Refactor

Egilea: Ander Pollacino Merino

GitHub helbidea: https://github.com/anderpm/Bets

Sonarcloud helbidea: https://sonarcloud.io/dashboard?id=anderpm

- Write short units of code:

  Hasierako kodea:

```
539    public boolean setResult(Pronostic p) {
540        try {
541            db.getTransaction().begin();
542            Pronostic pronostic = db.find(Pronostic.class, p.getId());
543            pronostic.deleteNull();
544            System.out.println(pronostic.getDescription());
545            Vector<MultipleBets> bets = pronostic.getBets();
546
547            for(MultipleBets b:bets) {
548                System.out.println(b.getId());
549                if(b.getBetLength() == 1) {
550                    b.deleteBet(pronostic);
551                    Bezeroa bezero = db.find(Bezeroa.class, b.getUserName());
552                    double actual = bezero.getCash();
553                    bezero.setCash(actual + (b.getAmount()*b.getTotalPronostic()));
554                    System.out.println(bezero.getCash());
555                    bezero.addTransactionInOut(0, (b.getAmount()*b.getTotalPronostic()));
556                    db.persist(bezero);
557                } else {
558                    b.deleteBet(pronostic);
559                }
560                db.persist(b);
561                if(b.getBetLength()==0) {
562                    p.deleteMB(b);
563                    db.remove(b);
564                }
565            }
566            db.getTransaction().commit();
567            return true;
568        } catch(Exception e) {
569            System.out.println(e);
570            return false;
571        }
572    }
```

  Kodea mantenua hobetzeko gehienez 15 lerro izan behar ditu, eta *setResult* metodoak 27 ditu, beraz hori zuzenduko dugu:

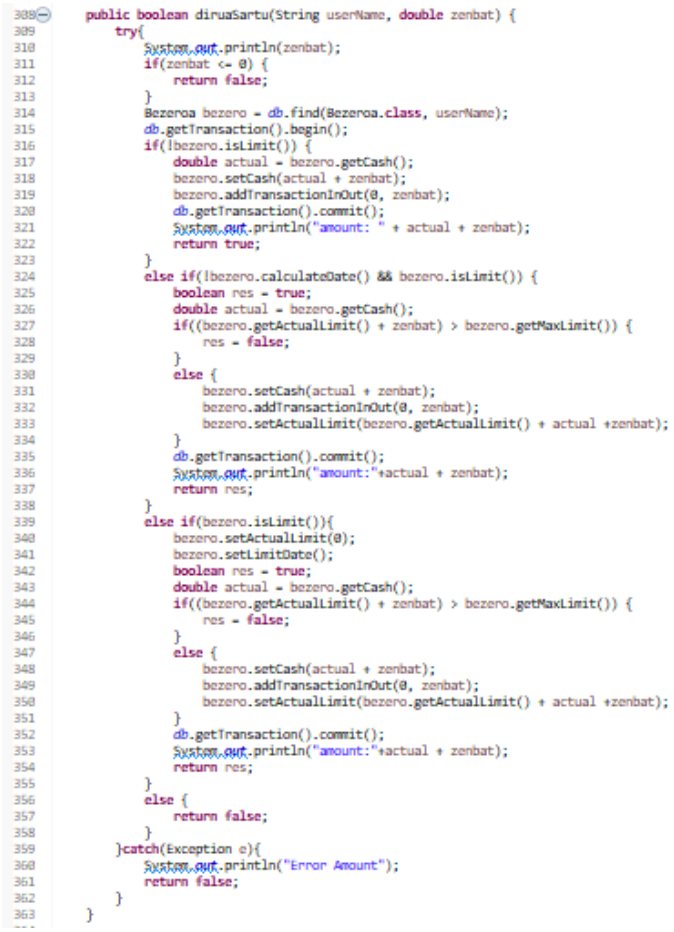
Errefaktorizatutako kodea:

```
539    public boolean setResult(Pronostic p) {
540        try {
541            db.getTransaction().begin();
542            Pronostic pronostic = db.find(Pronostic.class, p.getId());
543            pronostic.deleteNull();
544            System.out.println(pronostic.getDescription());
545            Vector<MultipleBets> bets = pronostic.getBets();
546
547            extracted1(p, pronostic, bets);
548
549            db.getTransaction().commit();
550            return true;
551        } catch(Exception e) {
552            System.out.println(e);
553            return false;
554        }
555    }
556
557    private void extracted1(Pronostic p, Pronostic pronostic, Vector<MultipleBets> bets) {
558        for(MultipleBets b:bets) {
559            System.out.println(b.getId());
560            if(b.getBetLength() == 1) {
561                extracted2(pronostic, b);
562            } else {
563                b.deleteBet(pronostic);
564            }
565            db.persist(b);
566            if(b.getBetLength()==0) {
567                p.deleteMB(b);
568                db.remove(b);
569            }
570        }
571    }
572
573    private void extracted2(Pronostic pronostic, MultipleBets b) {
574        b.deleteBet(pronostic);
575        Bezeroa bezero = db.find(Bezeroa.class, b.getUserName());
576        double actual = bezero.getCash();
577        bezero.setCash(actual + (b.getAmount()*b.getTotalPronostic()));
578        System.out.println(bezero.getCash());
579        bezero.addTransactionInOut(0, (b.getAmount()*b.getTotalPronostic()));
580        db.persist(bezero);
581    }
```

Kodearen matenua hobetzeko, ulergarria egiteko, test-ak hobeto egiteko… unitate txikietan banatu dugu, horretarako, *setResult*-eko for begizta aukeratu eta Refactor>>Extract Method egitean, metodo honetatik beste berri batera (*extracted1*) dei egingo du. Gero azken honetan if-aren barruan zegoena beste berri batera aldatu dugu (*extracted2*).

- Write simple units of code:

  Hasierako kodea:



```
public boolean diruaSartu(String userName, double zenbat) {
    try{
        System.out.println(zenbat);
        if(zenbat <= 0) {
            return false;
        }
        Bezeroa bezero = db.find(Bezeroa.class, userName);
        db.getTransaction().begin();
        if(!bezero.isLimit()) {
            double actual = bezero.getCash();
            bezero.setCash(actual + zenbat);
            bezero.addTransactionInOut(0, zenbat);
            db.getTransaction().commit();
            System.out.println("amount: " + actual + zenbat);
            return true;
        }
        else if(!bezero.calculateDate() && bezero.isLimit()) {
            boolean res = true;
            double actual = bezero.getCash();
            if((bezero.getActualLimit() + zenbat) > bezero.getMaxLimit()) {
                res = false;
            }
            else {
                bezero.setCash(actual + zenbat);
                bezero.addTransactionInOut(0, zenbat);
                bezero.setActualLimit(bezero.getActualLimit() + actual +zenbat);
            }
            db.getTransaction().commit();
            System.out.println("amount:"+actual + zenbat);
            return res;
        }
        else if(bezero.isLimit()){
            bezero.setActualLimit(0);
            bezero.setLimitDate();
            boolean res = true;
            double actual = bezero.getCash();
            if((bezero.getActualLimit() + zenbat) > bezero.getMaxLimit()) {
                res = false;
            }
            else {
                bezero.setCash(actual + zenbat);
                bezero.addTransactionInOut(0, zenbat);
                bezero.setActualLimit(bezero.getActualLimit() + actual +zenbat);
            }
            db.getTransaction().commit();
            System.out.println("amount:"+actual + zenbat);
            return res;
        }
        else {
            return false;
        }
    }catch(Exception e){
        System.out.println("Error Amount");
        return false;
    }
}
```

  Ikusi daikeenez *diruaSartu* metodoa luzea da eta bere konplexutasun ziklomatikoa 8 da. Beraz mantenua hobetzeko askoarekin, hurrengo errefactorizazioa egin dugu:

Errefaktorizatutako kodea:

```
308     public boolean diruaSartu(String userName, double zenbat) {
309         try{
310             System.out.println(zenbat);
311             if(zenbat <= 0) {
312                 return false;
313             } else {
314                 return extractedDiruaSartu1(userName, zenbat);
315             }
316         }catch(Exception e){
317             System.out.println("Error Amount");
318             return false;
319         }
320     }
321
322     private boolean extractedDiruaSartu1(String userName, double zenbat) {
323         Bezeroa bezero = db.find(Bezeroa.class, userName);
324         db.getTransaction().begin();
325         if(!bezero.isLimit()) {
326             double actual = bezero.getCash();
327             bezero.setCash(actual + zenbat);
328             bezero.addTransactionInOut(0, zenbat);
329             db.getTransaction().commit();
330             System.out.println("amount: " + actual + zenbat);
331             return true;
332         }
333         else if(!bezero.calculateDate() && bezero.isLimit()) {
334             return extractedDiruaSartu2(zenbat, bezero);
335         }
336         else if(bezero.isLimit()){
337             bezero.setActualLimit(0);
338             bezero.setLimitDate();
339             return extractedDiruaSartu2(zenbat, bezero);
340         }
341         else {
342             return false;
343         }
344     }
345
346     private boolean extractedDiruaSartu2(double zenbat, Bezeroa bezero) {
347         boolean res = true;
348         double actual = bezero.getCash();
349         if((bezero.getActualLimit() + zenbat) > bezero.getMaxLimit()) {
350             res = false;
351         }
352         else {
353             bezero.setCash(actual + zenbat);
354             bezero.addTransactionInOut(0, zenbat);
355             bezero.setActualLimit(bezero.getActualLimit() + actual +zenbat);
356         }
357         db.getTransaction().commit();
358         System.out.println("amount:"+actual + zenbat);
359         return res;
360     }
```

diruaSartu metodoaren konplexutasun ziklomatikoa gutxitzeko, lehenengo if-aren ondorengo guztia Refactor>>Extract Method egin dugu eta *extractedDiruaSartu1* metodoa sortudugu, ondoren metodo honetan berdina egin dugu *extractedDiruaSartu2* sortuz.
DiruaSartu metodoaren testak eginda zeudenez, errefaktorizazioa egin ondoren, testak ondo exekutatzen direla egiaztatu behar izan dugu.

- Duplicate code:

  Define a constant instead of duplicating this literal "¿Quién ganará el partido?" 3 times.
  Define a constant instead of duplicating this literal "Who will win the match?" 3 times.
  Define a constant instead of duplicating this literal "Zeinek irabaziko du partidua?" 3 times.

  Hasierako kodea:

```
105             if (Locale.getDefault().equals(new Locale("es"))) {
106                 q1=ev1.addQuestion("Â¿QuiÃ©n ganarÃ¡ el partido?",1);
107                 q2=ev1.addQuestion("Â¿QuiÃ©n meterÃ¡ el primer gol?",2);
108                 q3=ev11.addQuestion("Â¿QuiÃ©n ganarÃ¡ el partido?",1);
109                 q4=ev11.addQuestion("Â¿CuÃ¡ntos goles se marcarÃ¡n?",2);
110                 q5=ev17.addQuestion("Â¿QuiÃ©n ganarÃ¡ el partido?",1);
111                 q6=ev17.addQuestion("Â¿HabrÃ¡ goles en la primera parte?",2);
112             }
113             else if (Locale.getDefault().equals(new Locale("en"))) {
114                 q1=ev1.addQuestion("Who will win the match?",1);
115                 q2=ev1.addQuestion("Who will score first?",2);
116                 q3=ev11.addQuestion("Who will win the match?",1);
117                 q4=ev11.addQuestion("How many goals will be scored in the match?",2);
118                 q5=ev17.addQuestion("Who will win the match?",1);
119                 q6=ev17.addQuestion("Will there be goals in the first half?",2);
120             }
121             else {
122                 q1=ev1.addQuestion("Zeinek irabaziko du partidua?",1);
123                 q2=ev1.addQuestion("Zeinek sartuko du lehenengo gola?",2);
124                 q3=ev11.addQuestion("Zeinek irabaziko du partidua?",1);
125                 q4=ev11.addQuestion("Zenbat gol sartuko dira?",2);
126                 q5=ev17.addQuestion("Zeinek irabaziko du partidua?",1);
127                 q6=ev17.addQuestion("Golak sartuko dira lehenengo zatian?",2);
128             }
```

  Kode hau errefaktorizatzeko, nahi dugu kodea hautatu dugu, "Â¿QuiÃ©n ganarÃ¡ el partido?" adibidez, Refactor>>Extract Local Variable aukeratu eta honek hurrengo kodea sortuko du:

  Errefaktorizatutako kodea:

```
105             if (Locale.getDefault().equals(new Locale("es"))) {
106                 final String question1 = "Â¿QuiÃ©n ganarÃ¡ el partido?";
107                 q1=ev1.addQuestion(question1,1);
108                 q2=ev1.addQuestion("Â¿QuiÃ©n meterÃ¡ el primer gol?",2);
109                 q3=ev11.addQuestion(question1,1);
110                 q4=ev11.addQuestion("Â¿CuÃ¡ntos goles se marcarÃ¡n?",2);
111                 q5=ev17.addQuestion(question1,1);
112                 q6=ev17.addQuestion("Â¿HabrÃ¡ goles en la primera parte?",2);
113             }
114             else if (Locale.getDefault().equals(new Locale("en"))) {
115                 final String question2 = "Who will win the match?";
116                 q1=ev1.addQuestion(question2,1);
117                 q2=ev1.addQuestion("Who will score first?",2);
118                 q3=ev11.addQuestion(question2,1);
119                 q4=ev11.addQuestion("How many goals will be scored in the match?",2);
120                 q5=ev17.addQuestion(question2,1);
121                 q6=ev17.addQuestion("Will there be goals in the first half?",2);
122             }
123             else {
124                 final String question3 = "Zeinek irabaziko du partidua?";
125                 q1=ev1.addQuestion(question3,1);
126                 q2=ev1.addQuestion("Zeinek sartuko du lehenengo gola?",2);
127                 q3=ev11.addQuestion(question3,1);
128                 q4=ev11.addQuestion("Zenbat gol sartuko dira?",2);
129                 q5=ev17.addQuestion(question3,1);
130                 q6=ev17.addQuestion("Golak sartuko dira lehenengo zatian?",2);
131             }
```

This branch's code block is the same as the block for the branch on line 631.

Hasierako kodea:

```
631         if(b.getLimitDate() == null) {
632             b.setLimitDate();
633             b.setLimit(true);
634             b.setMaxLimit(limit);
635             res = true;
636         }
637         else if(b.calculateDate()) {
638             b.setLimitDate();
639             b.setLimit(true);
640             b.setMaxLimit(limit);
641             res = true;
642         }
```

Errefaktorizatutako kodea:

```
631         if(b.getLimitDate() == null || b.calculateDate()) {
632             b.setLimitDate();
633             b.setLimit(true);
634             b.setMaxLimit(limit);
635             res = true;
636         }
```

Errefaktorizazio honekin, kode errepikapena kentzeaz gain, kodearen luzera txikitzen dugu.

- Keep unit interfaces small:

Mantenua hobetzeko, egokia da metodoen parametro kopurua gehienez 4 izatea. DataAccess klasean metodo bakarra dago, *addTBet,* 6 parametrokin.

Hasierako kodea:

```
580    public void addTBet(Bezeroa bezero, Event event, Question question, Pronostic pronostic, double betAmount, Vector<Bet> b) {
581        TMBet tb = new TMBet(question, pronostic, betAmount, b);
582        Bezeroa b2 = db.find(Bezeroa.class, bezero.getUserName());
583        db.getTransaction().begin();
584        b2.addTransaction(tb);
585        db.getTransaction().commit();
586    }
```

Errefaktorizatutako kodea:

```
580     public void addTBet(Bezeroa bezero, double betAmount, Vector<Bet> b, Vector<Object> refactor) {
581         Question q = (Question)refactor.get(0);
582         Pronostic p = (Pronostic)refactor.get(1);
583         TMBet tb = new TMBet(q, p, betAmount, b);
584         Bezeroa b2 = db.find(Bezeroa.class, bezero.getUserName());
585         db.getTransaction().begin();
586         b2.addTransaction(tb);
587         db.getTransaction().commit();
588     }
```

Event parametroa ez da erabiltzen, beraz hau ezabatuko dugu.
Vector berria sortu dugu, klasea hautatuz eta Refactor > Change Method Signature eginez eta bertan question eta pronostic objektuak sartu ditugu.