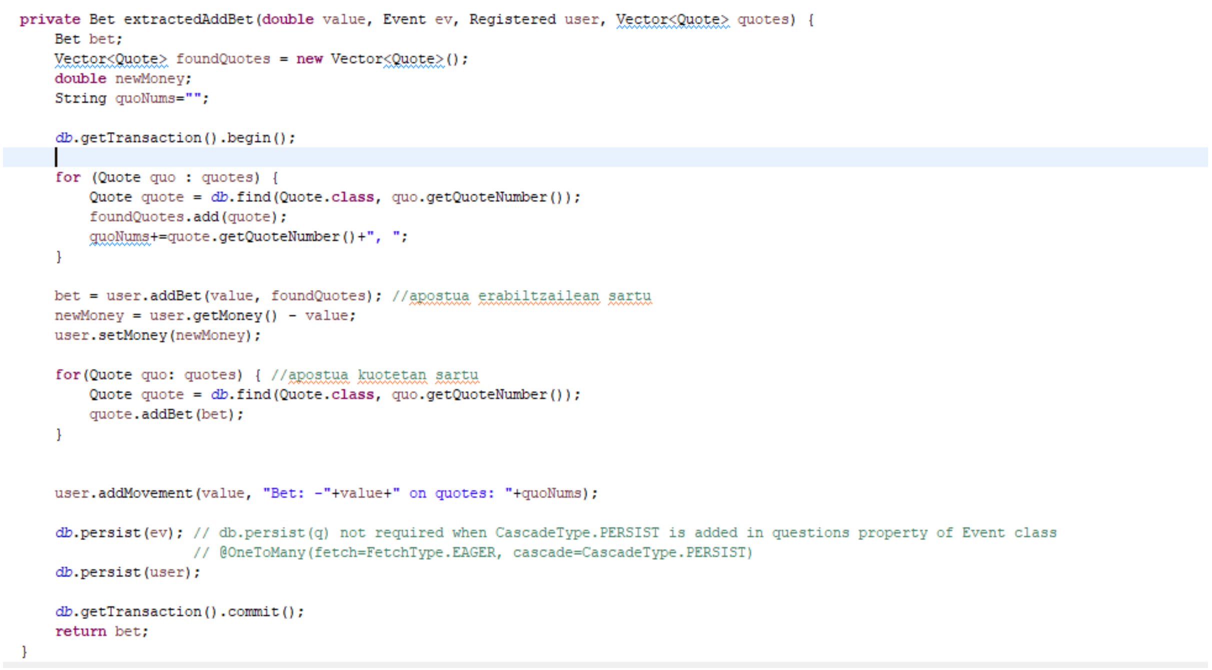
Github helbidea: https://github.com/anderberru/Bets21
Egileak: Damian Vela, Lander Soriano, Ander Berruezo

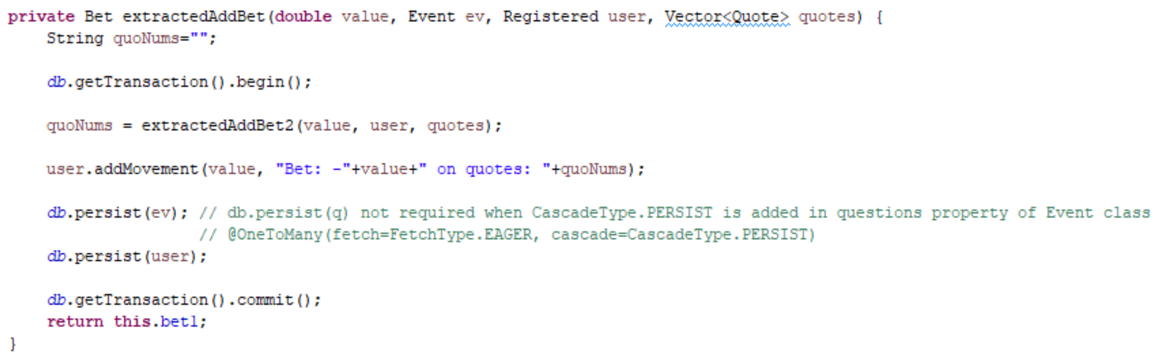
-Damian Vela:
"Write short units of code":

Hasierako kodea:

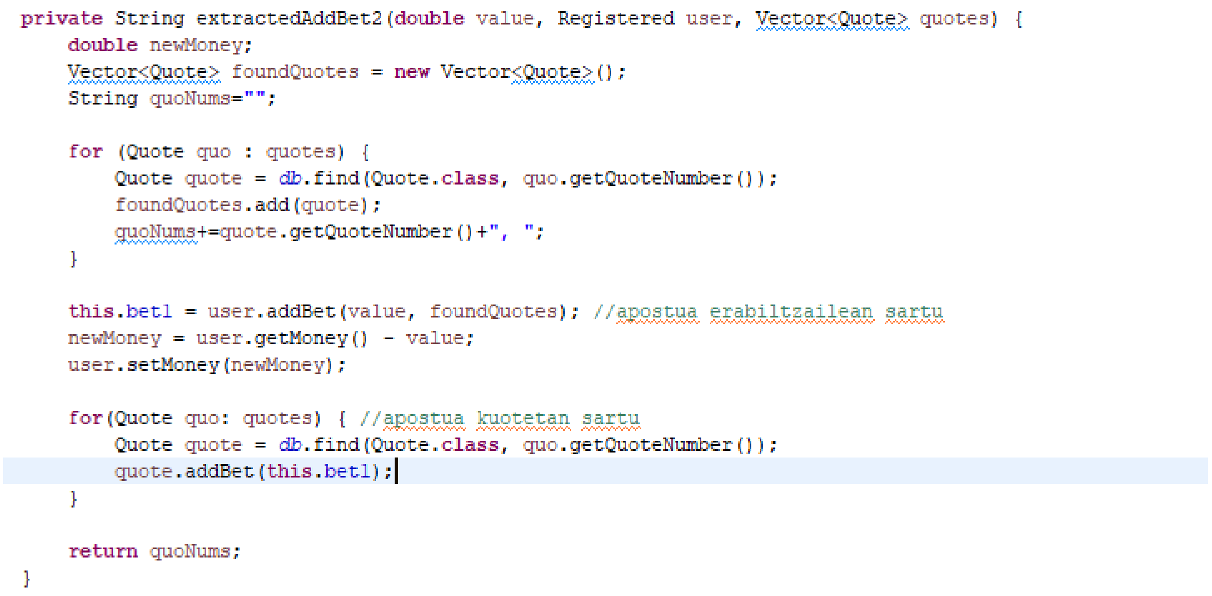


```java
private Bet extractedAddBet(double value, Event ev, Registered user, Vector<Quote> quotes) {
    Bet bet;
    Vector<Quote> foundQuotes = new Vector<Quote>();
    double newMoney;
    String quoNums="";

    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;
}
```

Errefaktorizatuko kodea:



```java
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;
}
```

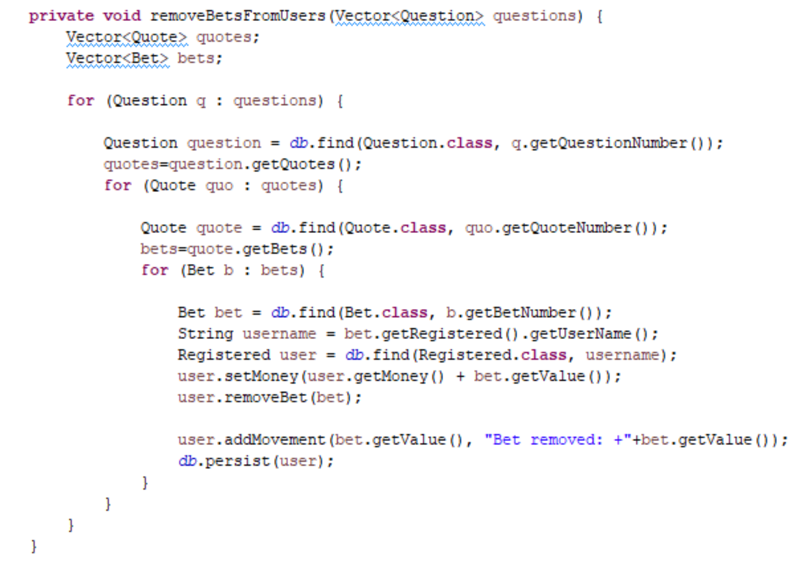


```java
private String extractedAddBet2(double value, Registered user, Vector<Quote> quotes) {
    double newMoney;
    Vector<Quote> foundQuotes = new Vector<Quote>();
    String quoNums="";

    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 quotetari sartzen dizkio.

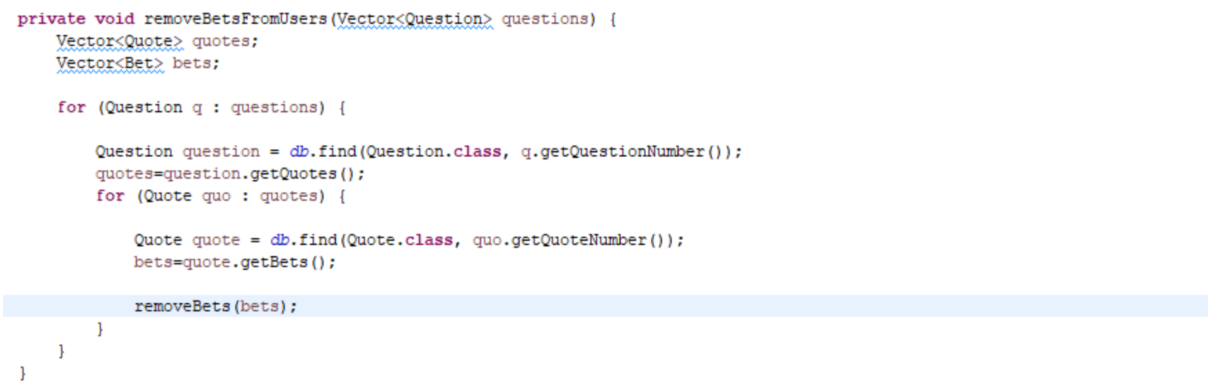
<u>"Write simple units of code":</u>

Hasierako kodea:



```java
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:



```java
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);
        }
    }
}
```

```java
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:

```java
 else if (Locale.getDefault().equals(new Locale("en"))) {
     q1=ev1.addQuestion("Who will win the match?",1);
     q2=ev1.addQuestion("Who will score first?",2);
     q3=ev11.addQuestion("Who will win the match?",1);
     q4=ev11.addQuestion("How many goals will be scored in the match?",2);
     q5=ev17.addQuestion("Who will win the match?",1);
     q6=ev17.addQuestion("Will there be goals in the first half?",2);
 }
```

Errefaktorizatuko kodea:

```java
else if (Locale.getDefault().equals(new Locale("en"))) {
    q1=ev1.addQuestion(s3,1);
    q2=ev1.addQuestion("Who will score first?",2);
    q3=ev11.addQuestion(s3,1);
    q4=ev11.addQuestion("How many goals will be scored in the match?",2);
    q5=ev17.addQuestion(s3,1);
    q6=ev17.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 emanez eta aldagai hori erabiliz, string berdina berrerabili beharrean.

"Keep unit interfaces small":

Hasierako kodea:

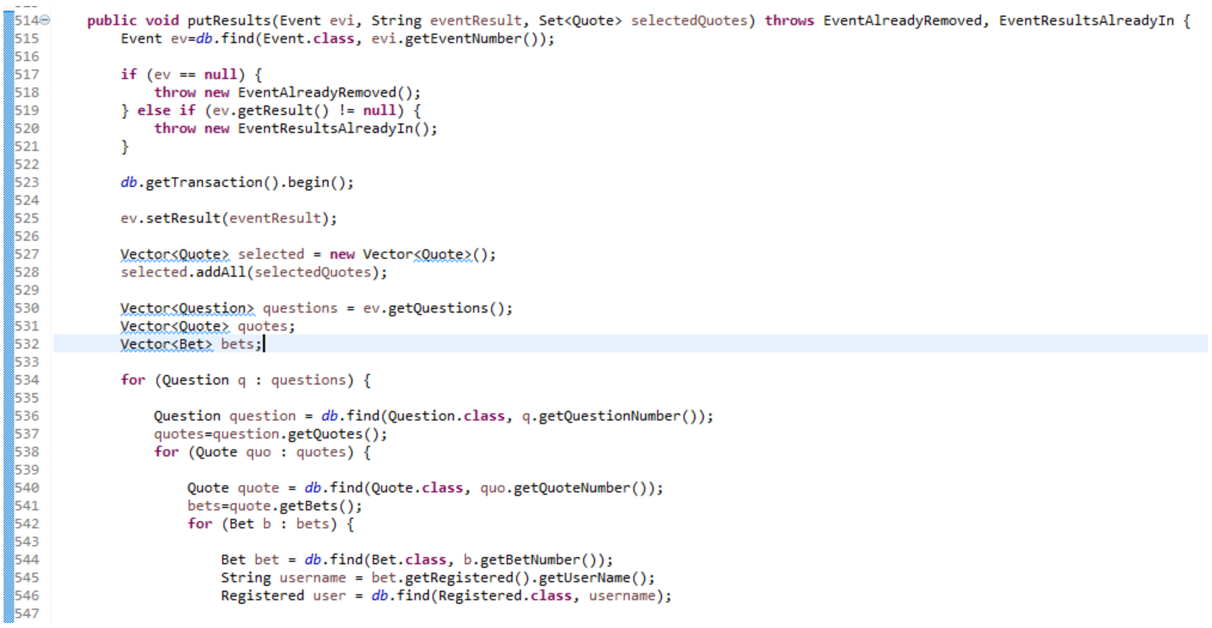
```java
private void followersTratatu(double value, Event event, Set<Quote> selectedQuotes, Vector<Registered> alreadyBet,
        Vector<Registered> followers) throws BetOnSameQuote, NotEnoughMoney {

    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.

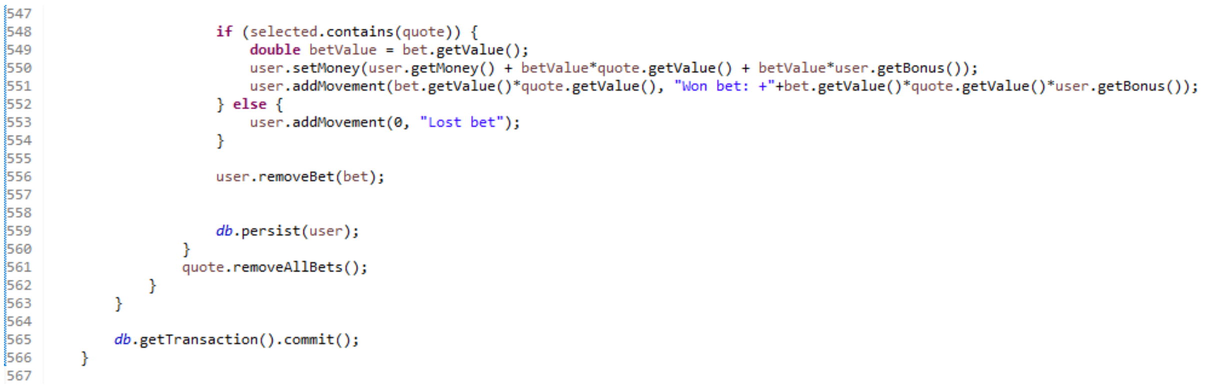
-Lander Soriano:

<u>"Write short units of code":</u>

Hasierako kodea:

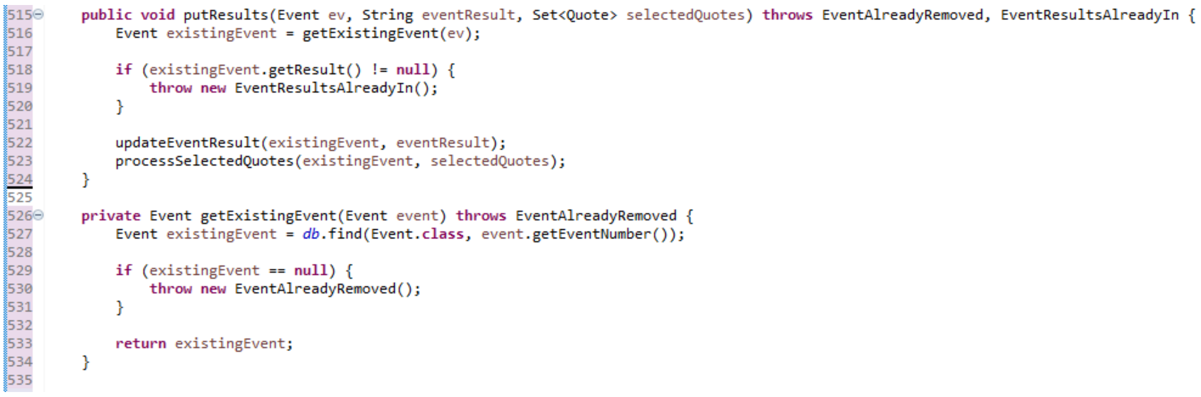


```java
public void putResults(Event evi, String eventResult, Set<Quote> selectedQuotes) throws EventAlreadyRemoved, EventResultsAlreadyIn {
    Event ev=db.find(Event.class, evi.getEventNumber());

    if (ev == null) {
        throw new EventAlreadyRemoved();
    } else if (ev.getResult() != null) {
        throw new EventResultsAlreadyIn();
    }

    db.getTransaction().begin();

    ev.setResult(eventResult);

    Vector<Quote> selected = new Vector<Quote>();
    selected.addAll(selectedQuotes);

    Vector<Question> questions = ev.getQuestions();
    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);

```

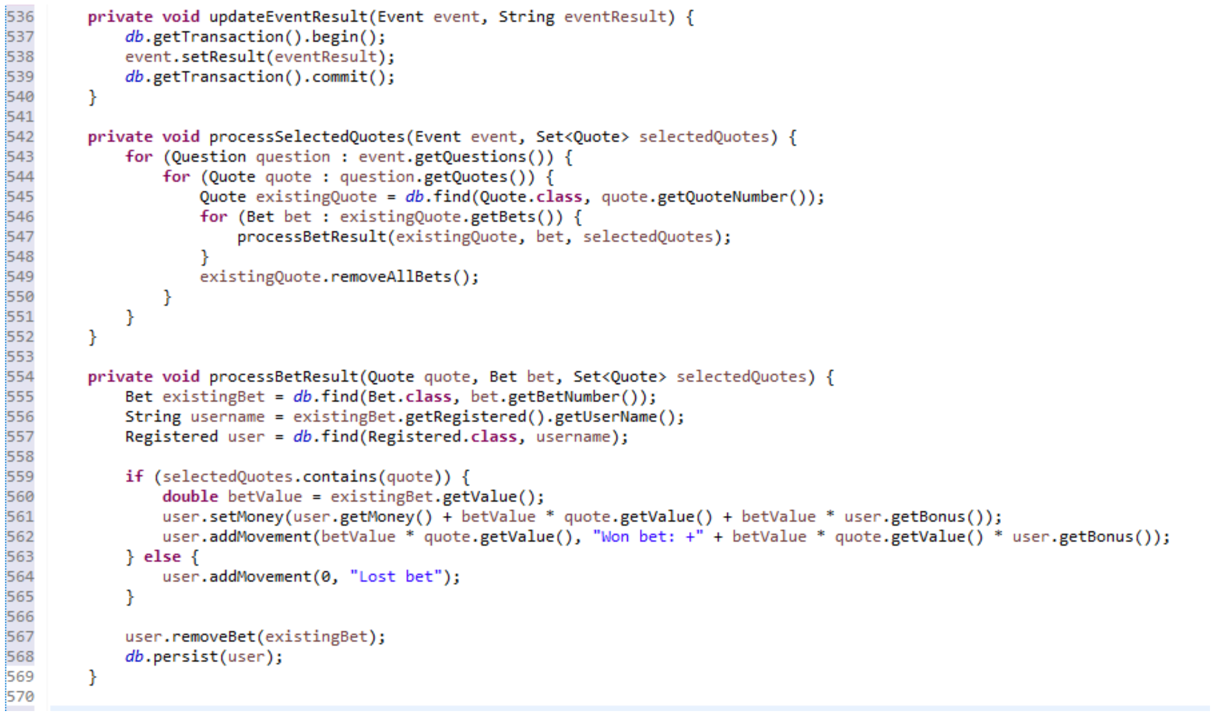


```java

                if (selected.contains(quote)) {
                    double betValue = bet.getValue();
                    user.setMoney(user.getMoney() + betValue*quote.getValue() + betValue*user.getBonus());
                    user.addMovement(bet.getValue()*quote.getValue(), "Won bet: +"+bet.getValue()*quote.getValue()*user.getBonus());
                } else {
                    user.addMovement(0, "Lost bet");
                }

                user.removeBet(bet);


                db.persist(user);
            }
            quote.removeAllBets();
        }
    }

    db.getTransaction().commit();
}

```

Errefaktorizatuko kodea:



```java
public void putResults(Event ev, String eventResult, Set<Quote> selectedQuotes) throws EventAlreadyRemoved, EventResultsAlreadyIn {
    Event existingEvent = getExistingEvent(ev);

    if (existingEvent.getResult() != null) {
        throw new EventResultsAlreadyIn();
    }

    updateEventResult(existingEvent, eventResult);
    processSelectedQuotes(existingEvent, selectedQuotes);
}

private Event getExistingEvent(Event event) throws EventAlreadyRemoved {
    Event existingEvent = db.find(Event.class, event.getEventNumber());

    if (existingEvent == null) {
        throw new EventAlreadyRemoved();
    }

    return existingEvent;
}

```

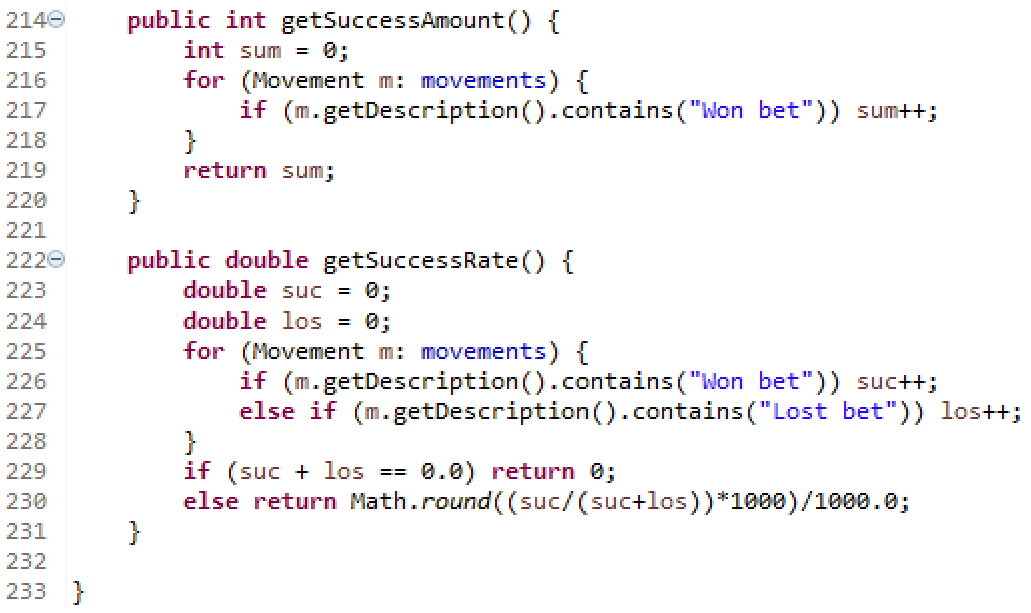


```java
    private void updateEventResult(Event event, String eventResult) {
        db.getTransaction().begin();
        event.setResult(eventResult);
        db.getTransaction().commit();
    }

    private void processSelectedQuotes(Event event, Set<Quote> selectedQuotes) {
        for (Question question : event.getQuestions()) {
            for (Quote quote : question.getQuotes()) {
                Quote existingQuote = db.find(Quote.class, quote.getQuoteNumber());
                for (Bet bet : existingQuote.getBets()) {
                    processBetResult(existingQuote, bet, selectedQuotes);
                }
                existingQuote.removeAllBets();
            }
        }
    }

    private void processBetResult(Quote quote, Bet bet, Set<Quote> selectedQuotes) {
        Bet existingBet = db.find(Bet.class, bet.getBetNumber());
        String username = existingBet.getRegistered().getUserName();
        Registered user = db.find(Registered.class, username);

        if (selectedQuotes.contains(quote)) {
            double betValue = existingBet.getValue();
            user.setMoney(user.getMoney() + betValue * quote.getValue() + betValue * user.getBonus());
            user.addMovement(betValue * quote.getValue(), "Won bet: +" + betValue * quote.getValue() * user.getBonus());
        } else {
            user.addMovement(0, "Lost bet");
        }

        user.removeBet(existingBet);
        db.persist(user);
    }

```

PutResuts 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.

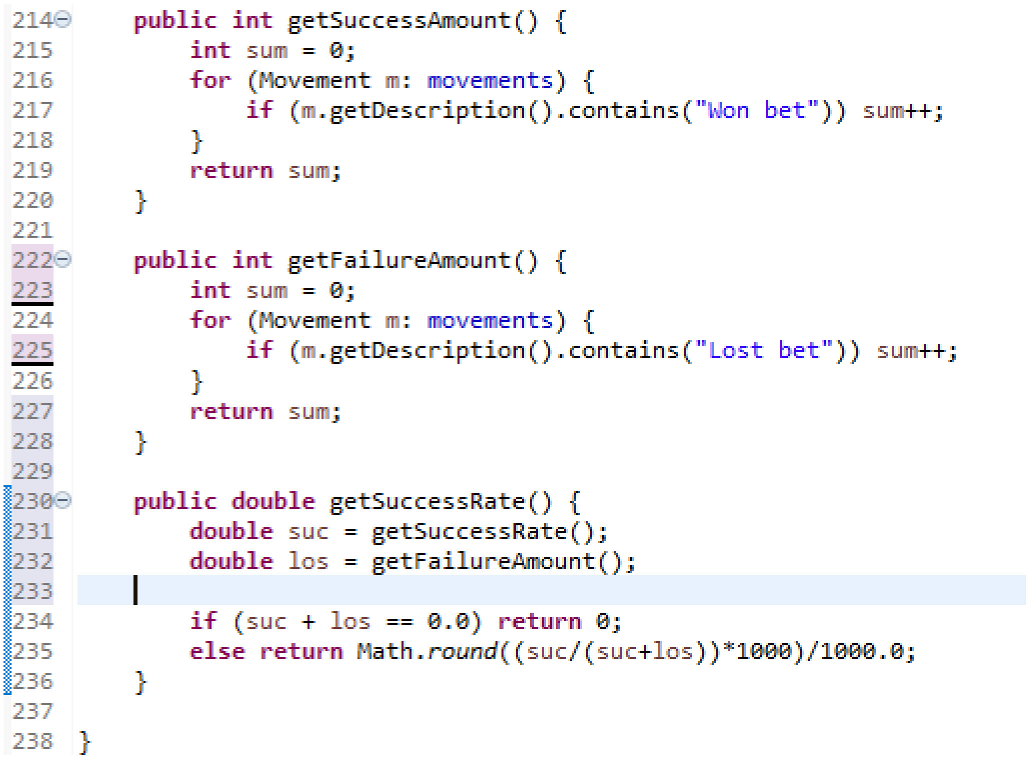
"Write simple units of code":

Hasierako kodea:



```java
    public int getSuccessAmount() {
        int sum = 0;
        for (Movement m: movements) {
            if (m.getDescription().contains("Won bet")) sum++;
        }
        return sum;
    }

    public double getSuccessRate() {
        double suc = 0;
        double los = 0;
        for (Movement m: movements) {
            if (m.getDescription().contains("Won bet")) suc++;
            else if (m.getDescription().contains("Lost bet")) los++;
        }
        if (suc + los == 0.0) return 0;
        else return Math.round((suc/(suc+los))*1000)/1000.0;
    }

}
```

Errefaktorizatutako kodea:

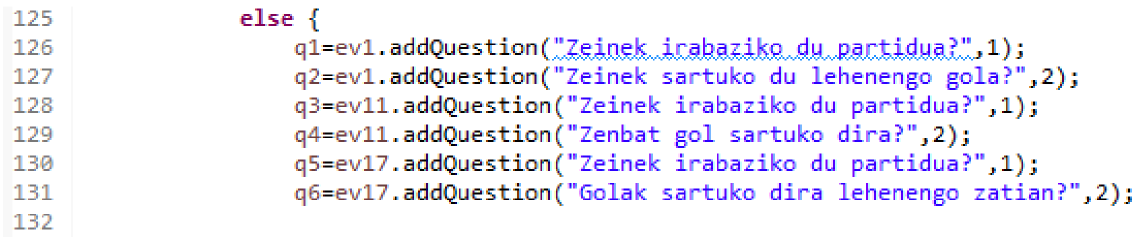


```java
    public int getSuccessAmount() {
        int sum = 0;
        for (Movement m: movements) {
            if (m.getDescription().contains("Won bet")) sum++;
        }
        return sum;
    }

    public int getFailureAmount() {
        int sum = 0;
        for (Movement m: movements) {
            if (m.getDescription().contains("Lost bet")) sum++;
        }
        return sum;
    }

    public double getSuccessRate() {
        double suc = getSuccessRate();
        double los = getFailureAmount();

        if (suc + los == 0.0) return 0;
        else return Math.round((suc/(suc+los))*1000)/1000.0;
    }

}
```

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

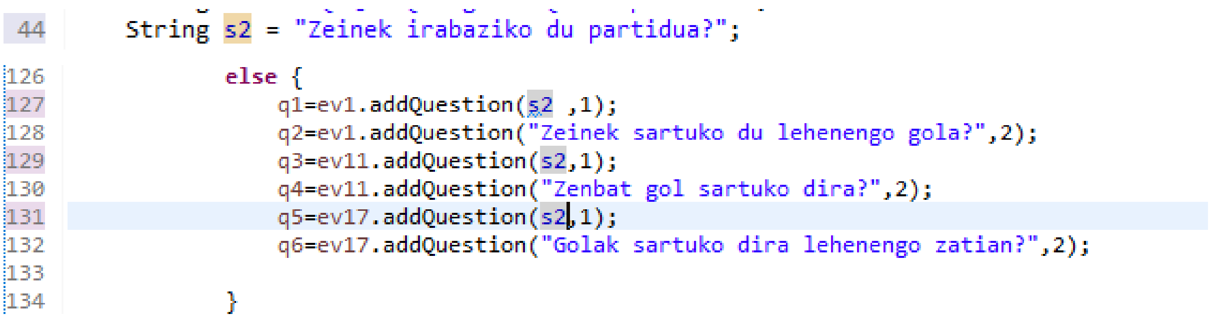
<u>"Duplicate code":</u>

Hasierako kodea:



```java
            else {
                q1=ev1.addQuestion("Zeinek irabaziko du partidua?",1);
                q2=ev1.addQuestion("Zeinek sartuko du lehenengo gola?",2);
                q3=ev11.addQuestion("Zeinek irabaziko du partidua?",1);
                q4=ev11.addQuestion("Zenbat gol sartuko dira?",2);
                q5=ev17.addQuestion("Zeinek irabaziko du partidua?",1);
                q6=ev17.addQuestion("Golak sartuko dira lehenengo zatian?",2);

```

Errefaktorizatutako kodea:



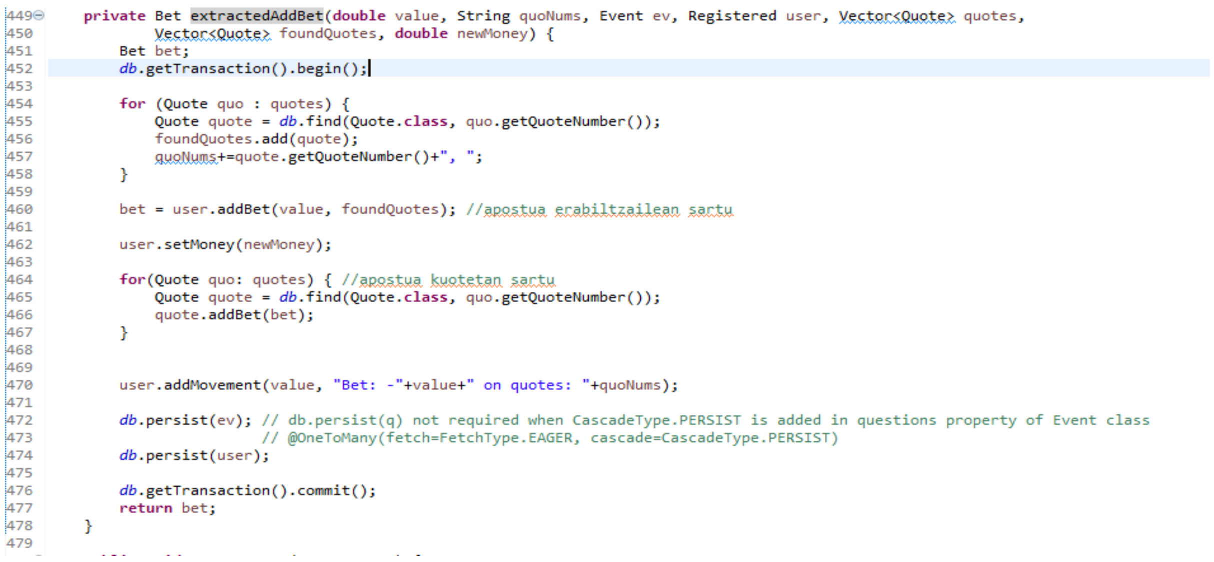
```java
    String s2 = "Zeinek irabaziko du partidua?";
```

```java
            else {
                q1=ev1.addQuestion(s2 ,1);
                q2=ev1.addQuestion("Zeinek sartuko du lehenengo gola?",2);
                q3=ev11.addQuestion(s2,1);
                q4=ev11.addQuestion("Zenbat gol sartuko dira?",2);
                q5=ev17.addQuestion(s2,1);
                q6=ev17.addQuestion("Golak sartuko dira lehenengo zatian?",2);

            }
```

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.

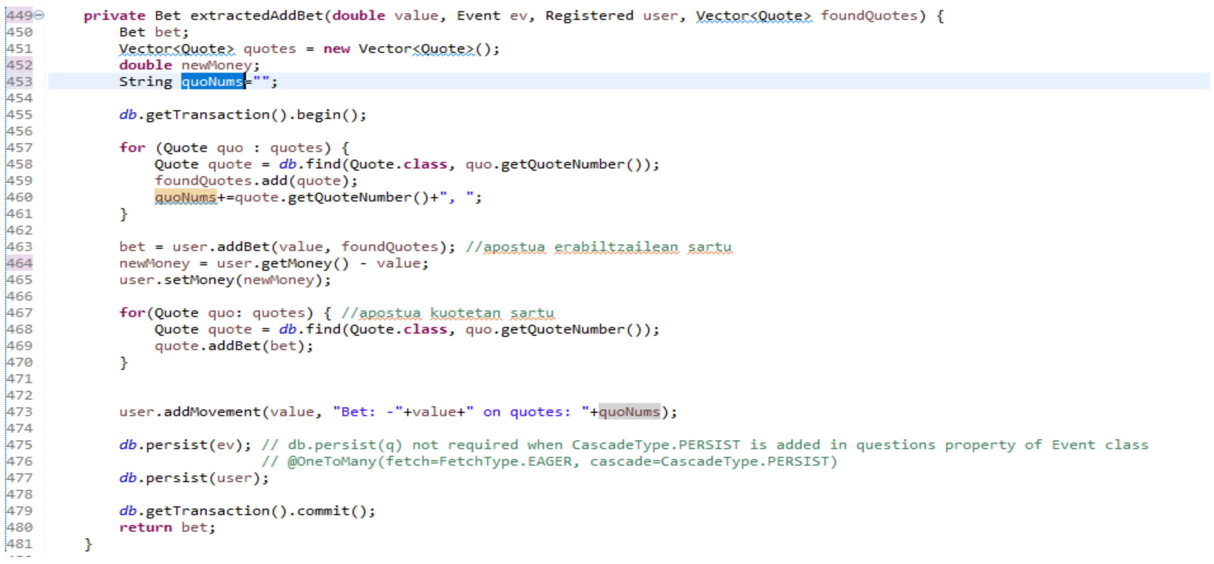
"Keep unit interfaces small":

Hasierako kodea:



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

        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;
    }
```

Errefaktorizatutako kodea:

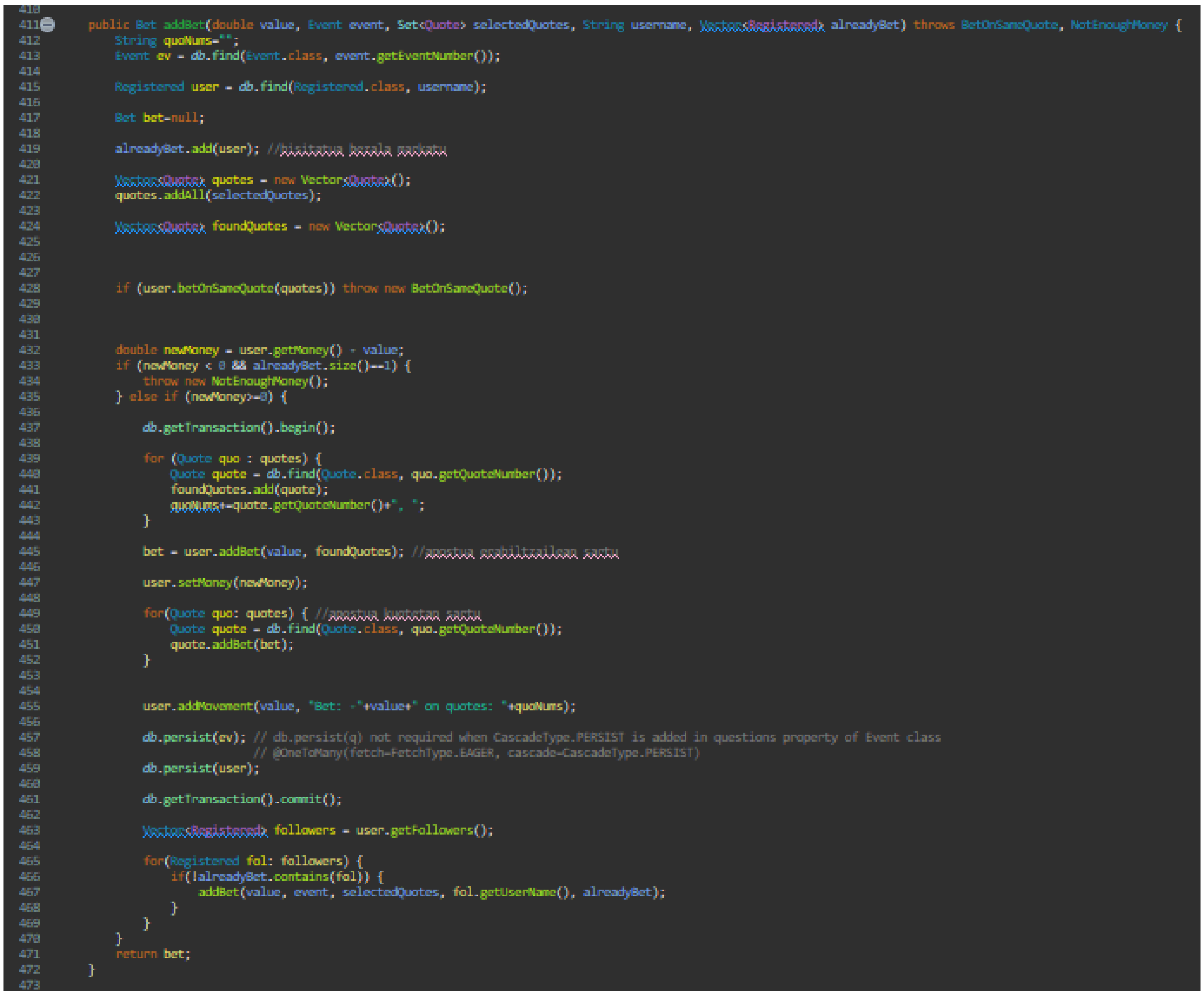


```java
    private Bet extractedAddBet(double value, Event ev, Registered user, Vector<Quote> foundQuotes) {
        Bet bet;
        Vector<Quote> quotes = new Vector<Quote>();
        double newMoney;
        String quoNums="";

        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;
    }
```

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

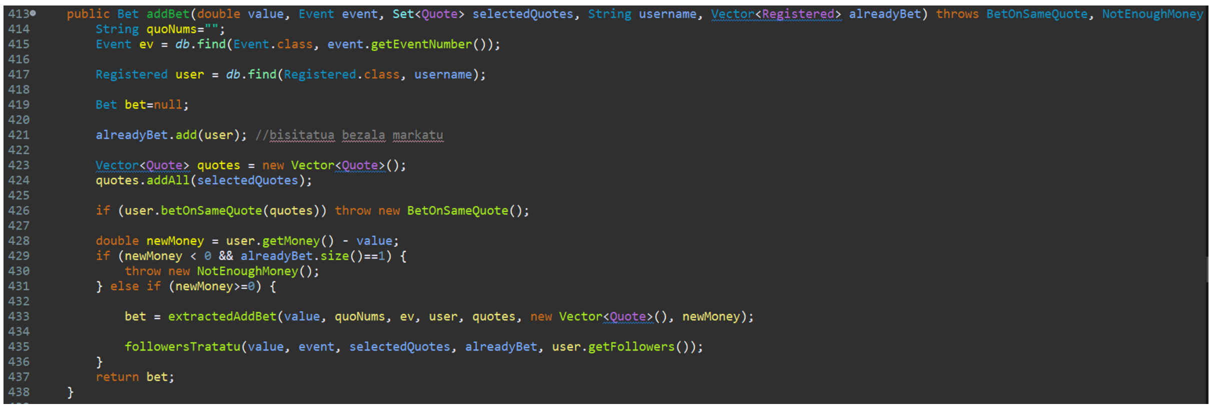
-Ander Berruezo:
"Write short units of code":

Hasierako kodea:



```java
    public Bet addBet(double value, Event event, Set<Quote> selectedQuotes, String username, Vector<Registered> alreadyBet) throws BetOnSameQuote, NotEnoughMoney {
        String quoNums="";
        Event ev = db.find(Event.class, event.getEventNumber());

        Registered user = db.find(Registered.class, username);

        Bet bet=null;

        alreadyBet.add(user); //bisitatua bezala markatu

        Vector<Quote> quotes = new Vector<Quote>();
        quotes.addAll(selectedQuotes);

        Vector<Quote> foundQuotes = new Vector<Quote>();



        if (user.betOnSameQuote(quotes)) throw new BetOnSameQuote();



        double newMoney = user.getMoney() - value;
        if (newMoney < 0 && alreadyBet.size()==1) {
            throw new NotEnoughMoney();
        } else if (newMoney>=0) {

            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

            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();

            Vector<Registered> followers = user.getFollowers();

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

Errefaktorizatuko kodea:



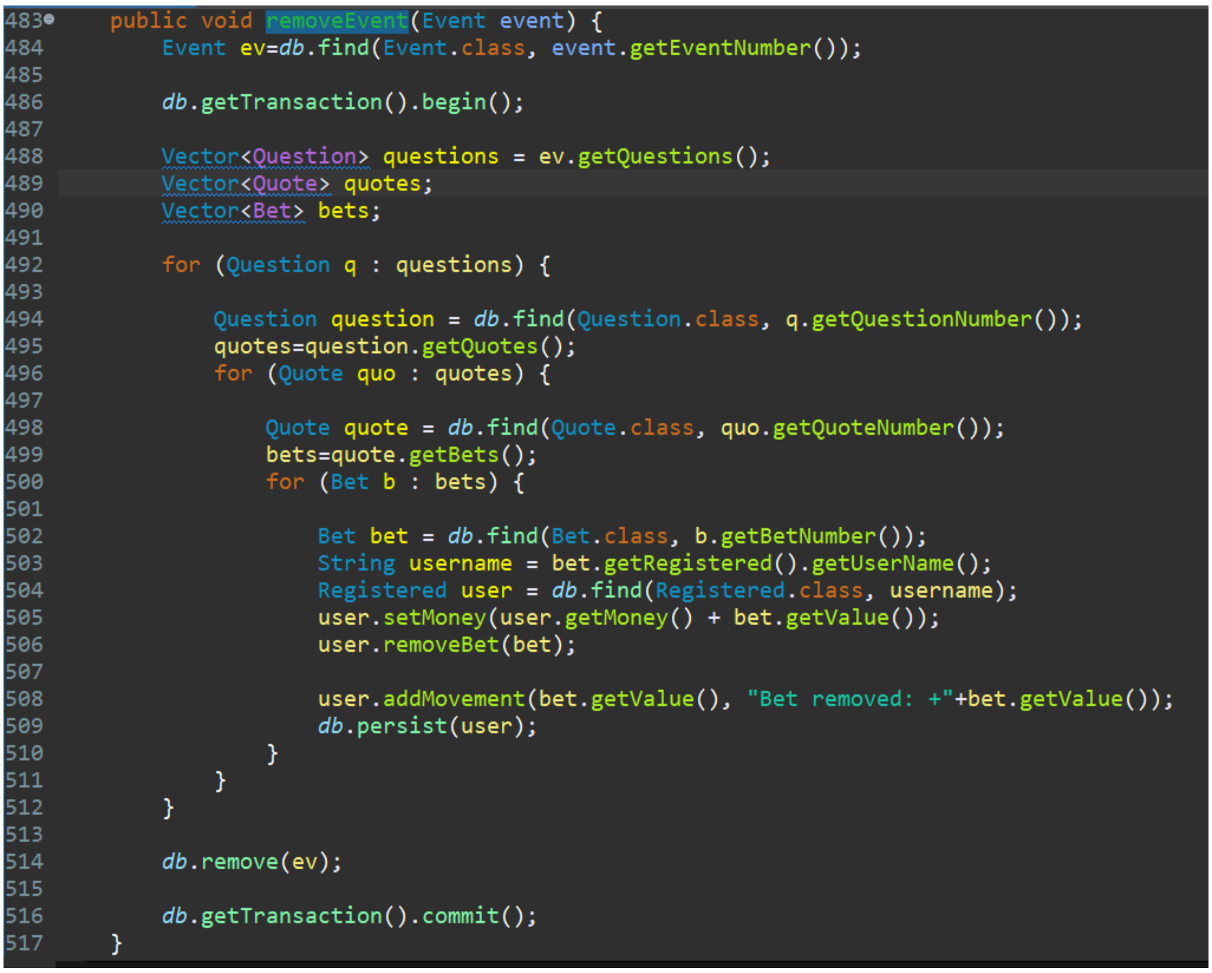
```java
    public Bet addBet(double value, Event event, Set<Quote> selectedQuotes, String username, Vector<Registered> alreadyBet) throws BetOnSameQuote, NotEnoughMoney
        String quoNums="";
        Event ev = db.find(Event.class, event.getEventNumber());

        Registered user = db.find(Registered.class, username);

        Bet bet=null;

        alreadyBet.add(user); //bisitatua bezala markatu

        Vector<Quote> quotes = new Vector<Quote>();
        quotes.addAll(selectedQuotes);

        if (user.betOnSameQuote(quotes)) throw new BetOnSameQuote();

        double newMoney = user.getMoney() - value;
        if (newMoney < 0 && alreadyBet.size()==1) {
            throw new NotEnoughMoney();
        } else if (newMoney>=0) {

            bet = extractedAddBet(value, quoNums, ev, user, quotes, new Vector<Quote>(), newMoney);

            followersTratatu(value, event, selectedQuotes, alreadyBet, user.getFollowers());
        }
        return bet;
    }
```

AddBet metodoa lerro gehiegi zituenez, 15 lerro izateko errefaktorizatu 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 dituena apustu berdina egiteko, addBet berriro deituz.

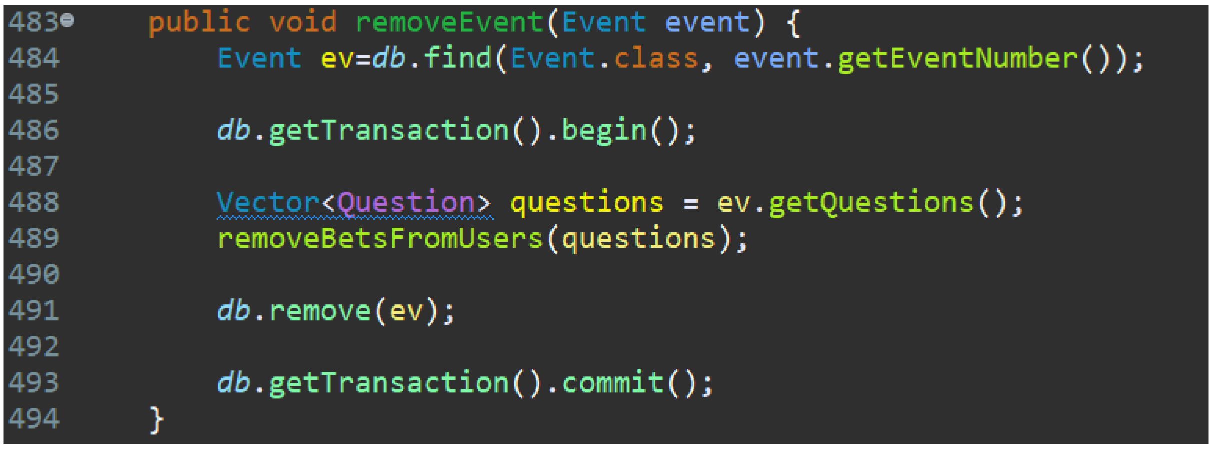
"Write simple units of code":

Hasierako kodea:

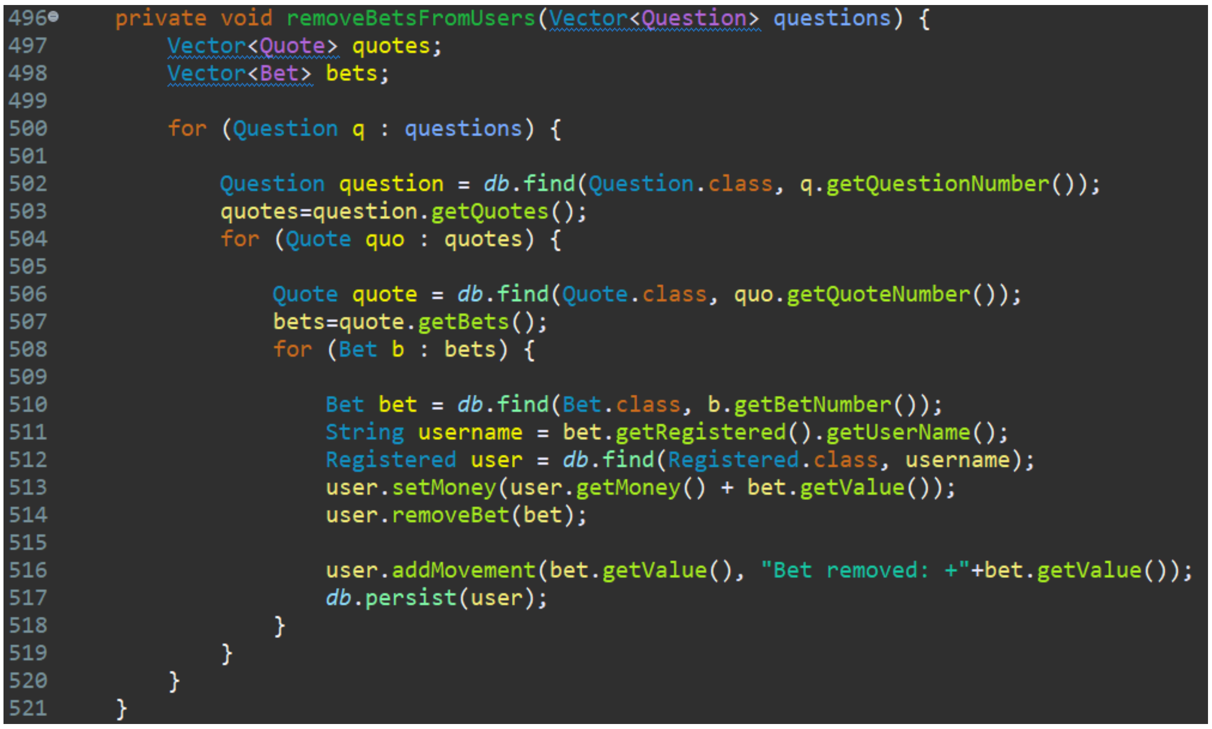


```
    public void removeEvent(Event event) {
        Event ev=db.find(Event.class, event.getEventNumber());

        db.getTransaction().begin();

        Vector<Question> questions = ev.getQuestions();
        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);
                }
            }
        }

        db.remove(ev);

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

Errefaktorizatutako kodea:



```
    public void removeEvent(Event event) {
        Event ev=db.find(Event.class, event.getEventNumber());

        db.getTransaction().begin();

        Vector<Question> questions = ev.getQuestions();
        removeBetsFromUsers(questions);

        db.remove(ev);

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

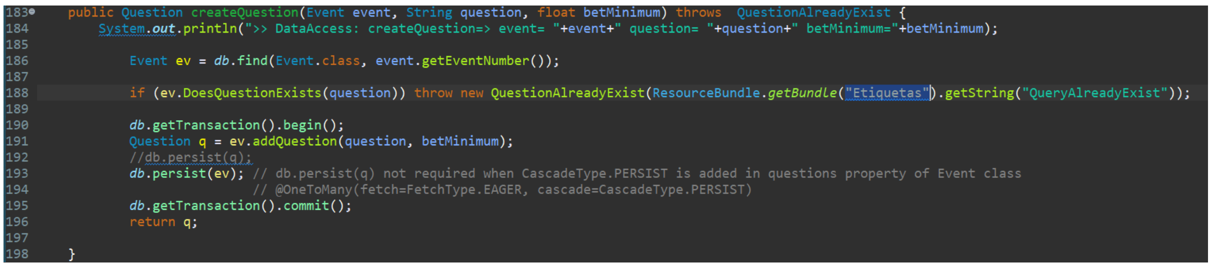


```java
    private void removeBetsFromUsers(Vector<Question> questions) {
        Vector<Quote> quotes;
        Vector<Bet> bets;

        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);
                }
            }
        }
    }
```

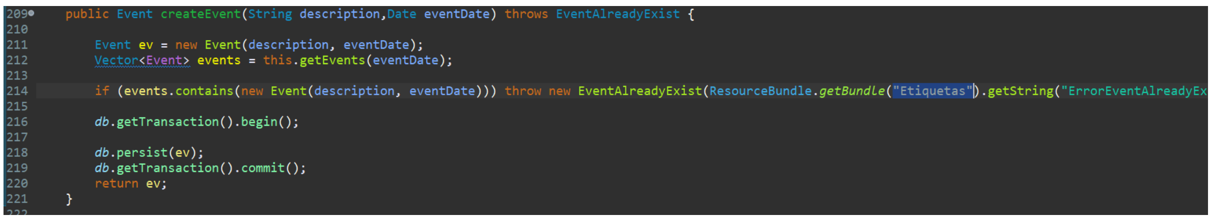
Metodo hau bi eginkizun zituenez, bi metodotan banatu da, removeBetsFromUsers sortuz. Metodo berrian ezabatuko den gertaeraren galdera guztien, kuota guztien apustu guztiak bere erabiltzailetatik ezabatzen dira. Metodo nagusian, metodo berria deitzen da eta ondoren gertaera datu-basetik ezabatzen da.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

<u>“Duplicate code”:</u>

Hasierako kodea:

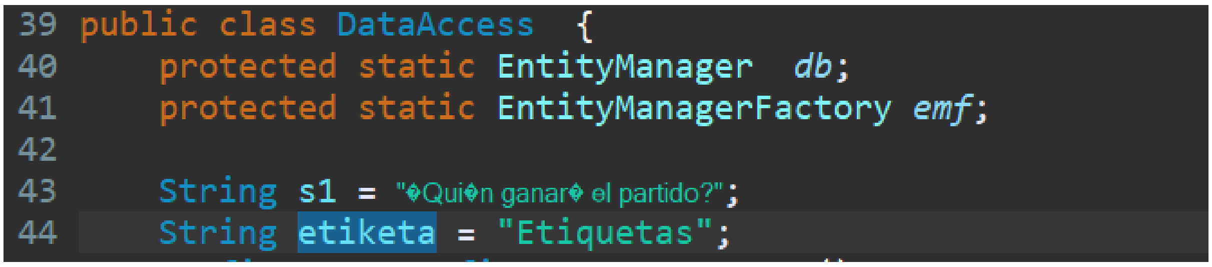


```java
    public Question createQuestion(Event event, String question, float betMinimum) throws  QuestionAlreadyExist {
        System.out.println(">> DataAccess: createQuestion=> event= "+event+" question= "+question+" betMinimum="+betMinimum);

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

            if (ev.DoesQuestionExists(question)) throw new QuestionAlreadyExist(ResourceBundle.getBundle("Etiquetas").getString("QueryAlreadyExist"));

            db.getTransaction().begin();
            Question q = ev.addQuestion(question, betMinimum);
            //db.persist(q);
            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.getTransaction().commit();
            return q;

    }
```

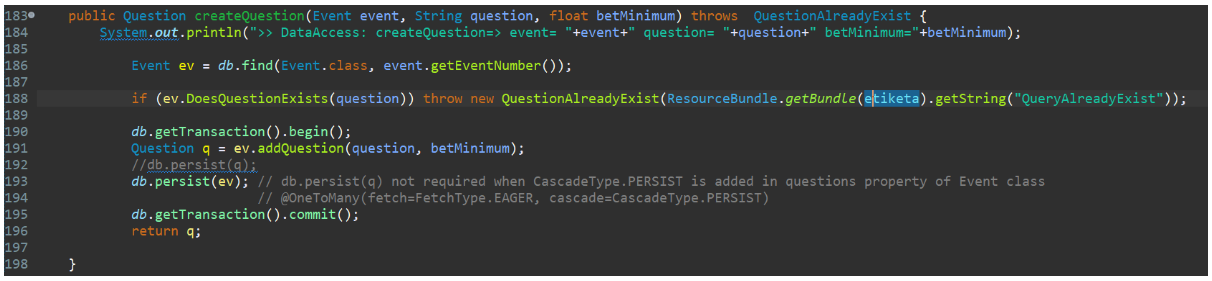


```java
    public Event createEvent(String description,Date eventDate) throws EventAlreadyExist {

        Event ev = new Event(description, eventDate);
        Vector<Event> events = this.getEvents(eventDate);

        if (events.contains(new Event(description, eventDate))) throw new EventAlreadyExist(ResourceBundle.getBundle("Etiquetas").getString("ErrorEventAlreadyEx

        db.getTransaction().begin();

        db.persist(ev);
        db.getTransaction().commit();
        return ev;
    }
```

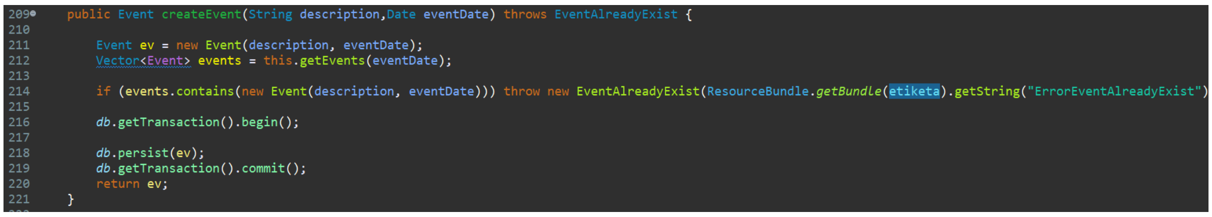
Errefaktorizatuko kodea:



```java
public class DataAccess  {
    protected static EntityManager  db;
    protected static EntityManagerFactory emf;

    String s1 = "�Qui�n ganar� el partido?";
    String etiketa = "Etiquetas";
```



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

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

            if (ev.DoesQuestionExists(question)) throw new QuestionAlreadyExist(ResourceBundle.getBundle(etiketa).getString("QueryAlreadyExist"));

            db.getTransaction().begin();
            Question q = ev.addQuestion(question, betMinimum);
            //db.persist(q);
            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.getTransaction().commit();
            return q;

    }
```

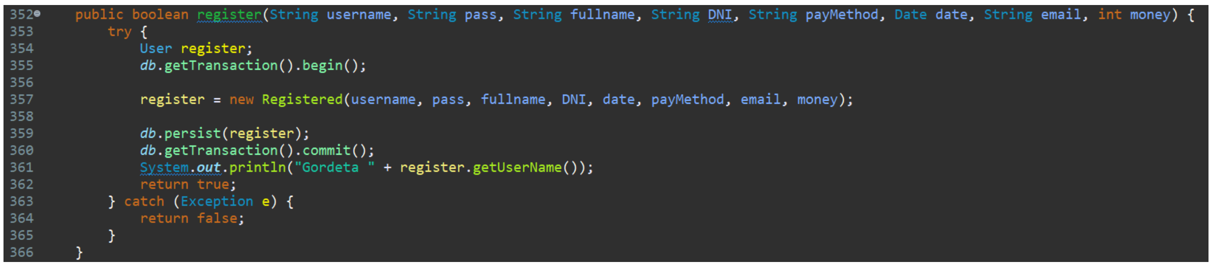


```java
    public Event createEvent(String description,Date eventDate) throws EventAlreadyExist {

        Event ev = new Event(description, eventDate);
        Vector<Event> events = this.getEvents(eventDate);

        if (events.contains(new Event(description, eventDate))) throw new EventAlreadyExist(ResourceBundle.getBundle(etiketa).getString("ErrorEventAlreadyExist")

        db.getTransaction().begin();

        db.persist(ev);
        db.getTransaction().commit();
        return ev;
    }
```

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

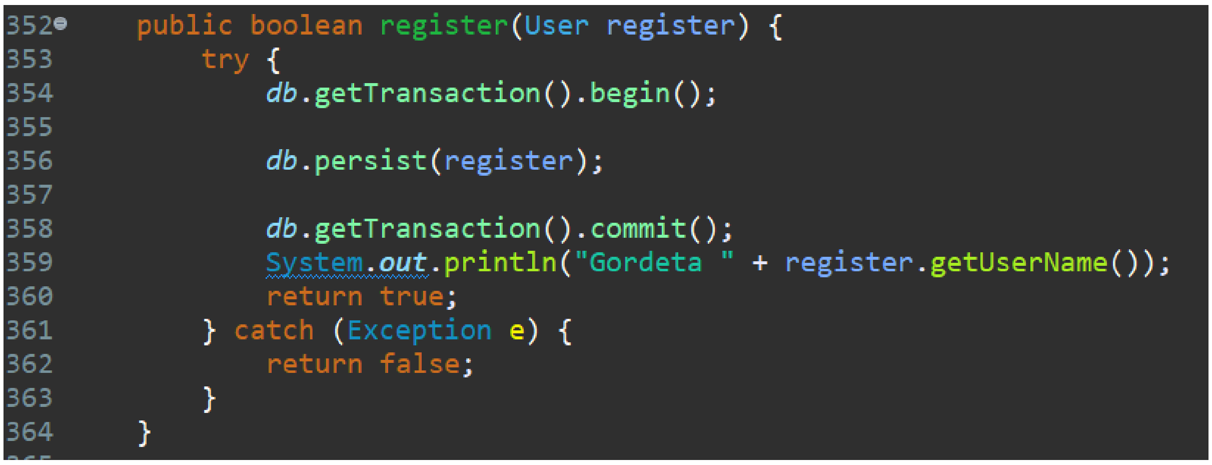
"Keep unit interfaces small":

Hasierako kodea:



```java
    public boolean register(String username, String pass, String fullname, String DNI, String payMethod, Date date, String email, int money) {
        try {
            User register;
            db.getTransaction().begin();

            register = new Registered(username, pass, fullname, DNI, date, payMethod, email, money);

            db.persist(register);
            db.getTransaction().commit();
            System.out.println("Gordeta " + register.getUserName());
            return true;
        } catch (Exception e) {
            return false;
        }
    }
```

Errefaktorizatuko kodea:



```
    public boolean register(User register) {
        try {
            db.getTransaction().begin();

            db.persist(register);

            db.getTransaction().commit();
            System.out.println("Gordeta " + register.getUserName());
            return true;
        } catch (Exception e) {
            return false;
        }
    }
```

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 objetua dataAccess-en barruan sortu beharrean, metodoa deitu aurretik sortzen da, lehen kodean zeuden parametroak erabiliz.