

1. Write short units of code

Bi metodo hartu ditugu 15 lerro baino gehiago dituztenak eta lerro kopurua gutxitu ditugu metodo berriak sortuz.

1.1

Metodoa: GertaerakSortu

Lehen zegoena:

```
737
738  public boolean gertaerakSortu(final String description,final Date eventDate, final S
739      boolean b = true;
740      db.beginTransaction().begin();
741      final Sport spo =db.find(Sport.class, sport);
742      if(spo!=null) {
743          final TypedQuery<Event> equery = db.createQuery("SELECT e FROM Event e WHERE
744              equery.setParameter(1, eventDate);
745          for(final Event ev: equery.getResultList()) {
746              if(ev.getDescription().equals(description)) {
747                  b = false;
748              }
749          }
750          if(b) {
751              final String[] taldeak = description.split("-");
752              final Team lokala = new Team(taldeak[0]);
753              final Team kanpokoa = new Team(taldeak[1]);
754              final Event e = new Event(description, eventDate, lokala, kanpokoa);
755              e.setSport(spo);
756              spo.addEvent(e);
757              db.persist(e);
758          }
759      }else {
760          return false;
761      }
762      db.beginTransaction().commit();
763      return b;
764  }
765 }
```

Egindakoa: Refactor -> Extract Method

Refactor ondoren:

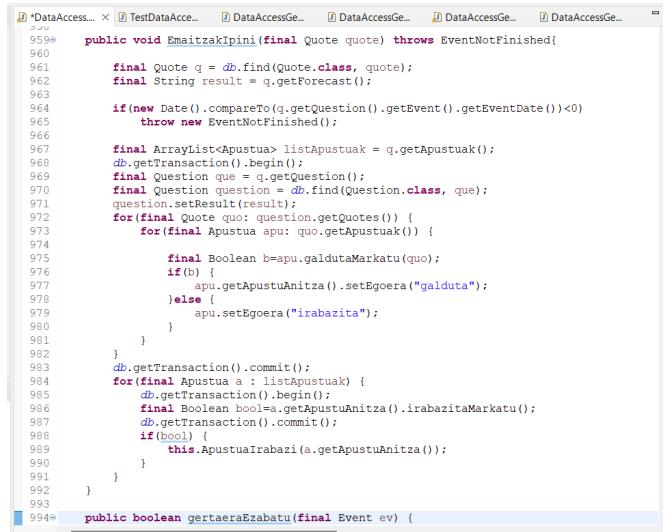
```
*DataAccess.... * TestDataAcc... * DataAccessGe... * DataAccessGe... * DataAccessGe...
738  public boolean gertaerakSortu(final String description,final Date eventDate, final S
739      boolean b = true;
740      db.beginTransaction().begin();
741      final Sport spo =db.find(Sport.class, sport);
742      if(spo!=null) {
743          b = gertaeraSortu2(description, eventDate, b, spo);
744      }else {
745          return false;
746      }
747      db.beginTransaction().commit();
748      return b;
749  }
750
751  private boolean gertaeraSortu2(final String description, final Date eventDate, boolean b
752      final TypedQuery<Event> equery = db.createQuery("SELECT e FROM Event e WHERE e.g
753          equery.setParameter(1, eventDate);
754          for(final Event ev: equery.getResultList()) {
755              if(ev.getDescription().equals(description)) {
756                  b = false;
757              }
758          }
759          if(b) {
760              eventuaSortu(description, eventDate, spo);
761          }
762          return b;
763      }
764
765  private void eventuaSortu(final String description, final Date eventDate, final Spor
766      final String[] taldeak = description.split("-");
767      final Team lokala = new Team(taldeak[0]);
768      final Team kanpokoa = new Team(taldeak[1]);
769      final Event e = new Event(description, eventDate, lokala, kanpokoa);
770      e.setSport(spo);
771      spo.addEvent(e);
772      db.persist(e);
773  }
```

Egilea: Helene Astaburuaga Velasco

1.2

Metodoa: Emaitzaklpinia

Lehen zegoena:



```
59 public void EmaitzaKipini(final Quote quote) throws EventNotFinished{
60     final Quote q = db.find(Quote.class, quote);
61     final String result = q.getForecast();
62
63     if(new Date().compareTo(q.getQuestion().getEvent().getEventDate())<0)
64         throw new EventNotFinished();
65
66     final ArrayList<Apustua> listApustuak = q.getApustuak();
67     db.beginTransaction().begin();
68     final Question que = q.getQuestion();
69     final Question question = db.find(Question.class, que);
70     question.setResult(result);
71
72     for(final Quote quo: question.getQuotes()) {
73         for(final Apustua apu: quo.getApustuak()) {
74             final Boolean b=apu.galdutaMarkatu(quo);
75             if(b) {
76                 apu.getApustuAnitza().setEgoera("galduta");
77             }else {
78                 apu.setEgoera("irabazita");
79             }
80         }
81     }
82     db.getTransaction().commit();
83     for(final Apustua a : listApustuak) {
84         db.beginTransaction().begin();
85         final Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
86         db.getTransaction().commit();
87         if(bool) {
88             this.ApustuaIrabazi(a.getApustuAnitza());
89         }
90     }
91 }
92
93 public boolean gertaeraEzabatu(final Event ev) {
94 }
```

Egindakoa: Refactor -> Extract Method

Refactor egin ondoren:

```
59 public void EmaitzaKipini(final Quote quote) throws EventNotFinished{
60
61     final Quote q = db.find(Quote.class, quote);
62     final String result = q.getForecast();
63
64     if(new Date().compareTo(q.getQuestion().getEvent().getEventDate())<0)
65         throw new EventNotFinished();
66
67     final ArrayList<Apustua> listApustuak = q.getApustuak();
68     db.beginTransaction().begin();
69     emaitzaIpini2(q, result);
70     db.getTransaction().commit();
71     emaitzaIpini3(listApustuak);
72 }
73
74 private void emaitzaIpini3(final ArrayList<Apustua> listApustuak) {
75     for(final Apustua a : listApustuak) {
76         db.beginTransaction().begin();
77         final Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
78         db.getTransaction().commit();
79         if(bool) {
80             this.ApustuaIrabazi(a.getApustuAnitza());
81         }
82     }
83 }
84
85 private void emaitzaIpini2(final Quote q, final String result) {
86     final Question que = q.getQuestion();
87     final Question question = db.find(Question.class, que);
88     question.setResult(result);
89     for(final Quote quo: question.getQuotes()) {
90         for(final Apustua apu: quo.getApustuak()) {
91
92             final Boolean b=apu.galdutaMarkatu(quo);
93             if(b) {
94                 apu.getApustuAnitza().setEgoera("galduta");
95             }else {
96                 apu.setEgoera("irabazita");
97             }
98         }
99     }
100 }
```

Egilea: Irati Kintana

2. Write simple units of code

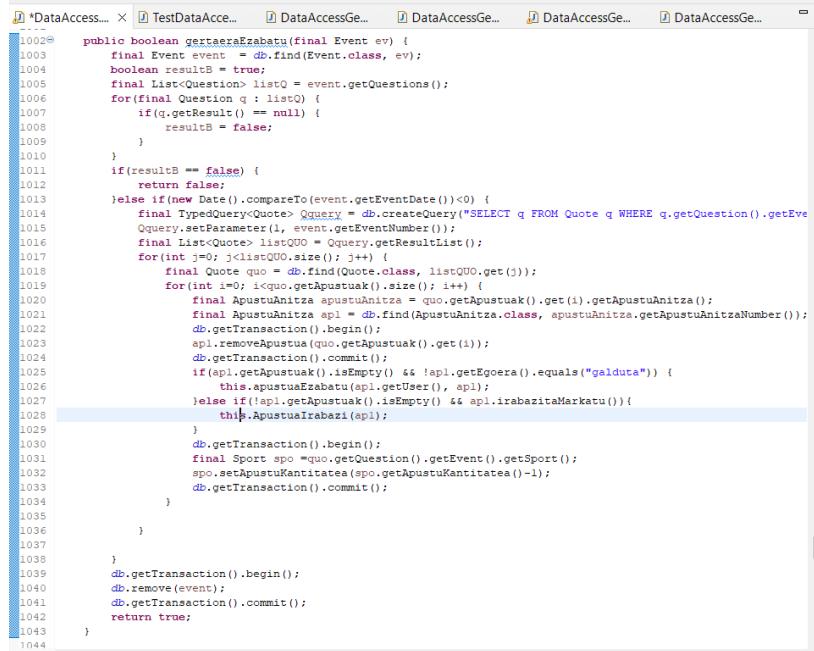
Bi metodo hartu ditugu konplexutasuna 4 baino gehiago dituztenak eta konplexutasuna gutxitu ditugu metodo berriak sortuz.

2.1

Metodoa: gertaerakEzabatu

Lehen zegoena:

- Hasierako konplexutasuna: 8



```
1002@ public boolean gertaeraEzabatu(final Event ev) {
1003     final Event event = db.find(Event.class, ev);
1004     boolean resultB = true;
1005     final List<Question> listQ = event.getQuestions();
1006     for(final Question q : listQ) {
1007         if(q.getResult() == null) {
1008             resultB = false;
1009         }
1010     }
1011     if(resultB == false) {
1012         return false;
1013     } else if(new Date().compareTo(event.getEventDate())<0) {
1014         final TypedQuery<Quote> query = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEventNumber() = :eventNumber");
1015         query.setParameter(1, event.getEventNumber());
1016         final List<Quote> listQUO = query.getResultList();
1017         for(int j=0; j<listQUO.size(); j++) {
1018             final Quote quo = db.find(Quote.class, listQUO.get(j));
1019             for(int i=0; i<quo.getApustuak().size(); i++) {
1020                 final ApustuaAnitza apustuaAnitza = quo.getApustuak().get(i).getApustuaAnitza();
1021                 final ApustuaAnitza apl = db.find(ApustuaAnitza.class, apustuaAnitza.getApustuaAnitzaNumber());
1022                 db.beginTransaction().begin();
1023                 apl.removeApustua(quo.getApustuak().get(i));
1024                 db.getTransaction().commit();
1025                 if(apl.getApustuak().isEmpty() && !apl.getUser().equals("galdueta")) {
1026                     this.apustuaEzabatu(apl.getUser(), apl);
1027                 } else if(!apl.getApustuak().isEmpty() && apl.irabazitaMarkatu()) {
1028                     this.apustuaIrabazi(apl);
1029                 }
1030                 db.beginTransaction().begin();
1031                 final Sport spo = quo.getQuestion().getEvent().getSport();
1032                 spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
1033                 db.getTransaction().commit();
1034             }
1035         }
1036     }
1037 }
1038
1039 db.beginTransaction().begin();
1040 db.remove(event);
1041 db.beginTransaction().commit();
1042 return true;
1043 }
1044 }
```

Egindakoa: Refactor -> Extract Method

Refactor egin ondoren:

- Lortutako konplexutasuna: 2



```
1000 }
1001
1002@ public boolean gertaeraEzabatu(final Event ev) {
1003     final Event event = db.find(Event.class, ev);
1004     boolean resultB = true;
1005     resultB = galderakBegiratu(event, resultB);
1006     if(resultB == false) {
1007         return false;
1008     }
1009     if(new Date().compareTo(event.getEventDate())<0) {
1010         eventBegiratu(event);
1011     }
1012     db.beginTransaction().begin();
1013     db.remove(event);
1014     db.beginTransaction().commit();
1015     return true;
1016 }
1017
1018 }
```

```

1019 private void eventBegiratu(final Event event) {
1020     final TypedQuery<Quote> Qquery = db.createQuery("SELECT q FROM Quote q WHERE q.g
1021     Qquery.setParameter(1, event.getEventNumber());
1022     final List<Quote> listQUO = Qquery.getResultList();
1023     for(int j=0; j<listQUO.size(); j++) {
1024         final Quote quo = db.find(Quote.class, listQUO.get(j));
1025         for(int i=0; i<quo.getApustuak().size(); i++) {
1026             final ApustuAnitza apustuAnitza = quo.getApustuak().get(i).getApustuAnit
1027             final ApustuAnitza apl = db.find(ApustuAnitza.class, apustuAnitza.getApu
1028             db.beginTransaction().begin();
1029             apl.removeApustua(quo.getApustuak().get(i));
1030             db.getTransaction().commit();
1031             apustuaBegiratu(apl);
1032             db.beginTransaction().begin();
1033             final Sport spo = quo.getQuestion().getEvent().getSport();
1034             spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
1035             db.getTransaction().commit();
1036         }
1037     }
1038 }
1039 }

1041 private void apustuaBegiratu(final ApustuAnitza apl) {
1042     if(apl.getApustuak().isEmpty() && !apl.getEgoera().equals("galdua")) {
1043         this.apustuaEzabatu(apl.getUser(), apl);
1044     }else if(!apl.getApustuak().isEmpty() & apl.irabazitaMarkatu()){
1045         this.ApustuaIrabazi(apl);
1046     }
1047 }
1048

1049 private boolean galderakBegiratu(final Event event, boolean resultB) {
1050     final List<Question> listQ = event.getQuestions();
1051     for(final Question q : listQ) {
1052         if(q.getResult() == null) {
1053             resultB = false;
1054         }
1055     }
1056     return resultB;
1057 }
1058

```

Egilea: Helene Astaburuaga Velasco

2.2

Metodoa: ApustuAnitza

Lehen zegoena:

- Hasierako konplexutasuna: 10

```

844 public boolean ApustuaEgin(final Registered u, final Vector<Quote> quote, final Double balioa, Integer apustuBikotzaGalarazi) {
845     final Registered user = (Registered) db.find(Registered.class, u.getUsername());
846     Boolean b=false;
847     if(db.getTxnScope()==Balioa) {
848         db.beginTransaction();
849         final ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
850         db.persist(apustuAnitza);
851         for(final Quote kuote : quote) {
852             final Quote kute = db.find(Quote.class, quo.getQuoteNumber());
853             final Apustua ap = new Apustua(apustuAnitza, kuote);
854             db.persist(ap);
855             apustuAnitza.addApustua(ap);
856             kute.addApustua(ap);
857         }
858         db.beginTransaction().commit();
859         db.beginTransaction();
860         if(apustuBikotzaGalarazi==11) {
861             apustuBikotzaSalzairia=apustuAnitza.getApustuAnitzaNumber();
862         }
863         apustuAnitza.setApustuEgia(apustuBikotzaGalarazi);
864         user.updateDienBikotza(balioa);
865         final Transaction t = new Transaction(user, new Date(), "ApustuaEgin");
866         user.addApustuAnitza(apustuAnitza);
867         db.beginTransaction();
868         for(final Apustua a : apustuAnitza.getApustuak()) {
869             final Jarraitzailea regiser = a.getApustuak();
870             final Quote q = db.find(Quote.class, a.getApustuak().getQuoteNumber());
871             final Sport spo = q.getQuestion().getEvent().getSport();
872             spo.setApustuKantitatea(spo.getApustuKantitatea()+1);
873         }
874         user.addTransaction(t);
875         db.persist(t);
876         db.beginTransaction().commit();
877         for(final Jarraitzailea regiser : jarraitzailea) {
878             final Jarraitzailea erab = db.find(Jarraitzailea.class, regiser.getJarraitzaileaNumber());
879             b=true;
880             for(final ApustuAnitza apu : erab.getApustuak()) {
881                 if(apu.getApustuak().equals(apustuAnitza.getApustuak())) {
882                     b=false;
883                 }
884             }
885         }
886         if(b) {
887             if(erab.getNork().getDiruLumiera()==balioa) {
888                 this.ApustuaEgin(erab.getNork(), quote, erab.getNork().getDiruLumiera(), apustuBikotzaGalarazi);
889             }else{
890                 this.ApustuaEgin(erab.getNork(), quote, balioa, apustuBikotzaGalarazi);
891             }
892         }
893     }
894     return true;
895 }else{
896     return false;
897 }
898

```

Egindakoa: Refactor -> Extract Method

Refactor egin ondoren:

- Lortutako konplexutasuna: 2

```
public boolean ApustuaEgin(final Registered u, final Vector<Quote> quote, final Doub
final Registered user = (Registered) db.find(Registered.class, u.getUsername());
Boolean b;
if(user.getDirukop()>=balioa) {
    db.beginTransaction().begin();
    final ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
    db.persist(apustuAnitza);
    apustuBikoitzagalearazi = apustuAnitza(quote, balioa, apustuBikoitzagalearazi,
    final Transaction t = new Transaction(user, balioa, new Date(), "ApustuaEgin
user.addApustuAnitza(apustuAnitza);
apustuKantitatea(apustuAnitza);
user.addTransaction(t);
db.persist(t);
db.beginTransaction().commit();
jarraitzaleakBegiratu(quote, balioa, apustuBikoitzagalearazi, user, apustuAn
    return true;
} else{
    return false;
}

}

private void jarraitzaleakBegiratu(final Vector<Quote> quote, final Double balioa,
final Registered user, final ApustuAnitza apustuAnitza) {
Boolean b;
for(final Jarraitzalea reg:user.getJarraitzaleaLista()) {
    final Jarraitzalea erab=db.find(Jarraitzalea.class, reg.getJarraitzaleaN
    b=true;
    b = jarraitzaleakBegiratu2(apustuAnitza, b, erab);
    if(b) {
        if(erab.getNork().getDiruLimitea()<balioa) {
            this.ApustuaEgin(erab.getNork(), quote, erab.getNork().getDiruLimite
        }else{
            this.ApustuaEgin(erab.getNork(), quote, balioa, apustuBikoitzagalearazi
        }
    }
}
}

private Boolean jarraitzaleakBegiratu2(final ApustuAnitza apustuAnitza, Boolean b,
for(final ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {
    if(apu.getApustuKopia().equals(apustuAnitza.getApustuKopia())) {
        b=false;
    }
}
return b;
}

private void apustuKantitatea(final ApustuAnitza apustuAnitza) {
for(final Apustua a: apustuAnitza.getApustuak()) {
    final Apustua apu = db.find(Apustua.class, a.getApostuaNumber());
    final Quote q = db.find(Quote.class, apu.getKuota().getQuoteNumber());
    final Sport spo =q.getQuestion().getEvent().getSport();
    spo.setApustuKantitatea(spo.getApustuKantitatea()+1);
}
}
```

Egilea: Irati Kintana

3. Duplicate code

Kodean duplicate zeuden lerroak kendu ditugu parametro berri bat sortuz(extract constant eginez).

3.1

Lehen zegoena:

```
Duplication
final Transaction t1 = new Transaction(reg1, apA1.getBalioa(), new Date(), 1 "ApustuaEgin");
Duplication
final Transaction t3 = new Transaction(reg2, apA4.getBalioa(), new Date(), 2 "ApustuaEgin");
Duplication
final Transaction t4 = new Transaction(reg3, apA5.getBalioa(), new Date(), 3 "ApustuaEgin");
Duplication
final Transaction t5 = new Transaction(reg4, apA3.getBalioa(), new Date(), 4 "ApustuaEgin");
Duplication
final Transaction t6 = new Transaction(reg4, apA6.getBalioa(), new Date(), 5 "ApustuaEgin");
Duplication
final Transaction t7 = new Transaction(reg1, apA7.getBalioa(), new Date(), 6 "ApustuaEgin");
Duplication
final Transaction t8 = new Transaction(reg1, apA8.getBalioa(), new Date(), 7 "ApustuaEgin");
Duplication
final Transaction t9 = new Transaction(reg2, apA9.getBalioa(), new Date(), 8 "ApustuaEgin");
Duplication
final Transaction t10 = new Transaction(reg2, apA10.getBalioa(), new Date(), 9 "ApustuaEgin");
Duplication
final Transaction t11 = new Transaction(reg3, apA11.getBalioa(), new Date(), 10 "ApustuaEgin");
Duplication
final Transaction t12 = new Transaction(reg3, apA12.getBalioa(), new Date(), 11 "ApustuaEgin");
```

Egindakoa: Refactor -> Extract Constant

Refactor egin ondoren:

```
private static final String APUSTUA_EGIN = "ApustuaEgin";
protected static EntityManager db;
protected static EntityManagerFactory emf;

7
8
9 final Transaction t1 = new Transaction(reg1, apA1.getBalioa(), new Date(), APUSTUA_EGIN);
0 final Transaction t3 = new Transaction(reg2, apA4.getBalioa(), new Date(), APUSTUA_EGIN);
1 final Transaction t4 = new Transaction(reg3, apA5.getBalioa(), new Date(), APUSTUA_EGIN);
2 final Transaction t5 = new Transaction(reg4, apA3.getBalioa(), new Date(), APUSTUA_EGIN);
3 final Transaction t6 = new Transaction(reg4, apA6.getBalioa(), new Date(), APUSTUA_EGIN);
4 final Transaction t7 = new Transaction(reg1, apA7.getBalioa(), new Date(), APUSTUA_EGIN);
5 final Transaction t8 = new Transaction(reg1, apA8.getBalioa(), new Date(), APUSTUA_EGIN);
6 final Transaction t9 = new Transaction(reg2, apA9.getBalioa(), new Date(), APUSTUA_EGIN);
7 final Transaction t10 = new Transaction(reg2, apA10.getBalioa(), new Date(), APUSTUA_EGIN);

8 final Transaction t11 = new Transaction(reg3, apA11.getBalioa(), new Date(), APUSTUA_EGIN);
9 final Transaction t12 = new Transaction(reg3, apA12.getBalioa(), new Date(), APUSTUA_EGIN);
0
1 j1.addTransaction(t1);
2 j2.addTransaction(t3);
3 j3.addTransaction(t4);
```

Egilea: Helene Astaburuaga Velasco

3.2

Lehen zegoena:

```
Duplication
this.DiruaSartu(reg1, 50.0, new Date(), 1 "DiruaSartu");
Duplication
this.DiruaSartu(reg2, 50.0, new Date(), 2 "DiruaSartu");
Duplication
this.DiruaSartu(reg3, 50.0, new Date(), 3 "DiruaSartu");
Duplication
this.DiruaSartu(reg4, 50.0, new Date(), 4 "DiruaSartu");

System.out.println("Db initialized");
}
catch (final Exception e){
    e.printStackTrace();
}
}
```

Egindakoa: Refactor -> Extract Constant

Refactor egin ondoren:

```

        db.getTransaction().commit();

        Duplication
        this.DiruaSartu(reg1, 50.0, new Date(), DIRUA_SARTU);
        Duplication
        this.DiruaSartu(reg2, 50.0, new Date(), DIRUA_SARTU);
        Duplication
        this.DiruaSartu(reg3, 50.0, new Date(), DIRUA_SARTU);
        Duplication
        this.DiruaSartu(reg4, 50.0, new Date(), DIRUA_SARTU);

        System.out.println("Db initialized");
    } catch (final Exception e) {
        e.printStackTrace();
    }
}

private static final String DIRUA_SARTU = "DiruaSartu";
private static final String APUSTUA_EGIN = "ApustuaEgin";
protected static EntityManager db;

```

Egilea: Irati Kintana

4. Keep unit interfaces small

Sarrerako parametroak 4 baino gehiago duten metodoak hartu eta sarrerako parametro bakarrera bihurtu ditugu. Horretarako klase berriak sortu ditugu atributu bezala sarrerako parametro guztiak jarriz eta gero metodoan sarrera bezala klase berria jarri dugu .

4.1

Lehen zegoena:

Metodoa: ApustuaEgin

```

844 db.getTransaction().commit();

845
846
847 boolean ApustuaEgin(final Registered u, final Vector<Quote> quote, final Double balioa, Integer a);
848 final Registered user = (Registered) db.find(Registered.class, u.getUsername());
849 oolean b;
850 f(user.getDirukop()>=balioa) {
851     db.beginTransaction().begin();
852     final ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
853     db.persist(apustuAnitza);
854
855     apustuBikoitzagaLarazi = apustuAnitza(quote, balioa, apustuBikoitzagaLarazi, user, apustuAnitza)
856
857     final Transaction t = new Transaction(user, balioa, new Date(), APUSTUA_EGIN);
858     user.addApustuAnitza(apustuAnitza);
859     user.addTransaction(t);
860     db.beginTransaction().begin();
861     db.persist(t);
862     db.getTransaction().commit();
863     jarraitzaileakBegiratu(quote, balioa, apustuBikoitzagaLarazi, user, apustuAnitza);
864     return true;
865 else{
866     return false;
867

```

Metodoa: apustuAnitza

```

897     spo.setApustuKantitatea(spo.getApustuKantitatea()+1);

898
899 }
900
901 private Integer apustuAnitza(final Vector<Quote> quote, final Double balioa, Integer apustuBikoitzagaLarazi,
902 final Registered user, final ApustuAnitza apustuAnitza) {
903     for(final Quote quo: quote) {
904         final Quote kuote = db.find(Quote.class, quo.getQuoteNumber());
905         final Apustua ap = new Apustua(apustuAnitza, kuote);
906         db.persist(ap);
907         apustuAnitza.addApustua(ap);
908         kuote.addApustua(ap);
909     }
910     db.getTransaction().commit();
911     db.beginTransaction().begin();
912     if(apustuBikoitzagaLarazi--<=1) {
913         apustuBikoitzagaLarazi=apustuAnitza.getApustuAnitzaNumber();
914     }
915     apustuAnitza.setApustuKopiea(apustuBikoitzagaLarazi);
916     user.updateDiruKontua(-balioa);
917     user.updateDiruKontua(balioa);
918     return apustuBikoitzagaLarazi;
919 }
920

```

Refactor egin ondoren:

Metodoa: ApustuaEgin

```
845     db.getTransaction().commit();
846
847
848     c boolean ApustuaEgin(final Registered u, final Vector<Quote> quote, final Double balioa, Integer a);
849     final Registered user = (Registered) db.find(Registered.class, u.getUsername());
850     boolean b;
851     f(user.getDirukop())>=balioa) {
852         db.beginTransaction().begin();
853         final ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
854         db.persist(apustuAnitza);
855         ApustuDataAccess apustuDA= new ApustuDataAccess(quote, balioa, apustuBikoitzagalarazi, user, apu:
856         apustubikoitzagalarazi = apustuAnitza(apustuDA);
857
858         final Transaction t = new Transaction(user, balioa, new Date(), APUSTUA_EGIN);
859         user.addApustuAnitza(apustuAnitza);
860         apustukantitatea(apustuAnitza);
861         user.addTransaction(t);
862         db.persist(t);
863         db.beginTransaction().commit();
864         jarraitzaileakBegiratu(quote, balioa, apustubikoitzagalarazi, user, apustuAnitza);
865         return true;
866     else{
867         return false;
868
869 }
```

Metodoa: apustuAnitza

```
902
903     private Integer apustuAnitza(ApustuDataAccess apustuDA) {
904         ApustuAnitza apustuAnitza= apustuDA.getApustuAnitza();
905         int apustubikoitzagalarazi=apustuDA.getApustubikoitzagalarazi();
906         Registered user= apustuDA.getUser();
907         double balioa= apustuDA.getBalioa();
908         for(final Quote quo: apustuDA.getQuote()) {
909             final Quote kuote = db.find(Quote.class, quo.getQuoteNumber());
910             final Apustua ap = new Apustua(apustuAnitza, kuote);
911             db.persist(ap);
912             apustuAnitza.addApustua(ap);
913             kuote.addApustua(ap);
914         }
915         db.getTransaction().commit();
916         db.beginTransaction().begin();
917         if(apustubikoitzagalarazi==-1) {
918             apustubikoitzagalarazi=apustuAnitza.getApustuAnitzaNumber();
919         }
920         apustuAnitza.setApustukopia(apustubikoitzagalarazi);
921         user.updateDirukontua(-balioa);
922         return apustubikoitzagalarazi;
923     }
924 }
```

Sortutako klasea: ApustuDataAccess

```
5
6     public class ApustuDataAccess {
7         private Vector<Quote> quote = new Vector<Quote>();
8         private double balioa;
9         private int apustubikoitzagalarazi;
10        private Registered user;
11        private ApustuAnitza apustuAnitza;
12
13
14        public ApustuDataAccess(Vector<Quote> quote, double balioa,int apustubikoitzagalarazi,
15        Registered user,ApustuAnitza apustuAnitza) {
16            this.quote=quote;
17            this.balioa=balioa;
18            this.apustubikoitzagalarazi=apustubikoitzagalarazi;
19            this.user=user;
20            this.apustuAnitza=apustuAnitza;
21
22
23
24        public Vector<Quote> getQuote() {
25            return quote;
26        }
27
28
29        public void setQuote(Vector<Quote> quote) {
```

Egilea: Helene Astaburuaga Velasco

4.2

Lehen zegoena:

Metodoa: jarraitzaileakBegiratu

```
private void jarraitzaleakBegiratu(final Vector<Quote> quote, final Double balioa, Integer apustuBikoitzaGalarazi,
    final Registered user, final ApustuAnitza apustuAnitza) {
    Boolean b;
    for(final Jarraitzalea reg:user.getJarraitzaleaLista()) {
        final Jarraitzalea erab=db.find(Jarraitzalea.class, reg.getJarraitzaleaNumber());
        b=true;
        for(final ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {
            if(apu.getApustuKopia().equals(apustuAnitza.getApustuKopia())) {
                b=false;
            }
        }
    }
    if(b) {
        if(erab.getNork().getDiruLimitea()<balioa) {
            this.ApustuaEgin(erab.getNork(), quote, erab.getNork().getDiruLimitea(), apustuBikoitzaGalarazi);
        }else{
            this.ApustuaEgin(erab.getNork(), quote, balioa, apustuBikoitzaGalarazi);
        }
    }
}
```

Metodoa: ApustuaEgin

```
 845 b.getTransaction().commit();
846
847
848 c boolean ApustuaEgin(final Registered u, final Vector<Quote> quote, final Double balioa, Integer aja
849 final Registered user = (Registered) db.find(Registered.class, u.getUsername());
850
851 boolean b;
852 if(user.getDirukop()>=balioa) {
853     db.beginTransaction();
854     final ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
855     db.persist(apustuAnitza);
856     ApustuDataAccess apustuDA = new ApustuDataAccess(quote, balioa, apustuBikoitzagaLalarazi, user, apu
857     apustuBikoitzagaLalarazi = apustuAnitza(apustuDA);
858
859     final Transaction t = new Transaction(user, balioa, new Date(), APUSTUA_EGIN);
860     user.addApustuAnitza(apustuAnitza);
861     apustuKantitatea(apustuAnitza);
862     user.addTransaction(t);
863     db.persist(t);
864     db.getTransaction().commit();
865     jarraitzaileakBegiratu(quote, balioa, apustuBikoitzagaLalarazi, user, apustuAnitza);
866     return true;
867 } else{
868     return false;
869 }
```

Refactor egin ondoren:

Metodoa: jarraitzaileak Begiratu

```
private void jarraitzaileakBegiratu(JarraitzaileakBegiratuDataAccess jarraitzaileakBegiratuDA) {
    Boolean b;
    for(final Jarraitzailea reg:jarraitzaileakBegiratuDA.getUser().getJarraitzaileLista()) {
        final Jarraitzailea erab=db.find(Jarraitzailea.class, reg.getJarraitzaileaNumber());
        b=true;
        for(final ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {
            if(apu.getApustuKopia().equals(jarraitzaileakBegiratuDA.getApustuAnitza().getApustuKopia())) {
                b=false;
            }
        }
        double balioa=jarraitzaileakBegiratuDA.getBalioa();
        Vector<Quote> quote= jarraitzaileakBegiratuDA.getQuote();
        int apustuBikoitzaGalarazi=jarraitzaileakBegiratuDA.getApustuBikoitzaGalarazi();
        if(b) {
            if(erab.getNork().getDiruLimitea()<balioa) {
                this.ApustuaEgin(erab.getNork(), quote, erab.getNork().getDiruLimitea(), apustuBikoitzaGalarazi);
            }else{
                this.ApustuaEgin(erab.getNork(), quote, balioa, apustuBikoitzaGalarazi);
            }
        }
    }
}
```

Metodoa: ApustuaEgin

```

    public boolean ApustuaEgin(final Registered u, final Vector<Quote> quote, final Double balioa, Integer apustuBikoitzagaizarri) {
        final Registered user = (Registered) db.find(Registered.class, u.getUsername());
        Boolean b;
        if(user.getDirukop()>=balioa) {
            db.beginTransaction().begin();
            final ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
            db.persist(apustuAnitza);
            ApustuDataAccess apustuDA= new ApustuDataAccess(quote, balioa, apustuBikoitzagaizarri, user, apustuBikoitzagaizarri);
            ApustuAnitza apustuAnitza = apustuAnitza(apustuDA);

            final Transaction t = new Transaction(user, balioa, new Date(), APUSTUA_EGIN);
            user.addApustuAnitza(apustuAnitza);
            apustuKantitatea(apustuAnitza);
            user.addTransaction(t);
            db.persist(t);
            db.getTransaction().commit();
            JarraitzaleakBegiratuDataAccess jarraitzaleakBegiratuDA= new JarraitzaleakBegiratuDataAccess(quote, balioa, apustuBikoitzagaizarri, user, apustuAnitza);
            jarraitzaleakBegiratu(jarraitzaleakBegiratuDA);
            return true;
        }else{
            return false;
        }
    }

```

Sortutako klasea:

```

import java.util.Vector;

public class JarraitzaleakBegiratuDataAccess {
    private Vector<Quote> quote = new Vector<Quote>();
    private Double balioa ;
    private Integer apustuBikoitzagaizarri;
    private Registered user;
    private ApustuAnitza apustuAnitza;

    public JarraitzaleakBegiratuDataAccess( Vector<Quote> quote, Double balioa, Integer apustuBikoitzagaizarri, Registered user, ApustuAnitza apustuAnitza) {
        this.quote=quote;
        this.balioa= balioa;
        this.apustuBikoitzagaizarri=apustuBikoitzagaizarri;
        this.user=user;
        this.apustuAnitza=apustuAnitza;
    }

    public Vector<Quote> getQuote() {
        return quote;
    }
}

```

Egilea: Irati Kintana