortu ditut, *gertaeraEzabatuLag1*, *2* eta *3*. Hauek dira 3 metodo horien kodeak:

```
1157@ private boolean gertaeraEzabatuLag3(Event event) {
1158     boolean resultB = true;
1159     List<Question> listQ = event.getQuestions();
1160     for(Question q : listQ) {
1161         if(q.getResult() == null) {
1162             resultB = false;
1163         }
1164     }
1165     return resultB;
1166 }
1167
1168@ private void gertaeraEzabatuLag2(Quote quo) {
1169     for(int i=0; i<quo.getApustuak().size(); i++) {
1170         ApustuAnitza apustuAnitza = quo.getApustuak().get(i).getApustuAnitza();
1171         ApustuAnitza ap1 = db.find(ApustuAnitza.class, apustuAnitza.getApustuAnitzaNumber());
1172         db.beginTransaction().begin();
1173         ap1.removeApustua(quo.getApustuak().get(i));
1174         db.getTransaction().commit();
1175         gertaeraEzabatuLag1(ap1);
1176         db.beginTransaction();
1177         Sport spo = quo.getQuestion().getEvent().getSport();
1178         spo.setApustukantitatea(spo.getApustukantitatea()-1);
1179         KirolEstatistikak ke=ap1.getUser().kirolEstatistikakLortu(spo);
1180         ke.setKont(ke.getKont()-1);
1181         db.getTransaction().commit();
1182     }
1183 }
1184
1185@ private void gertaeraEzabatuLag1(ApustuAnitza ap1) {
1186     if(ap1.getApustuak().isEmpty() && !ap1.getEgoera().equals("galduak")) {
1187         this.apustuaEzabatu(ap1.getUser(), ap1);
1188     }else if(!ap1.getApustuak().isEmpty() && ap1.irabazitaMarkatu()){
1189         this.ApustuaIrabazi(ap1);
1190     }
1191 }
```

Ikusten denez, metodo horietan baldintza/begiztak sartu ditut, horrela, lerro kopurua gutxitzeaz gain (puntu honen helburua zena, eta jada lortuta dago) zuzendu beharreko bigarren *code smell* mota ("Write simple units of code") zuzentzeko aprobetxatu dut, *gertaeraEzabatu* metodoaren konplexutasun ziklomatikoa 9 izatetik 4 izatera pasa baita; 1, 2 eta 3 metodo laguntzaileak 9 horietatik 2, 1 eta 2 barneratu baitituzte, hurrenez hurren.

4. Egilea: Aimar Mancisidor

- *emaitzakIpini* metodoa (23 lerro (soilik “}” duten lerroak kontatu gabe))

1. Hasierako kodea

```

11010 public void EmaitzakIpini(Quote quote) throws EventNotFinished{
1102
1103     Quote q = db.find(Quote.class, quote);
1104     String result = q.getForecast();
1105
1106     if(new Date().compareTo(q.getQuestion().getEvent().getEventDate())<0)
1107         throw new EventNotFinished();
1108
1109     Vector<Apustua> listApustuak = q.getApustuak();
1110     db.beginTransaction().begin();
1111     Question que = q.getQuestion();
1112     Question question = db.find(Question.class, que);
1113     question.setResult(result);
1114     for(Quote quo: question.getQuotes()) {
1115         for(Apustua apu: quo.getApustuak()) {
1116
1117             Boolean b=apu.galdutaMarkatu(quo);
1118             if(b) {
1119                 apu.getApustuAnitza().setEgoera("galduta");
1120             }else {
1121                 apu.setEgoera("irabazita");
1122             }
1123         }
1124     }
1125     db.getTransaction().commit();
1126     for(Apustua a : listApustuak) {
1127         db.beginTransaction().begin();
1128         Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
1129         db.getTransaction().commit();
1130         if(bool) {
1131             this.ApustuaIrabazi(a.getApustuAnitza());
1132         }
1133     }
1134 }
```

2. Errefaktorizatutako kodea

```

11010 public void EmaitzakIpini(Quote quote) throws EventNotFinished{
1102
1103     Quote q = db.find(Quote.class, quote);
1104     String result = q.getForecast();
1105
1106     if(new Date().compareTo(q.getQuestion().getEvent().getEventDate())<0)
1107         throw new EventNotFinished();
1108
1109     Vector<Apustua> listApustuak = q.getApustuak();
1110     db.beginTransaction().begin();
1111     Question que = q.getQuestion();
1112     Question question = db.find(Question.class, que);
1113     question.setResult(result);
1114     for(Quote quo: question.getQuotes()) { emaitzakIpiniLag1(quo); }
1115     db.beginTransaction().commit();
1116     for(Apustua a : listApustuak) { emaitzakIpiniLag2(a); }
1117 }
```

3. Egindako errefaktorizazioaren azalpena

Errefaktorizazio hau egin ahal izateko, aztertutako metodoaren hainbat atal beste metodo desberdinan banatu ditut. Zehazki 2 metodo berri sortu ditut, *emaitzakIpiniLag1* eta *emaitzakIpiniLag2* direnak, hasierako kodea 12 lerrorekin utziz. Honakoak dira 2 metodo horien kodeak:

```
1118✉ public void emaitzakIpiniLag1(Quote quo) {  
1119    for(Apustua apu: quo.getApustuak()) {  
1120        Boolean b=apu.galdutaMarkatu(quo);  
1121        if(b) {  
1122            apu.getApustuAnitza().setEgoera("galduta");  
1123        }else {  
1124            apu.setEgoera("irabazita");  
1125        }  
1126    }  
1127}  
1128✉ public void emaitzakIpiniLag2(Apustua a) {  
1129    db.beginTransaction().begin();  
1130    Boolean bool=a.getApustuAnitza().irabazitaMarkatu();  
1131    db.getTransaction().commit();  
1132    if(bool) {  
1133        this.ApustuaIrabazi(a.getApustuAnitza());  
1134    }  
1135}  
1136
```

Ikusi daitekeen bezala, metodo hauetan hasierako metodoko baldintza/begiztak sartu ditut. Modu honetan, lerro kopurua gutxitzeaz gain (puntu honen helburua zena, eta jada lortuta dago), bigarren *code smell* mota ("Write simple units of code") zuzentzea ere lortu dut. *EmaitzakIpini* metodoaren konplexutasun ziklomatikoa 8 izatetik 4 izatera pasa baita. 2 laguntzaileei esker, 9 horiek banatu eta laguntzaileek 3 eta 1 barneratu dituzte.

4. Egilea: Ainhitze Ituarte

- *gertaerakSortu* metodoa (22 lerro (soilik “}” duten lerroak kontatu gabe))

1. Hasierako kodea

```

882 public boolean gertaerakSortu(String description, Date eventDate, String sport) {
883     boolean b = true;
884     db.beginTransaction().begin();
885     Sport spo = db.find(Sport.class, sport);
886     if(spo!=null) {
887         TypedQuery<Event> Equery = db.createQuery("SELECT e FROM Event e WHERE e.getEventDate() =>1 ",Event.class);
888         Equery.setParameter(1, eventDate);
889         for(Event ev: Equery.getResultList()) {
890             if(ev.getDescription().equals(description)) {
891                 b = false;
892             }
893         }
894         if(b) {
895             String[] taldeak = description.split("-");
896             Team lokala = new Team(taldeak[0]);
897             Team kanpokoa = new Team(taldeak[1]);
898             Event e = new Event(description, eventDate, lokala, kanpokoa);
899             e.setSport(spo);
900             spo.addEvent(e);
901             db.persist(e);
902         }
903     }else {
904         db.beginTransaction().commit();
905         return false;
906     }
907     db.beginTransaction().commit();
908     return b;
909 }
```

2. Errefaktorizatutako kodea

```

881 public boolean gertaerakSortu(String description, Date eventDate, String sport) {
882     boolean b = true;
883     db.beginTransaction().begin();
884     b = gertaerakSortuLag1(description, eventDate, sport, b);
885     db.beginTransaction().commit();
886     return b;
887 }
```

3. Egindako errefaktorizazioaren azalpena

Errefaktorizazio hau burutzeko hiru azpimethodo laguntzaile sortu ditut: *gertaeraBadago*, *gertaerakSortuLag1* eta *gertaerakSortuLag2*. Hauetan metodo nagusiaren begizta eta azpibegiztak sartu ditut, honekin metodo nagusia 22 lerro izatetik 5 lerro izatera pasatzea lortuz. Hona hemen sortutako hiru azpimethodo berriak:

```

889 public boolean gertaeraBadago(String description, Date eventDate) {
890     TypedQuery<Event> Equery = db.createQuery("SELECT e FROM Event e WHERE e.getEventDate() =>1 ",Event.class);
891     Equery.setParameter(1, eventDate);
892     for(Event ev: Equery.getResultList()) {
893         if(ev.getDescription().equals(description)) {
894             return false;
895         }
896     }
897     return true;
898 }
899
900 public boolean gertaerakSortuLag1(String description, Date eventDate, String sport, boolean b) {
901     Sport spo = db.find(Sport.class, sport);
902     if(spo!=null) {
903         b = gertaeraBadago(description, eventDate);
904         if(b) {
905             gertaerakSortuLag2(description, eventDate, spo);
906         }
907     }else {
908         b = false;
909     }
910     return b;
911 }
912
913 public void gertaerakSortuLag2(String description, Date eventDate, Sport spo) {
914     String[] taldeak = description.split("-");
915     Team lokala = new Team(taldeak[0]);
916     Team kanpokoa = new Team(taldeak[1]);
917     Event e = new Event(description, eventDate, lokala, kanpokoa);
918     e.setSport(spo);
919     spo.addEvent(e);
920     db.persist(e);
921 }
```

4. Egilea: Ander Rubio

"Write simple units of code" motako errefaktorizazioak

- *gertaeraEzabatu* metodoa (Konplexutasun ziklomatikoa: 9)

1. Hasierako kodea

```
1136@ public boolean gertaeraEzabatu(Event ev) {  
1137     Event event = db.find(Event.class, ev);  
1138     boolean resultB = true;  
1139     List<Question> listQ = event.getQuestions();  
1140  
1141     for(Question q : listQ) {  
1142         if(q.getResult() == null) {  
1143             resultB = false;  
1144         }  
1145     }  
1146     if(resultB == false) {  
1147         return false;  
1148     }else if(new Date().compareTo(event.getEventDate())<0) {  
1149         TypedQuery<Quote> Qquery = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEventNumber() =?1", Quote.class);  
1150         Qquery.setParameter(1, event.getEventNumber());  
1151         List<Quote> listQUO = Qquery.getResultList();  
1152         for(int j=0; j<listQUO.size(); j++) {  
1153             Quote quo = db.find(Quote.class, listQUO.get(j));  
1154             for(int i=0; i<quo.getApustuak().size(); i++) {  
1155                 ApustuaAnitza apustuAnitza = quo.getApustuak().get(i).getApustuaAnitza();  
1156                 ApustuaAnitza api = db.find(ApustuaAnitza.class, apustuAnitza.getApustuaAnitzaNumber());  
1157                 db.beginTransaction().begin();  
1158                 api.removeApustua(quo.getApustuak().get(i));  
1159                 db.getTransaction().commit();  
1160                 if(api.getApustuak().isEmpty() && !api.getEgoera().equals("galdua")) {  
1161                     this.apustuaEzabatu(api.getUser(), api);  
1162                 }else if(!api.getApustuak().isEmpty() && api.irabazitaMarkatu()){  
1163                     this.apustuaIrabazi(api);  
1164                 }  
1165                 db.beginTransaction().begin();  
1166                 Sport spo = quo.getQuestion().getEvent().getSport();  
1167                 spo.setApustukantitatea(spo.getApustukantitatea()-1);  
1168                 Kirolestatistikak ke=api.getUser().kirolEstatistikakLortu(spo);  
1169                 ke.setKont(ke.getKont()-1);  
1170                 db.beginTransaction().commit();  
1171             }  
1172         }  
1173     }  
1174     db.beginTransaction().begin();  
1175     db.remove(event);  
1176     db.beginTransaction().commit();  
1177     return true;  
1178 }  
1179 }
```

2. Errefaktorizatutako kodea

```
1136@ public boolean gertaeraEzabatu(Event ev) {  
1137     Event event = db.find(Event.class, ev);  
1138     boolean resultB = gertaeraEzabatulag3(event);  
1139     if(resultB == false) {  
1140         return false;  
1141     }else if(new Date().compareTo(event.getEventDate())<0) {  
1142         TypedQuery<Quote> Qquery = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEventNumber() =?1", Quote.class);  
1143         Qquery.setParameter(1, event.getEventNumber());  
1144         List<Quote> listQUO = Qquery.getResultList();  
1145         for(int j=0; j<listQUO.size(); j++) {  
1146             Quote quo = db.find(Quote.class, listQUO.get(j));  
1147             gertaeraEzabatuLag2(quo);  
1148         }  
1149     }  
1150     db.beginTransaction().begin();  
1151     db.remove(event);  
1152     db.beginTransaction().commit();  
1153     return true;  
1154 }  
1155 }
```

3. Egindako errefaktorizazioaren azalpena

Errefaktorizazio hau egiteko, metodoaren hainbat atal metodo ezberdinan bihurtu ditut. Konkretuki 3 metodo berri sortu ditut, *gertaeraEzabatuLag1*, *2* eta *3*. Hauak dira 3 metodo horien kodeak:

```
1157@ private boolean gertaeraEzabatuLag3(Event event) {  
1158     boolean resultB = true;  
1159     List<Question> listQ = event.getQuestions();  
1160     for(Question q : listQ) {  
1161         if(q.getResult() == null) {  
1162             resultB = false;  
1163         }  
1164     }  
1165     return resultB;  
1166 }  
1167  
1168@ private void gertaeraEzabatuLag2(Quote quo) {  
1169     for(int i=0; i<quo.getApustuak().size(); i++) {  
1170         ApustuAnitza apustuAnitza = quo.getApustuak().get(i).getApustuAnitza();  
1171         ApustuAnitza ap1 = db.find(ApustuAnitza.class, apustuAnitza.getApustuAnitzaNumber());  
1172         db.beginTransaction().begin();  
1173         ap1.removeApustua(quo.getApustuak().get(i));  
1174         db.getTransaction().commit();  
1175         gertaeraEzabatuLag1(ap1);  
1176         db.beginTransaction();  
1177         Sport spo = quo.getQuestion().getEvent().getSport();  
1178         spo.setApustukantitatea(spo.getApustukantitatea()-1);  
1179         KirolEstatistikak ke=ap1.getUser().kirolEstatistikakLortu(spo);  
1180         ke.setKont(ke.getKont()-1);  
1181         db.getTransaction().commit();  
1182     }  
1183 }  
1184  
1185@ private void gertaeraEzabatuLag1(ApustuAnitza ap1) {  
1186     if(ap1.getApustuak().isEmpty() && !ap1.getEgoera().equals("galduak")) {  
1187         this.apustuaEzabatu(ap1.getUser(), ap1);  
1188     }else if(!ap1.getApustuak().isEmpty() && ap1.irabazitaMarkatu()){  
1189         this.ApustuaIrabazi(ap1);  
1190     }  
1191 }
```

Ikusten denez, metodo horietan baldintza/begiztak sartu ditut, horrela, *gertaeraEzabatu* metodoaren konplexutasun ziklomatikoa 9 izatetik 4 izatera pasa baita; 1, 2 eta 3 metodo laguntzaileak 9 horietatik 2, 1 eta 2 barneratu baitituzte, hurrenez hurren.

4. Egilea: Aimar Mancisidor

- *emaitzakIpini* metodoa (Konplexutasun ziklomatikoa: 7)

1. Hasierako kodea

```

11010 public void EmaitzakIpini(Quote quote) throws EventNotFinished{
1102
1103     Quote q = db.find(Quote.class, quote);
1104     String result = q.getForecast();
1105
1106     if(new Date().compareTo(q.getQuestion().getEvent().getEventDate())<0)
1107         throw new EventNotFinished();
1108
1109     Vector<Apustua> listApustuak = q.getApustuak();
1110     db.beginTransaction().begin();
1111     Question que = q.getQuestion();
1112     Question question = db.find(Question.class, que);
1113     question.setResult(result);
1114     for(Quote quo: question.getQuotes()) {
1115         for(Apustua apu: quo.getApustuak()) {
1116
1117             Boolean b=apu.galdutaMarkatu(quo);
1118             if(b) {
1119                 apu.getApustuAnitza().setEgoera("galduta");
1120             }else {
1121                 apu.setEgoera("irabazita");
1122             }
1123         }
1124     }
1125     db.getTransaction().commit();
1126     for(Apustua a : listApustuak) {
1127         db.beginTransaction().begin();
1128         Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
1129         db.getTransaction().commit();
1130         if(bool) {
1131             this.ApustuaIrabazi(a.getApustuAnitza());
1132         }
1133     }
1134 }
```

2. Errefaktorizatutako kodea

```

11010 public void EmaitzakIpini(Quote quote) throws EventNotFinished{
1102
1103     Quote q = db.find(Quote.class, quote);
1104     String result = q.getForecast();
1105
1106     if(new Date().compareTo(q.getQuestion().getEvent().getEventDate())<0)
1107         throw new EventNotFinished();
1108
1109     Vector<Apustua> listApustuak = q.getApustuak();
1110     db.beginTransaction().begin();
1111     Question que = q.getQuestion();
1112     Question question = db.find(Question.class, que);
1113     question.setResult(result);
1114     for(Quote quo: question.getQuotes()) { emaitzakIpiniLag1(quo); }
1115     db.beginTransaction().commit();
1116     for(Apustua a : listApustuak) { emaitzakIpiniLag2(a); }
1117 }
```

3. Egindako errefaktorizazioaren azalpena

Errefaktorizazio hau egin ahal izateko, aztertutako metodoaren hainbat atal beste metodo desberdinan banatu ditut. Zehazki 2 metodo berri sortu ditut; *emaitzakIpiniLag1* eta *emaitzakIpiniLag2* direnak, hasierako kodea 12 lerrorekin utziz. Honakoak dira 2 metodo horien kodeak:

```
1118✉ public void emaitzakIpiniLag1(Quote quo) {  
1119    for(Apustua apu: quo.getApustuak()) {  
1120        Boolean b=apu.galdutaMarkatu(quo);  
1121        if(b) {  
1122            apu.getApustuAnitza().setEgoera("galduta");  
1123        }else {  
1124            apu.setEgoera("irabazita");  
1125        }  
1126    }  
1127}  
1128✉ public void emaitzakIpiniLag2(Apustua a) {  
1129    db.beginTransaction().begin();  
1130    Boolean bool=a.getApustuAnitza().irabazitaMarkatu();  
1131    db.getTransaction().commit();  
1132    if(bool) {  
1133        this.ApustuaIrabazi(a.getApustuAnitza());  
1134    }  
1135}  
1136
```

Ikusi daitekeen bezala, metodo hauetan hasierako metodoko baldintza/begiztak sartu ditut. Era horetan, emaitzakIpini metodoaren konplexutasun ziklomatikoa 8 izatetik 4 izatera pasa da. 2 laguntzaileei esker, 9 horiek banatu eta laguntzaileek 3 eta 1 barneratu dituzte.

4. Egilea: Ainhitze Ituarte

- *gertaerakSortu* metodoa (Konplexutasun ziklomatikoa: 5)

- Hasierako kodea

```

882⊕ public boolean gertaerakSortu(String description, Date eventDate, String sport) {
883     boolean b = true;
884     db.beginTransaction().begin();
885     Sport spo = db.find(Sport.class, sport);
886     if(spo!=null) {
887         TypedQuery<Event> Equery = db.createQuery("SELECT e FROM Event e WHERE e.getEventDate() =?1 ",Event.class);
888         Equery.setParameter(1, eventDate);
889         for(Event ev: Equery.getResultList()) {
890             if(ev.getDescription().equals(description)) {
891                 b = false;
892             }
893         }
894     if(b) {
895         String[] taldeak = description.split("-");
896         Team lokala = new Team(taldeak[0]);
897         Team kanpokoa = new Team(taldeak[1]);
898         Event e = new Event(description, eventDate, lokala, kanpokoa);
899         e.setSport(spo);
900         spo.addEvent(e);
901         db.persist(e);
902     }
903 } else {
904     db.beginTransaction().commit();
905     return false;
906 }
907 db.beginTransaction().commit();
908 return b;
909 }
```

- Errefaktorizatutako kodea

```

881⊕ public boolean gertaerakSortu(String description, Date eventDate, String sport) {
882     boolean b = true;
883     db.beginTransaction().begin();
884     b = gertaerakSortuLag1(description, eventDate, sport, b);
885     db.beginTransaction().commit();
886     return b;
887 }
```

- Egindako errefaktorizazioaren azalpena

"Write short units of code" motako errefaktorizazioan egindakoak honetarako ere balio dit. Izan ere, *gertaeraBadago*, *gertaerakSortuLag1* eta *gertaerakSortuLag2* azpimethodoak sortuz, *gertaerakSortu* metodoaren konplexutasun ziklomatikoa 5 izatetik 1 izatera pasa da. Hona hemen hiru azpimethodoak:

```

889⊕ public boolean gertaeraBadago(String description, Date eventDate) {
890     TypedQuery<Event> Equery = db.createQuery("SELECT e FROM Event e WHERE e.getEventDate() =?1 ",Event.class);
891     Equery.setParameter(1, eventDate);
892     for(Event ev: Equery.getResultList()) {
893         if(ev.getDescription().equals(description)) {
894             return false;
895         }
896     }
897     return true;
898 }
899
900⊕ public boolean gertaerakSortuLag1(String description, Date eventDate, String sport, boolean b) {
901     Sport spo = db.find(Sport.class, sport);
902     if(spo!=null) {
903         b = gertaeraBadago(description, eventDate);
904         if(b) {
905             gertaerakSortuLag2(description, eventDate, spo);
906         }
907     } else {
908         b = false;
909     }
910     return b;
911 }
912
913⊕ public void gertaerakSortuLag2(String description, Date eventDate, Sport spo) {
914     String[] taldeak = description.split("-");
915     Team lokala = new Team(taldeak[0]);
916     Team kanpokoa = new Team(taldeak[1]);
917     Event e = new Event(description, eventDate, lokala, kanpokoa);
918     e.setSport(spo);
919     spo.addEvent(e);
920     db.persist(e);
921 }
```

- Egilea: Ander Rubio

“Duplicate code” motako errefaktorizazioak

Mota honetako *code smell*-ak identifikatzeko SonarLint baliabidea erabili dugu.

- “ApustuaEgin” String-a 12 aldiz errepikatzen da. Konstante bat definitu.

1. Hasierako kodea

```
431     Duplication
432     Transaction t1 = new Transaction(reg1, apA1.getBalioa(), new Date(), 1 "ApustuaEgin");
433     Duplication
434     Transaction t3 = new Transaction(reg2, apA4.getBalioa(), new Date(), 2 "ApustuaEgin");
435     Duplication
436     Transaction t4 = new Transaction(reg3, apA5.getBalioa(), new Date(), 3 "ApustuaEgin");
437     Duplication
438     Transaction t5 = new Transaction(reg4, apA3.getBalioa(), new Date(), 4 "ApustuaEgin");
439     Duplication
440     Transaction t6 = new Transaction(reg4, apA6.getBalioa(), new Date(), 5 "ApustuaEgin");
441     Duplication
442     Transaction t7 = new Transaction(reg1, apA7.getBalioa(), new Date(), 6 "ApustuaEgin");
443     Duplication
444     Transaction t8 = new Transaction(reg1, apA8.getBalioa(), new Date(), 7 "ApustuaEgin");
445     Duplication
446     Transaction t9 = new Transaction(reg2, apA9.getBalioa(), new Date(), 8 "ApustuaEgin");
447     Duplication
448     Transaction t10 = new Transaction(reg2, apA10.getBalioa(), new Date(), 9 "ApustuaEgin");
449     Duplication
450     Transaction t11 = new Transaction(reg3, apA11.getBalioa(), new Date(), 10 "ApustuaEgin");
451     Duplication
452     Transaction t12 = new Transaction(reg3, apA12.getBalioa(), new Date(), 11 "ApustuaEgin");
453     Duplication
454     Transaction t = new Transaction(user, balioa, new Date(), 12 "ApustuaEgin");
```

2. Errefaktorizatutako kodea

```
47 public class DataAccess {
48     protected static EntityManager db;
49     protected static EntityManagerFactory emf;
50     private final String apEginString = "ApustuaEgin";
51
52     Transaction t1 = new Transaction(reg1, apA1.getBalioa(), new Date(), apEginString);
53     Transaction t3 = new Transaction(reg2, apA4.getBalioa(), new Date(), apEginString);
54     Transaction t4 = new Transaction(reg3, apA5.getBalioa(), new Date(), apEginString);
55     Transaction t5 = new Transaction(reg4, apA3.getBalioa(), new Date(), apEginString);
56     Transaction t6 = new Transaction(reg4, apA6.getBalioa(), new Date(), apEginString);
57     Transaction t7 = new Transaction(reg1, apA7.getBalioa(), new Date(), apEginString);
58     Transaction t8 = new Transaction(reg1, apA8.getBalioa(), new Date(), apEginString);
59     Transaction t9 = new Transaction(reg2, apA9.getBalioa(), new Date(), apEginString);
60     Transaction t10 = new Transaction(reg2, apA10.getBalioa(), new Date(), apEginString);
61     Transaction t11 = new Transaction(reg3, apA11.getBalioa(), new Date(), apEginString);
62     Transaction t12 = new Transaction(reg3, apA12.getBalioa(), new Date(), apEginString);
63
64     Transaction t = new Transaction(user, balioa, new Date(), apEginString);
```

3. Egindako errefaktorizazioaren azalpena

“ApustuaEgin” aukeratu dut eta *refactor >> Extract Local Variable* egin dut. Hala ere lehen 11 errepikapenak initialize metodoaren barruan daude, eta azkena *ApustuaEgin* metodoan, beraz aldagai lokal batek ez zidan balio. Hori konpontzeko DataAccess klaseari gehitu diot *apEginString* konstantea, eta horrela bai erabili ahal izan dut bi metodoetan.

4. Egilea: Aimar Mancisidor

- “DiruaSartu” String-a 4 aldiz errepikatzen da. Hau konpontzeko konstante bat definitu dut:

1. Hasierako kodea

```
/41
748     Duplication
749     this.DiruaSartu(reg1, 50.0, new Date(), ① "DiruaSartu");
750     Duplication
751     this.DiruaSartu(reg2, 50.0, new Date(), ② "DiruaSartu");
752     Duplication
753     this.DiruaSartu(reg3, 50.0, new Date(), ③ "DiruaSartu");
754     Duplication
755     this.DiruaSartu(reg4, 50.0, new Date(), ④ "DiruaSartu");
756
```

2. Errefaktorizatutako kodea

```
/41
748     String diruaSartuString = "DiruaSartu";
749     Duplication
750     this.DiruaSartu(reg1, 50.0, new Date(), diruaSartuString);
751     Duplication
752     this.DiruaSartu(reg2, 50.0, new Date(), diruaSartuString);
753     Duplication
754     this.DiruaSartu(reg3, 50.0, new Date(), diruaSartuString);
755     Duplication
756     this.DiruaSartu(reg4, 50.0, new Date(), diruaSartuString);
```

3. Egindako errefaktorizazioaren azalpena

Honako *bad smell* hau ezabatu ahal izateko, “DiruaSartu” String-a aukeratu eta *refactor >> Extract Local Variable* egin dut. Modu honetan, String motako objektu bat sortu dit, errepiaktua dagoen String balioa duena eta arazoa ezabatzen duena. Aldaketa egin aurretik preview egin dut eta arazoa DataAccess klasean zegoela bakarrik ikusi dut. Hortaz, ez dut aldaketa gehiagorik egin behar izan.

4. Egilea: Ainhitze Ituarte

- "Who will win the match?" String-a 3 aldiz errepikatzen da. Hau konpontzeko konstante bat definitu dut:

1. Hasierako kodea

```

238     else if (Locale.getDefault().equals(new Locale("en"))) {
239         Duplication
240         q1=ev1.addQuestion(1, "Who will win the match?",1);
241         q2=ev1.addQuestion("Who will score first?",2);
242         Duplication
243         q3=ev11.addQuestion(2, "Who will win the match?",1);
244         q4=ev11.addQuestion("How many goals will be scored in the match?",2);
245         Duplication
246         q5=ev17.addQuestion(3, "Who will win the match?",1);
247         q6=ev17.addQuestion("Will there be goals in the first half?",2);
248     }

```

2. Errefaktorizatutako kodea

```

238     else if (Locale.getDefault().equals(new Locale("en"))) {
239         String whoWins = "Who will win the match?";
240         Duplication
241         q1=ev1.addQuestion(whoWins,1);
242         q2=ev1.addQuestion("Who will score first?",2);
243         Duplication
244         q3=ev11.addQuestion(whoWins,1);
245         q4=ev11.addQuestion("How many goals will be scored in the match?",2);
246         Duplication
247         q5=ev17.addQuestion(whoWins,1);
248         q6=ev17.addQuestion("Will there be goals in the first half?",2);
249     }

```

3. Egindako errefaktorizazioaren azalpena

"Duplicate code" motako errefaktorizazio hau burutzeko, "Who will win the match?" String-a aukeratu eta *refactor >> Extract Local Variable* egin dut. Honela, Eclipsek automatikoki bariable batean "Who will win the match?" String-a gorde dit, eta sortutako berri hau erabiltzen dit String-a behin eta berriz errepikatu beharrean. Aldaketa egin aurretik *preview* tresna erabili dut arazoa DataAccess klasean bakarrik zegoela ziurtatzeko. Beraz, ez dut aldaketa gehiagorik egin behar izan.

4. Egilea: Ander Rubio

"Keep unit interfaces small" motako errefaktorizazioak

- *mezuaBidali* metodoa (5 parametro behar ditu)

1. Hasierako kodea

```
1207 public boolean mezuaBidali(User igorlea, String hartzalea, String titulo, String test, Elkarrizketa elkarrizketa) {  
1208     User igorle = db.find(User.class, igorlea.getUsername());  
1209     User hartzale = db.find(User.class, hartzalea);  
1210     Elkarrizketa elk=null;  
1211     if(hartzale==null) {  
1212         return false;  
1213     }else {  
1214         db.beginTransaction().begin();  
1215         Message m = new Message(igorle, hartzale, test);  
1216         db.persist(m);  
1217         if(elkarrizketa!=null) {  
1218             elk = db.find(Elkarrizketa.class, elkarrizketa.getElkarrizketaNumber());  
1219         }else {  
1220             elk= new Elkarrizketa(titulo, igorle, hartzale);  
1221             db.persist(elk);  
1222             m.setElkarrizketa(elk);  
1223             igorle.addElkarrizketak(elk);  
1224             hartzale.addElkarrizketak(elk);  
1225         }  
1226         elk.addMezua(m);  
1227         igorle.addBidalitakoMezuak(m);  
1228         hartzale.addJasotakoMezuak(m);  
1229         db.beginTransaction().commit();  
1230         return true;  
1231     }  
1232 }  
1233 }
```

2. Errefaktorizatutako kodea

```
1234 public boolean mezuaBidali(User igorlea, String hartzalea, MezuEstruktura me, Elkarrizketa elkarrizketa) {  
1235     User igorle = db.find(User.class, igorlea.getUsername());  
1236     User hartzale = db.find(User.class, hartzalea);  
1237     Elkarrizketa elk=null;  
1238     if(hartzale==null) {  
1239         return false;  
1240     }else {  
1241         db.beginTransaction().begin();  
1242         Message m = new Message(igorle, hartzale, me.getTestua());  
1243         db.persist(m);  
1244         if(elkarrizketa!=null) {  
1245             elk = db.find(Elkarrizketa.class, elkarrizketa.getElkarrizketaNumber());  
1246         }else {  
1247             elk= new Elkarrizketa(me.getTitulo(), igorle, hartzale);  
1248             db.persist(elk);  
1249             m.setElkarrizketa(elk);  
1250             igorle.addElkarrizketak(elk);  
1251             hartzale.addElkarrizketak(elk);  
1252         }  
1253         elk.addMezua(m);  
1254         igorle.addBidalitakoMezuak(m);  
1255         hartzale.addJasotakoMezuak(m);  
1256         db.beginTransaction().commit();  
1257         return true;  
1258     }  
1259 }
```

3. Egindako errefaktorizazioaren azalpena

Errefaktorizatutako kodean ikusi daiteken bezala, *titulo* eta *test* objektuak kendu eta *me* *MezuEstruktura* motako objektu bat ezarri dut. Honi esker, metodoak 4 parametro beharko ditu 5 beharrean. Hau egin ahal izateko, lehenik *MezuEstruktura* izeneko klase berri bat sortu dut *domain-en*, honakoa dena:

```
10 public class MezuEstruktura{  
11     private String testua;  
12     private String titulo;  
13  
14     public MezuEstruktura(String testua, String titulo) {  
15         this.testua = testua;  
16         this.titulo = titulo;  
17     }  
18  
19     public String getTestua() {  
20         return testua;  
21     }  
22  
23     public void setTestua(String testua) {  
24         this.testua = testua;  
25     }  
26  
27     public String getTitulo() {  
28         return titulo;  
29     }  
30  
31     public void setTitulo(String titulo) {  
32         this.titulo = titulo;  
33     }  
34 }
```

Eraikitzaleak beharrezkoak ditugun izenburu eta testuak jasoko ditu, *mezuaBidali* metodorako beharrezkoak ditugunak.

Jarraian, BLFacade aldatu dut, dataAcceseko metodo honi parametro unitate bat kentzean, *facade*-ko metodoan aldaketa bera egin behar dudalako. Honela geratuz:

```
111 @WebMethod public boolean mezuaBidali(User igorle, String hartzalea, MezuEstruktura me, Elkarrizketa m);
112
113 }
```

Gauza bera gertatu da BLFacadeImplementation klasean:

```
276@ @WebMethod
277 public boolean mezuaBidali(User igorle, String hartzalea, MezuEstruktura me, Elkarrizketa m) {
278     dbManager.open(false);
279     Boolean ema = dbManager.mezuaBidali(igorle, hartzalea, me, m);
280     dbManager.close();
281     return ema;
282 }
```

Aldaketa hauek egin ondoren, MezuakBidaliGUI-an erroreak zeudelaz ohartu naiz. Izan ere, *mezuaBidali* metodoari dei egiten dio eta honek 5 parametro izaten jarraitzen zuen:

```
109 btnEnviar = new JButton(ResourceBundle.getBundle("Etiquetas").getString("Enviar"));
110 btnEnviar.setForeground(Color.DARK_GRAY);
111 btnEnviar.setBackground(Color.PINK);
112 btnEnviar.addActionListener(new ActionListener() {
113     public void actionPerformed(ActionEvent e) {
114         if(user.getUsername().equals(txtHartzalea.getText())) {
115             Boolean ema= businesslogic.mezuaBidali(user, txtHartzalea.getText(), txtAsunto.getText(), txtTestua.getText(), elkarrizketa
116             if(ema) {
117                 thisw.setVisible(false);
118                 JFrame a = new SarrearenziaGUI(user);
119                 a.setVisible(true);
120             }else {
121                 lblError.setVisible(true);
122                 lblError.setText(ResourceBundle.getBundle("Etiquetas").getString("HartzError"));
123             }
124         }else {
125             lblError.setVisible(true);
126             lblError.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorMessage"));
127         }
128     }
129 });
130 });

131 }
```

Hau konpontzeko, *MezuEstruktura* motako *me* objektu berri bat sortu dut, zeinek *txtAsunto* eta *txtTestua* parametroak jasoko dituen. Horrela, *meazuaBidali* metodoari dei egitean bi parametro hauek me parametro berriarengatik aldatu eta errorea konpontzen dut, kodea horrela geratuz:

```
111 btnEnviar = new JButton(ResourceBundle.getBundle("Etiquetas").getString("Enviar"));
112 btnEnviar.setForeground(Color.DARK_GRAY);
113 btnEnviar.setBackground(Color.PINK);
114 btnEnviar.addActionListener(new ActionListener() {
115     public void actionPerformed(ActionEvent e) {
116         if(user.getUsername().equals(txtHartzalea.getText())) {
117             me = new MezuEstruktura(txtTestua.getText(), txtAsunto.getText());
118             Boolean ema= businesslogic.mezuaBidali(user, txtHartzalea.getText(), me, elkarrizketa);
119             if(ema) {
120                 thisw.setVisible(false);
121                 JFrame a = new SarrearenziaGUI(user);
122                 a.setVisible(true);
123             }else {
124                 lblError.setVisible(true);
125                 lblError.setText(ResourceBundle.getBundle("Etiquetas").getString("HartzError"));
126             }
127         }else {
128             lblError.setVisible(true);
129             lblError.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorMessage"));
130         }
131     }
132 });
133 }
```

4. Egilea: Ainhitze Ituarte

- ApustuaEgin metodoa (4 parametro jasotzen ditu)

*Oharra: Metodo honek 4 parametro behar dituenez, ez da zehazki *bad smell* bat, baina dataAccess-en ez zeuden metodo gehiago 4 parametro baino gehiago jasotzen zituen *mezuaBidi-taz* aparte, beraz, 4 parametrodun metodoa errefaktorizatzea erabaki dut 3 parametro izatera pasa dadin.

1. Hasierako kodea

```

977@     public boolean ApustuaEgin(User u, Vector<Quote> quote, Double balioa, Integer apustuBikoitzagalarazi) {
978         Registered user = (Registered) db.find(User.class, u.getUsername());
979         Boolean b;
980         if(user.getDirukop()>=balioa) {
981             db.beginTransaction().begin();
982             ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
983             db.persist(apustuAnitza);
984             for(Quote quo: quote) {
985                 Quote kuote = db.find(Quote.class, quo.getQuoteNumber());
986                 Apustua ap = new Apustua(apustuAnitza, kuote);
987                 db.persist(ap);
988                 apustuAnitza.addApustua(ap);
989                 kuote.addApustua(ap);
990             }
991             db.beginTransaction().commit();
992             db.beginTransaction().begin();
993             if(apustuBikoitzagalarazi==1) {
994                 apustuBikoitzagalarazi=apustuAnitza.getApustuAnitzaNumber();
995             }
996             apustuAnitza.setApustukopia(apustuBikoitzagalarazi);
997             user.updateDirukontua(-balioa);
998             Transaction t = new Transaction(user, balioa, new Date(), apEginString);
999             user.addApustuAnitza(apustuAnitza);
1000             for(Apustua a: apustuAnitza.getApustuak()) {
1001                 Apustua apu = db.find(Apustua.class, a.getApostuaNumber());
1002                 Quote q = db.find(Quote.class, apu.getKuota().getQuoteNumber());
1003                 Sport spo = a.getQuestion().getEvent().getSport();
1004                 spo.setApustukantitatea(spo.getApustukantitatea()+1);
1005                 if(!user.containsKirola(spo)) {
1006                     KirolEstatistikak ke=new KirolEstatistikak(user, spo);
1007                     ke.setKont(1);
1008                     user.addKirolEstatistikak(ke);
1009                     spo.addKirolEstatistikak(ke);
1010                 }else {
1011                     KirolEstatistikak ke=user.kirolEstatistikakLortu(spo);
1012                     ke.eguneratuKont(1);
1013                 }
1014             }
1015             user.addTransaction(t);
1016             db.persist(t);
1017             db.beginTransaction().commit();
1018             for(Jarraitzalea reg:user.getJarraitzaleLista()) {
1019                 Jarraitzalea erab=db.find(Jarraitzalea.class, reg.getJarraitzaleaNumber());
1020                 b=true;
1021                 for(ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {
1022                     if(apu.getApustukopia()>=apustuAnitza.getApustukopia()) {
1023                         b=false;
1024                     }
1025                 }
1026                 if(b) {
1027                     if(erab.getNork().getDiruLimitea()<balioa) {
1028                         this.ApustuaEgin(erab.getNork(), quote, erab.getNork().getDiruLimitea(), apustuBikoitzagalarazi);
1029                     }else{
1030                         this.ApustuaEgin(erab.getNork(), quote, balioa, apustuBikoitzagalarazi);
1031                     }
1032                 }
1033             }
1034             return true;
1035         }else{
1036             return false;
1037         }
1038     }
1039 }
```

2. Errefaktorizatutako kodea

ApustuEginGUI klasean:

```
228     if(zenb>=maxMinBet) {
229         ApustuEginKlaseLag apLag = new ApustuEginKlaseLag(zenb, -1);
230         Boolean b = businessLogic.ApustuaEgin(user, quoteVec, apLag);
```

BLFacade klasean:

```
88     @WebMethod public boolean ApustuaEgin(User u, Vector<Quote> q, ApustuEginKlaseLag apLag);  
89 }
```

BLFacadeImplementation klasean:

```
203@ WebMethod  
204 public boolean ApustuaEgin(User u, Vector<Quote> q, ApustuEginKlaseLag apLag) {  
205     dbManager.open(false);  
206     boolean b = dbManager.ApustuaEgin(u, q, apLag);  
207     dbManager.close();  
208     return b;  
209 }
```

DataAccess klasean:

```
978@ public boolean ApustuaEgin(User u, Vector<Quote> quote, ApustuEginKlaseLag aplag) {  
979     Registered user = (Registered) db.find(User.class, u.getUsername());  
980     Boolean b;  
981     if(user.getDirukop()>=aplag.getBalioa()) {  
982         db.beginTransaction().begin();  
983         ApustuAnitza apustuAnitza = new ApustuAnitza(user, aplag.getBalioa());  
984         db.persist(apustuAnitza);  
985         for(Quote quo: quote) {  
986             Quote kuote = db.find(Quote.class, quo.getQuoteNumber());  
987             Apustua ap = new Apustua(apustuAnitza, kuote);  
988             db.persist(ap);  
989             apustuAnitza.addApustua(ap);  
990             kuote.addApustua(ap);  
991         }  
992         db.getTransaction().commit();  
993         db.beginTransaction().begin();  
994         if(apLag.getApustuBikoitzagaLalarazi() == -1) {  
995             aplag.setApustuBikoitzagaLalarazi(apustuAnitza.getApustuAnitzaNumber());  
996         }  
997         apustuAnitza.setApustuKopia(apLag.getApustuBikoitzagaLalarazi());  
998         user.updateDiruKontua(-aplag.getBalioa());  
999         Transaction t = new Transaction(user, aplag.getBalioa(), new Date(), apEginString);  
1000         user.addApustuaAnitza(apustuAnitza);  
1001         for(Apustua a: apustuAnitza.getApustuak()) {  
1002             Apustua apu = db.find(Apustua.class, a.getApostuaNumber());  
1003             Quote q = db.find(Quote.class, apu.getKuota().getQuoteNumber());  
1004             Sport spo = q.getQuestion().getEvent().getSport();  
1005             spo.setApustuKantitatea(spo.getApustuKantitatea() + 1);  
1006             if(!user.containsKirola(spo)) {  
1007                 KirolEstatistikak ke = new KirolEstatistikak(user, spo);  
1008                 ke.setKont(1);  
1009                 user.addKirolEstatistikak(ke);  
1010                 spo.addKirolEstatistikak(ke);  
1011             } else {  
1012                 KirolEstatistikak ke = user.kirolEstatistikakLortu(spo);  
1013                 ke.egeneratuKont(1);  
1014             }  
1015         }  
1016         user.addTransaction(t);  
1017         db.persist(t);  
1018         db.beginTransaction().commit();  
1019         for(Jarraitzalea reg: user.getJarraitzaileLista()) {  
1020             Jarraitzailea erab = db.find(Jarraitzalea.class, reg.getJarraitzaileaNumber());  
1021             b = true;  
1022             for(ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {  
1023                 if(apu.getApustuKopia() == apustuAnitza.getApustuKopia()) {  
1024                     b = false;  
1025                 }  
1026             }  
1027             if(b) {  
1028                 if(erab.getNork().getDiruLimitea() < aplag.getBalioa()) {  
1029                     ApustuEginKlaseLag aplag2 = new ApustuEginKlaseLag(erab.getNork().getDiruLimitea(), aplag.getApustuBikoitzagaLalarazi());  
1030                     this.ApustuaEgin(erab.getNork(), quote, aplag2);  
1031                 } else {  
1032                     ApustuEginKlaseLag aplag2 = new ApustuEginKlaseLag(apLag.getBalioa(), aplag.getApustuBikoitzagaLalarazi());  
1033                     this.ApustuaEgin(erab.getNork(), quote, aplag2);  
1034                 }  
1035             }  
1036         }  
1037         return true;  
1038     } else {  
1039         return false;  
1040     }  
1041 }
```

3. Egindako errefaktorizazioaren azalpena

Errefaktorizazio hau egiteko klase berri bat sortu dut, *ApustuEginKlaseLag*, eta honakoa da bere kodea:

```
1 package domain;
2
3 public class ApustuEginKlaseLag {
4     private Double balioa;
5     private Integer apustuBikoitzagaLarazi;
6
7     public ApustuEginKlaseLag(Double balioa, Integer apustuBikoitzagaLarazi) {
8         this.balioa=balioa;
9         this.apustuBikoitzagaLarazi=apustuBikoitzagaLarazi;
10    }
11
12    public Double getBalioa() {
13        return balioa;
14    }
15
16    public void setBalioa(Double balioa) {
17        this.balioa = balioa;
18    }
19
20    public Integer getApustuBikoitzagaLarazi() {
21        return apustuBikoitzagaLarazi;
22    }
23
24    public void setApustuBikoitzagaLarazi(Integer apustuBikoitzagaLarazi) {
25        this.apustuBikoitzagaLarazi = apustuBikoitzagaLarazi;
26    }
27 }
```

Klase honetan *ApustuaEgin* metodoak jasotzen zituen bi parametro atributu bezala ditu, *balioa* eta *apustuBikoitzagaLarazi*. Horrela *ApustuaEgin* metodoan balioa eta *apustuBikoitzagaLarazi* jaso beharrean *ApustuEginKlaseLag* motako objektu bat pasatuko zaio, eta balioa edo *apustuBikoitzagaLarazi* behar ditugunean *getterrak* erabiltzen ditugu.

4. Egilea: Aimar Mancisidor

- DiruaSartu metodoa (4 parametro jasotzen ditu)

*Oharra: Metodo honek 4 parametro behar dituenez, ez da zehazki *bad smell* bat, baina *dataAccess-en* ez zeuden metodo gehiago 4 parametro baino gehiago jasotzen zituen *mezuaBidi-taz* aparte, beraz 4 parametrodun metodoa errefaktorizatzea erabaki dut 3 parametro izatera pasa dadin.

1. Hasierako kodea

1.1. DataAccess-en:

```

967@ public void DiruaSartu(User u, Double dirua, Date data, String mota) {
968    Registered user = (Registered) db.find(User.class, u.getUsername());
969    db.beginTransaction().begin();
970    Transaction t = new Transaction(user, dirua, data, mota);
971    System.out.println(t.getMota());
972    user.addTransaction(t);
973    user.updateDirukontua(dirua);
974    db.persist(t);
975    db.getTransaction().commit();
976 }

eta

//+
748      this.DiruaSartu(reg1, 50.0, new Date(), "DiruaSartu");
749      this.DiruaSartu(reg2, 50.0, new Date(), "DiruaSartu");
750      this.DiruaSartu(reg3, 50.0, new Date(), "DiruaSartu");
751      this.DiruaSartu(reg4, 50.0, new Date(), "DiruaSartu");
752

```

1.2. DiruaSartuGUI-n::

```

77      businessLogic.DiruaSartu(user, zenb, "DiruaSartu");

```

1.3. BLFacade-n:

```

82
83     @WebMethod public void DiruaSartu(User u, Double dirua, String mota);
84

```

1.4. BLFacadeImplementation-en:

```

195@     @WebMethod
196     public void DiruaSartu(User u, Double dirua, String mota) {
197         Date data = new Date();
198         dbManager.open(false);
199         dbManager.DiruaSartu(u, dirua, data, mota);
200         dbManager.close();
201     }

```

2. Errefaktorizatutako kodea

2.1. DataAccess-en:

```
982+     public void DiruaSartu(User u, Date data, DiruKudeatzailea dk) {  
983         Registered user = (Registered) db.find(User.class, u.getUsername());  
984         db.getTransaction().begin();  
985         Transaction t = new Transaction(user, dk.getDirua(), data, dk.getMota());  
986         System.out.println(t.getMota());  
987         user.addTransaction(t);  
988         user.updateDiruKontua(dk.getDirua());  
989         db.persist(t);  
990         db.getTransaction().commit();  
991     }  
  
eta  
  
751     this.DiruaSartu(reg1, new Date(), new DiruKudeatzailea(50.0, diruaSartuString));  
752     this.DiruaSartu(reg2, new Date(), new DiruKudeatzailea(50.0, diruaSartuString));  
753     this.DiruaSartu(reg3, new Date(), new DiruKudeatzailea(50.0, diruaSartuString));  
754     this.DiruaSartu(reg4, new Date(), new DiruKudeatzailea(50.0, diruaSartuString));  
...  
...
```

2.2. DiruaSartuGUI-n:

```
77  
78         businessLogic.DiruaSartu(user, new DiruKudeatzailea(zenb, "DiruaSartu"));  
...
```

2.3. BLFacade-n:

```
83  
84     @WebMethod public void DiruaSartu(User u, DiruKudeatzailea dk);  
85
```

2.4. BLFacadeImplementation-en:

```
196+     @WebMethod  
197     public void DiruaSartu(User u, DiruKudeatzailea dk) {  
198         Date data = new Date();  
199         dbManager.open(false);  
200         dbManager.DiruaSartu(u, data, dk);  
201         dbManager.close();  
202     }
```

3. Egindako errefaktorizazioaren azalpena

Errefaktorizazio hau burutzeko klase berri bat sortu dut, *DiruKudeatzailea* izenekoa. Hona hemen bere kodea:

```
DiruKudeatzailea.java ×
1 package domain;
2
3 public class DiruKudeatzailea {
4
5     private Double dirua;
6     private String mota;
7
8     public DiruKudeatzailea(Double dirua, String mota) {
9         this.dirua = dirua;
10        this.mota = mota;
11    }
12
13    public Double getDirua() {
14        return dirua;
15    }
16
17    public void setDirua(Double dirua) {
18        this.dirua = dirua;
19    }
20
21    public String getMota() {
22        return mota;
23    }
24
25    public void setMota(String mota) {
26        this.mota = mota;
27    }
28
29 }
```

Klase honen barruan *DiruaSartu* metodoak jasotzen zituen *dirua* eta *mota* balioak jaso ditut atributu bezala. Honela, metodoari dei egiten zaion bakoitzean *dirua* eta *mota* sartu ordez nahikoa da *DiruKudeatzailea* motako objektu bat sartzea, eta *getters*-en bidez *dirua* eta *mota* eskuratzea.

4. Egilea: Ander Rubio