

Github helbidea: <https://github.com/anderberru/Bets21>

Egileak: Damian Vela, Lander Soriano, Ander Berruezo

-Damian Vela:

"Write short units of code":

Hasierako kodea:

```
private Bet extractedAddBet(double value, Event ev, Registered user, Vector<Quote> quotes) {
    Bet bet;
    Vector<Quote> foundQuotes = new Vector<Quote>();
    double newMoney;
    String quoNums="";

    db.getTransaction().begin();
    for (Quote quo : quotes) {
        Quote quote = db.find(Quote.class, quo.getQuoteNumber());
        foundQuotes.add(quote);
        quoNums+=quote.getQuoteNumber()+" ";
    }

    bet = user.addBet(value, foundQuotes); //apostua erabiltzailean sartu
    newMoney = user.getMoney() - value;
    user.setMoney(newMoney);

    for(Quote quo: quotes) { //apostua kuotetan sartu
        Quote quote = db.find(Quote.class, quo.getQuoteNumber());
        quote.addBet(bet);
    }

    user.addMovement(value, "Bet: -"+value+" on quotes: "+quoNums);

    db.persist(ev); // db.persist(q) not required when CascadeType.PERSIST is added in questions property of Event class
    // @OneToMany(fetch=FetchType.EAGER, cascade=CascadeType.PERSIST)
    db.persist(user);

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

Errefaktoretzeko kodea:

```
private Bet extractedAddBet(double value, Event ev, Registered user, Vector<Quote> quotes) {
    String quoNums="";

    db.getTransaction().begin();

    quoNums = extractedAddBet2(value, user, quotes);

    user.addMovement(value, "Bet: -"+value+" on quotes: "+quoNums);

    db.persist(ev); // db.persist(q) not required when CascadeType.PERSIST is added in questions property of Event class
    // @OneToMany(fetch=FetchType.EAGER, cascade=CascadeType.PERSIST)
    db.persist(user);

    db.getTransaction().commit();
    return this.bet1;
}
```

```

private String extractedAddBet2(double value, Registered user, Vector<Quote> quotes) {
    double newMoney;
    Vector<Quote> foundQuotes = new Vector<Quote>();
    String quoNums="";

    for (Quote quo : quotes) {
        Quote quote = db.find(Quote.class, quo.getQuoteNumber());
        foundQuotes.add(quote);
        quoNums+=quote.getQuoteNumber()+" ";
    }

    this.bet1 = user.addBet(value, foundQuotes); //apostua erabiltzailean sartu
    newMoney = user.getMoney() - value;
    user.setMoney(newMoney);

    for(Quote quo: quotes) { //apostua kuotetan sartu
        Quote quote = db.find(Quote.class, quo.getQuoteNumber());
        quote.addBet(this.bet1);
    }

    return quoNums;
}

```

ExtractedAddbet metodoa luzegia zenez bi metodoetan banandu dut, lehenengo metodoak bigarrengo metodoari balio batzuk pasatuz. ExtractedAddbet2 metodoak betak erabiltzaileari eta kuotetari sartzen dizkio.

"Write simple units of code":

Hasierako kodea:

```

private void removeBetsFromUsers(Vector<Question> questions) {
    Vector<Quote> quotes;
    Vector<Bet> bets;

    for (Question q : questions) {

        Question question = db.find(Question.class, q.getQuestionNumber());
        quotes=question.getQuotes();
        for (Quote quo : quotes) {

            Quote quote = db.find(Quote.class, quo.getQuoteNumber());
            bets=quote.getBets();
            for (Bet b : bets) {

                Bet bet = db.find(Bet.class, b.getBetNumber());
                String username = bet.getRegistered().getUserName();
                Registered user = db.find(Registered.class, username);
                user.setMoney(user.getMoney() + bet.getValue());
                user.removeBet(bet);

                user.addMovement(bet.getValue(), "Bet removed: "+bet.getValue());
                db.persist(user);
            }
        }
    }
}

```

Errefraktorizatutako kodea:

```

private void removeBetsFromUsers(Vector<Question> questions) {
    Vector<Quote> quotes;
    Vector<Bet> bets;

    for (Question q : questions) {

        Question question = db.find(Question.class, q.getQuestionNumber());
        quotes=question.getQuotes();
        for (Quote quo : quotes) {

            Quote quote = db.find(Quote.class, quo.getQuoteNumber());
            bets=quote.getBets();

            removeBets(bets);
        }
    }
}

```

```

private void removeBets(Vector<Bet> bets) {

    for (Bet b : bets) {

        Bet bet = db.find(Bet.class, b.getBetNumber());
        String username = bet.getRegistered().getUserName();
        Registered user = db.find(Registered.class, username);
        user.setMoney(user.getMoney() + bet.getValue());
        user.removeBet(bet);

        user.addMovement(bet.getValue(), "Bet removed: +" + bet.getValue());
        db.persist(user);
    }
}

```

RemoveBetsFromUsers luzegia eta klomplesuegia zen, egin degunarekin for bat kendu eta kode zati bat beste metodo batean jarri dugu, horrela simpleagoa gelditzen da eta ez 3 for bata bestearen atzetik

"Duplicate code":

Hasierako kodea:

```

else if (Locale.getDefault().equals(new Locale("en"))) {
    q1=evl.addQuestion("Who will win the match?",1);
    q2=evl.addQuestion("Who will score first?",2);
    q3=evl1.addQuestion("Who will win the match?",1);
    q4=evl1.addQuestion("How many goals will be scored in the match?",2);
    q5=evl7.addQuestion("Who will win the match?",1);
    q6=evl7.addQuestion("Will there be goals in the first half?",2);
}

```

Errefaktoretzako kodea:

```

else if (Locale.getDefault().equals(new Locale("en"))) {
    q1=evl.addQuestion(s3,1);
    q2=evl.addQuestion("Who will score first?",2);
    q3=evl1.addQuestion(s3,1);
    q4=evl1.addQuestion("How many goals will be scored in the match?",2);
    q5=evl7.addQuestion(s3,1);
    q6=evl7.addQuestion("Will there be goals in the first half?",2);
}

```

DataAccess klasean "Who will win the match?" string-a hainbat aldiz errepikatzen zen, eta horregatik, s3 izeneko aldagai berri bat sortu dugu "Who will win the match?" balioa emanaz eta aldagai hori erabiliz, string berdina berrerabili beharrean.

"Keep unit interfaces small":

Hasierako kodea:

```

private void followersTratatu(double value, Event event, Set<Quote> selectedQuotes, Vector<Registered> alreadyBet,
    Vector<Registered> followers) throws BetOnSameQuote, NotEnoughMoney {

    for(Registered fol: followers) {
        if(!alreadyBet.contains(fol)) {
            addBet(value, event, selectedQuotes, fol.getUserName(), alreadyBet);
        }
    }
}

```

Ez det aurkitu nola jaitsi parametroak metodo honetan (followersTratatu) baina gelditzen zen metodo bakarra da.

-Lander Soriano:

"Write short units of code":

Hasierako kodea:

```
514 public void putResults(Event evi, String eventResult, Set<Quote> selectedQuotes) throws EventAlreadyRemoved, EventResultsAlreadyIn {
515     Event ev=db.find(Event.class, evi.getEventNumber());
516
517     if (ev == null) {
518         throw new EventAlreadyRemoved();
519     } else if (ev.getResult() != null) {
520         throw new EventResultsAlreadyIn();
521     }
522
523     db.getTransaction().begin();
524
525     ev.setResult(eventResult);
526
527     Vector<Quote> selected = new Vector<Quote>();
528     selected.addAll(selectedQuotes);
529
530     Vector<Question> questions = ev.getQuestions();
531     Vector<Quote> quotes;
532     Vector<Bet> bets;
533
534     for (Question q : questions) {
535
536         Question question = db.find(Question.class, q.getQuestionNumber());
537         quotes=question.getQuotes();
538         for (Quote quo : quotes) {
539
540             Quote quote = db.find(Quote.class, quo.getQuoteNumber());
541             bets=quote.getBets();
542             for (Bet b : bets) {
543
544                 Bet bet = db.find(Bet.class, b.getBetNumber());
545                 String username = bet.getRegistered().getUserName();
546                 Registered user = db.find(Registered.class, username);
547
548
549                 if (selected.contains(quote)) {
550                     double betValue = bet.getValue();
551                     user.setMoney(user.getMoney() + betValue*quote.getValue() + betValue*user.getBonus());
552                     user.addMovement(bet.getValue()*quote.getValue(), "Won bet: +" +bet.getValue()*quote.getValue()*user.getBonus());
553                 } else {
554                     user.addMovement(0, "Lost bet");
555                 }
556
557                 user.removeBet(bet);
558
559                 db.persist(user);
560             }
561             quote.removeAllBets();
562         }
563     }
564
565     db.getTransaction().commit();
566 }
567
```

Errefaktoretzeko kodea:

```
515 public void putResults(Event ev, String eventResult, Set<Quote> selectedQuotes) throws EventAlreadyRemoved, EventResultsAlreadyIn {
516     Event existingEvent = getExistingEvent(ev);
517
518     if (existingEvent.getResult() != null) {
519         throw new EventResultsAlreadyIn();
520     }
521
522     updateEventResult(existingEvent, eventResult);
523     processSelectedQuotes(existingEvent, selectedQuotes);
524 }
525
526 private Event getExistingEvent(Event event) throws EventAlreadyRemoved {
527     Event existingEvent = db.find(Event.class, event.getEventNumber());
528
529     if (existingEvent == null) {
530         throw new EventAlreadyRemoved();
531     }
532
533     return existingEvent;
534 }
535
```

```

536 private void updateEventResult(Event event, String eventResult) {
537     db.getTransaction().begin();
538     event.setResult(eventResult);
539     db.getTransaction().commit();
540 }
541
542 private void processSelectedQuotes(Event event, Set<Quote> selectedQuotes) {
543     for (Question question : event.getQuestions()) {
544         for (Quote quote : question.getQuotes()) {
545             Quote existingQuote = db.find(Quote.class, quote.getQuoteNumber());
546             for (Bet bet : existingQuote.getBets()) {
547                 processBetResult(existingQuote, bet, selectedQuotes);
548             }
549             existingQuote.removeAllBets();
550         }
551     }
552 }
553
554 private void processBetResult(Quote quote, Bet bet, Set<Quote> selectedQuotes) {
555     Bet existingBet = db.find(Bet.class, bet.getBetNumber());
556     String username = existingBet.getRegistered().getUserName();
557     Registered user = db.find(Registered.class, username);
558
559     if (selectedQuotes.contains(quote)) {
560         double betValue = existingBet.getValue();
561         user.setMoney(user.getMoney() + betValue * quote.getValue() + betValue * user.getBonus());
562         user.addMovement(betValue * quote.getValue(), "Won bet: +" + betValue * quote.getValue() * user.getBonus());
563     } else {
564         user.addMovement(0, "Lost bet");
565     }
566
567     user.removeBet(existingBet);
568     db.persist(user);
569 }
570

```

PutResults metodoa lerro asko zituenez, 5 metodo txikiagotan banandu egin da. PutResults berria sarrerako parametroak hartzen ditu eta beste metodo berriei birbideratzen ditu. GetExistingEvent egiaztatzen du ekitaldia existitzen dela, bestela salbuespena jaurtitzen du, updateEventResult ekitaldien emaitzak eguneratzen ditu. Bukatzeko, processSelectedQuotes datuak prestatzen ditu, processBetResult exekuzioa amaitzeko.

"Write simple units of code":

Hasierako kodea:

```

214 public int getSuccessAmount() {
215     int sum = 0;
216     for (Movement m: movements) {
217         if (m.getDescription().contains("Won bet")) sum++;
218     }
219     return sum;
220 }
221
222 public double getSuccessRate() {
223     double suc = 0;
224     double los = 0;
225     for (Movement m: movements) {
226         if (m.getDescription().contains("Won bet")) suc++;
227         else if (m.getDescription().contains("Lost bet")) los++;
228     }
229     if (suc + los == 0.0) return 0;
230     else return Math.round((suc/(suc+los))*1000)/1000.0;
231 }
232
233 }

```

Errefaktoretutako kodea:

```
214 public int getSuccessAmount() {
215     int sum = 0;
216     for (Movement m: movements) {
217         if (m.getDescription().contains("Won bet")) sum++;
218     }
219     return sum;
220 }
221
222 public int getFailureAmount() {
223     int sum = 0;
224     for (Movement m: movements) {
225         if (m.getDescription().contains("Lost bet")) sum++;
226     }
227     return sum;
228 }
229
230 public double getSuccessRate() {
231     double suc = getSuccessAmount();
232     double los = getFailureAmount();
233     |
234     if (suc + los == 0.0) return 0;
235     else return Math.round((suc/(suc+los))*1000)/1000.0;
236 }
237
238 }
```

Registered klasean getSuccessAmount agenda zegoen, baina ez zen erabiltzen getSuccessRate metodoan. Horregatik, getSuccessRate eraldatu ditu getSuccessAmount erabiltzeko eta getFailureAmount sortu dut dinamika berdina jarraitzeko.

"Duplicate code":

Hasierako kodea:

```
125     else {
126         q1=ev1.addQuestion("Zeinek irabaziko du partidua?",1);
127         q2=ev1.addQuestion("Zeinek sartuko du lehenengo gola?",2);
128         q3=ev11.addQuestion("Zeinek irabaziko du partidua?",1);
129         q4=ev11.addQuestion("Zenbat gol sartuko dira?",2);
130         q5=ev17.addQuestion("Zeinek irabaziko du partidua?",1);
131         q6=ev17.addQuestion("Golak sartuko dira lehenengo zatian?",2);
132     }
```

Errefaktoretutako kodea:

```
44     String s2 = "Zeinek irabaziko du partidua?";
126     else {
127         q1=ev1.addQuestion(s2,1);
128         q2=ev1.addQuestion("Zeinek sartuko du lehenengo gola?",2);
129         q3=ev11.addQuestion(s2,1);
130         q4=ev11.addQuestion("Zenbat gol sartuko dira?",2);
131         q5=ev17.addQuestion(s2,1);
132         q6=ev17.addQuestion("Golak sartuko dira lehenengo zatian?",2);
133     }
134 }
```

Datu-basea abiaraztean, behin baino gehiagotan erabiltzen da "Nork irabaziko du partidua?" esaldia. Horregatik, hobe da aldagai batean sartzea, etorkizunean aldaketak egin behar badira ere.

"Keep unit interfaces small":

Hasierako kodea:

```
449 private Bet extractedAddBet(double value, String quoNums, Event ev, Registered user, Vector<Quote> quotes,
450 Vector<Quote> foundQuotes, double newMoney) {
451     Bet bet;
452     db.getTransaction().begin();
453
454     for (Quote quo : quotes) {
455         Quote quote = db.find(Quote.class, quo.getQuoteNumber());
456         foundQuotes.add(quote);
457         quoNums+=quote.getQuoteNumber()+" ";
458     }
459
460     bet = user.addBet(value, foundQuotes); //apostua erabiltzailean sartu
461
462     user.setMoney(newMoney);
463
464     for(Quote quo: quotes) { //apostua kuotetan sartu
465         Quote quote = db.find(Quote.class, quo.getQuoteNumber());
466         quote.addBet(bet);
467     }
468
469
470     user.addMovement(value, "Bet: -"+value+" on quotes: "+quoNums);
471
472     db.persist(ev); // db.persist(q) not required when CascadeType.PERSIST is added in questions property of Event class
473     // @OneToMany(fetch=FetchType.EAGER, cascade=CascadeType.PERSIST)
474     db.persist(user);
475
476     db.getTransaction().commit();
477     return bet;
478 }
479 }
```

Errefaktoratutako kodea:

```
449 private Bet extractedAddBet(double value, Event ev, Registered user, Vector<Quote> foundQuotes) {
450     Bet bet;
451     Vector<Quote> quotes = new Vector<Quote>();
452     double newMoney;
453     String quoNums="";
454
455     db.getTransaction().begin();
456
457     for (Quote quo : quotes) {
458         Quote quote = db.find(Quote.class, quo.getQuoteNumber());
459         foundQuotes.add(quote);
460         quoNums+=quote.getQuoteNumber()+" ";
461     }
462
463     bet = user.addBet(value, foundQuotes); //apostua erabiltzailean sartu
464     newMoney = user.getMoney() - value;
465     user.setMoney(newMoney);
466
467     for(Quote quo: quotes) { //apostua kuotetan sartu
468         Quote quote = db.find(Quote.class, quo.getQuoteNumber());
469         quote.addBet(bet);
470     }
471
472
473     user.addMovement(value, "Bet: -"+value+" on quotes: "+quoNums);
474
475     db.persist(ev); // db.persist(q) not required when CascadeType.PERSIST is added in questions property of Event class
476     // @OneToMany(fetch=FetchType.EAGER, cascade=CascadeType.PERSIST)
477     db.persist(user);
478
479     db.getTransaction().commit();
480     return bet;
481 }
482 }
```

ExtractedAddBet 7 parametro zituen, horietatik 2 parametro hutsak ziren, eta beste batek edukia ateratzen zuen beste bi parametroekin egindako eragiketa batetik. ErrefaktORIZAZIOA egin ondoren quotes eta quoNums metodo bertan sortzen dira eta newMoney metodoan kalkulatzeko da.

-Ander Berruezo:

"Write short units of code":

Hasierako kodea:

```
410 public Bet addBet(double value, Event event, Set<Quote> selectedQuotes, String username, Vector<Registered> alreadyBet) throws BetOnSameQuote, NotEnoughMoney {
411     String quoNums="";
412     Event ev = db.find(Event.class, event.getEventNumber());
413
414     Registered user = db.find(Registered.class, username);
415
416     Bet bet=null;
417
418     alreadyBet.add(user); //bisitatu bezala markatu
419
420     Vector<Quote> quotes = new Vector<Quote>();
421     quotes.addAll(selectedQuotes);
422
423     Vector<Quote> foundQuotes = new Vector<Quote>();
424
425
426
427     if (user.betOnSameQuote(quotes)) throw new BetOnSameQuote();
428
429
430
431     double newMoney = user.getMoney() - value;
432     if (newMoney < 0 && alreadyBet.size()-1) {
433         throw new NotEnoughMoney();
434     } else if (newMoney>=0) {
435
436         db.getTransaction().begin();
437
438         for (Quote quo : quotes) {
439             Quote quote = db.find(Quote.class, quo.getQuoteNumber());
440             foundQuotes.add(quote);
441             quoNums+=quote.getQuoteNumber()+" ";
442         }
443
444         bet = user.addBet(value, foundQuotes); //zerotua azalduzailan sartu
445
446         user.setMoney(newMoney);
447
448         for(Quote quo: quotes) { //zerotua azalduzailan sartu
449             Quote quote = db.find(Quote.class, quo.getQuoteNumber());
450             quote.addBet(bet);
451         }
452
453
454         user.addMovement(value, "Bet: -"+value+" on quotes: "+quoNums);
455
456         db.persist(ev); // db.persist(q) not required when CascadeType.PERSIST is added in questions property of Event class
457         // @OneToMany(fetch=FetchType.EAGER, cascade=CascadeType.PERSIST)
458         db.persist(user);
459
460         db.getTransaction().commit();
461
462         Vector<Registered> followers = user.getFollowers();
463
464         for(Registered fol: followers) {
465             if(!alreadyBet.contains(fol)) {
466                 addBet(value, event, selectedQuotes, fol.getUserName(), alreadyBet);
467             }
468         }
469     }
470     return bet;
471 }
472
473 }
```

Errefaktoretzako kodea:

```
413* public Bet addBet(double value, Event event, Set<Quote> selectedQuotes, String username, Vector<Registered> alreadyBet) throws BetOnSameQuote, NotEnoughMoney {
414     String quoNums="";
415     Event ev = db.find(Event.class, event.getEventNumber());
416
417     Registered user = db.find(Registered.class, username);
418
419     Bet bet=null;
420
421     alreadyBet.add(user); //bisitatu bezala markatu
422
423     Vector<Quote> quotes = new Vector<Quote>();
424     quotes.addAll(selectedQuotes);
425
426     if (user.betOnSameQuote(quotes)) throw new BetOnSameQuote();
427
428     double newMoney = user.getMoney() - value;
429     if (newMoney < 0 && alreadyBet.size()-1) {
430         throw new NotEnoughMoney();
431     } else if (newMoney>=0) {
432
433         bet = extractedAddBet(value, quoNums, ev, user, quotes, new Vector<Quote>(), newMoney);
434
435         followersTratatu(value, event, selectedQuotes, alreadyBet, user.getFollowers());
436     }
437     return bet;
438 }
```

AddBet metodoa lerro gehiegi zituenenez, 15 lerro izateko errefaktoretzatu da. Horretarako, beste bi metodo sortu dira *extract Method* aukerarekin, *extractedAddBet* eta *followersTratatu*. *ExtractedAddBet* metodoan datu-basearen aldaketak tratatzen dira, apustu berria sortuz.

FollowersTratatu metodoan begizta bat dago, erabiltzailearen jarraitzaileak tratatzen dituen apustu berdina egiteko, addBet berriro deituz.

"Write simple units of code":

Hasierako kodea:

```
483 public void removeEvent(Event event) {
484     Event ev=db.find(Event.class, event.getEventNumber());
485
486     db.getTransaction().begin();
487
488     Vector<Question> questions = ev.getQuestions();
489     Vector<Quote> quotes;
490     Vector<Bet> bets;
491
492     for (Question q : questions) {
493
494         Question question = db.find(Question.class, q.getQuestionNumber());
495         quotes=question.getQuotes();
496         for (Quote quo : quotes) {
497
498             Quote quote = db.find(Quote.class, quo.getQuoteNumber());
499             bets=quote.getBets();
500             for (Bet b : bets) {
501
502                 Bet bet = db.find(Bet.class, b.getBetNumber());
503                 String username = bet.getRegistered().getUserName();
504                 Registered user = db.find(Registered.class, username);
505                 user.setMoney(user.getMoney() + bet.getValue());
506                 user.removeBet(bet);
507
508                 user.addMovement(bet.getValue(), "Bet removed: "+bet.getValue());
509                 db.persist(user);
510             }
511         }
512     }
513
514     db.remove(ev);
515
516     db.getTransaction().commit();
517 }
```

ErrefaktORIZATUTAKO kodea:

```
483 public void removeEvent(Event event) {
484     Event ev=db.find(Event.class, event.getEventNumber());
485
486     db.getTransaction().begin();
487
488     Vector<Question> questions = ev.getQuestions();
489     removeBetsFromUsers(questions);
490
491     db.remove(ev);
492
493     db.getTransaction().commit();
494 }
```

```

496● private void removeBetsFromUsers(Vector<Question> questions) {
497     Vector<Quote> quotes;
498     Vector<Bet> bets;
499
500     for (Question q : questions) {
501
502         Question question = db.find(Question.class, q.getQuestionNumber());
503         quotes=question.getQuotes();
504         for (Quote quo : quotes) {
505
506             Quote quote = db.find(Quote.class, quo.getQuoteNumber());
507             bets=quote.getBets();
508             for (Bet b : bets) {
509
510                 Bet bet = db.find(Bet.class, b.getBetNumber());
511                 String username = bet.getRegistered().getUserName();
512                 Registered user = db.find(Registered.class, username);
513                 user.setMoney(user.getMoney() + bet.getValue());
514                 user.removeBet(bet);
515
516                 user.addMovement(bet.getValue(), "Bet removed: +" + bet.getValue());
517                 db.persist(user);
518             }
519         }
520     }
521 }

```

Metodo hau bi eginkizun zituen, bi metodotan banatu da, removeBetsFromUsers sortuz. Metodo berrian ezabatuko den gertaeraren galdera guztien, kuota guztien apustu guztiak bere erabiltzailetatik ezabatzen dira. Metodo nagusian, metodo berria deitzen da eta ondoren gertaera datu-basetik ezabatzen da. Horrela metodo bakoitzak gauza bakarra egiten du.

“Duplicate code”:

Hasierako kodea:

```

183● public Question createQuestion(Event event, String question, float betMinimum) throws QuestionAlreadyExist {
184     System.out.println(">>> DataAccess: createQuestion=> event= "+event+" question= "+question+" betMinimum="+betMinimum);
185
186     Event ev = db.find(Event.class, event.getEventNumber());
187
188     if (ev.DoesQuestionExists(question)) throw new QuestionAlreadyExist(ResourceBundle.getBundle("Etiquetas").getString("QueryAlreadyExist"));
189
190     db.getTransaction().begin();
191     Question q = ev.addQuestion(question, betMinimum);
192     //db.persist(q);
193     db.persist(ev); // db.persist(q) not required when CascadeType.PERSIST is added in questions property of Event class
194     // @OneToMany(fetch=FetchType.EAGER, cascade=CascadeType.PERSIST)
195     db.getTransaction().commit();
196     return q;
197
198 }

```

```

209● public Event createEvent(String description, Date eventDate) throws EventAlreadyExist {
210
211     Event ev = new Event(description, eventDate);
212     Vector<Event> events = this.getEvents(eventDate);
213
214     if (events.contains(new Event(description, eventDate))) throw new EventAlreadyExist(ResourceBundle.getBundle("Etiquetas").getString("ErrorEventAlreadyExist"));
215
216     db.getTransaction().begin();
217
218     db.persist(ev);
219     db.getTransaction().commit();
220     return ev;
221 }
222

```

Errefaktoretzeko kodea:

```
39 public class DataAccess {
40     protected static EntityManager db;
41     protected static EntityManagerFactory emf;
42
43     String s1 = "¿Quién ganará el partido?";
44     String etiqueta = "Etiquetas";
45
46     public Question createQuestion(Event event, String question, float betMinimum) throws QuestionAlreadyExist {
47         System.out.println(">> DataAccess: createQuestion=> event= "+event+" question= "+question+" betMinimum="+betMinimum);
48
49         Event ev = db.find(Event.class, event.getEventNumber());
50
51         if (ev.DoesQuestionExists(question)) throw new QuestionAlreadyExist(ResourceBundle.getBundle(etiqueta).getString("QueryAlreadyExist"));
52
53         db.getTransaction().begin();
54         Question q = ev.addQuestion(question, betMinimum);
55         //db.persist(q);
56         db.persist(ev); // db.persist(q) not required when CascadeType.PERSIST is added in questions property of Event class
57         // @OneToMany(fetch=FetchType.EAGER, cascade=CascadeType.PERSIST)
58         db.getTransaction().commit();
59         return q;
60     }
61
62     public Event createEvent(String description, Date eventDate) throws EventAlreadyExist {
63         Event ev = new Event(description, eventDate);
64         Vector<Event> events = this.getEvents(eventDate);
65
66         if (events.contains(new Event(description, eventDate))) throw new EventAlreadyExist(ResourceBundle.getBundle(etiqueta).getString("ErrorEventAlreadyExist"));
67
68         db.getTransaction().begin();
69
70         db.persist(ev);
71         db.getTransaction().commit();
72         return ev;
73     }
74 }
```

DataAccess klasean "Etiquetas" string-a hainbat aldiz errepikatzen zen, eta horregatik, *etiqueta* izeneko aldagai berri bat sortu dugu "Etiquetas" balioa emanez eta aldagai hori erabiliz, string berdina berrerabili beharrean.

"Keep unit interfaces small":

Hasierako kodea:

```
352 public boolean register(String username, String pass, String fullname, String DNI, String payMethod, Date date, String email, int money) {
353     try {
354         User register;
355         db.getTransaction().begin();
356
357         register = new Registered(username, pass, fullname, DNI, date, payMethod, email, money);
358
359         db.persist(register);
360         db.getTransaction().commit();
361         System.out.println("Gordeta " + register.getUserName());
362         return true;
363     } catch (Exception e) {
364         return false;
365     }
366 }
```

Errefaktorizatuko kodea:

```
352●    public boolean register(User register) {
353        try {
354            db.getTransaction().begin();
355
356            db.persist(register);
357
358            db.getTransaction().commit();
359            System.out.println("Gordeta " + register.getUserName());
360            return true;
361        } catch (Exception e) {
362            return false;
363        }
364    }
365
```

Register metodoak 8 parametro zituen, 4 baino askoz gehiago zirenak. Konpontzeko, *change method signature* aukera erabiliz User motako registered parametro bakarra izateko aldatu dugu, parametroa zuzenean datu-basean sartzeko. Horrela registered motako objektua dataAccess-en barruan sortu beharrean, metodoa deitu aurretik sortzen da, lehen kodean zeuden parametroak erabiliz.