# ERREFAKTORIZIOA

- Danel Osa:
  - Write short units of code and write simple units of code :
    - Kode originala:

```
public boolean ApustuaEgin(Registered u, Vector<Quote> quote, Double balioa, Integer
apustuBikoitzaGalarazi) {
		Registered user = (Registered) db.find(Registered.class, u.getUsername());
		Boolean b;
		if(user.getDirukop()>=balioa) {
			db.getTransaction().begin();
			ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
			db.persist(apustuAnitza);
			for(Quote quo: quote) {
				Quote kuote = db.find(Quote.class, quo.getQuoteNumber());
				Apustua ap = new Apustua(apustuAnitza, kuote);
				db.persist(ap);
				apustuAnitza.addApustua(ap);
				kuote.addApustua(ap);
			}
			db.getTransaction().commit();
			db.getTransaction().begin();
			if(apustuBikoitzaGalarazi==-1) {
				apustuBikoitzaGalarazi=apustuAnitza.getApustuAnitzaNumber();
			}
			apustuAnitza.setApustuKopia(apustuBikoitzaGalarazi);
			user.updateDiruKontua(-balioa);
			Transaction t = new Transaction(user, balioa, new Date(), "ApustuaEgin");
			user.addApustuAnitza(apustuAnitza);
			for(Apustua a: apustuAnitza.getApustuak()) {
				Apustua apu = db.find(Apustua.class, a.getApostuaNumber());
				Quote q = db.find(Quote.class, apu.getKuota().getQuoteNumber());
				Sport spo =q.getQuestion().getEvent().getSport();
				spo.setApustuKantitatea(spo.getApustuKantitatea()+1);

			}
			user.addTransaction(t);
			db.persist(t);
			db.getTransaction().commit();
			for(Jarraitzailea reg:user.getJarraitzaileLista()) {
				Jarraitzailea erab=db.find(Jarraitzailea.class,
reg.getJarraitzaileaNumber());
				b=true;
				for(ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {
					if(apu.getApustuKopia()==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);
                                    }
                                }
                        }
                        return true;
                }else{
                        return false;
                }

        }
```

- Errefaktorizatutako kodea:

```
public boolean ApustuaEgin(Registered u, Vector<Quote> quote, Double balioa,
Integer apustuBikoitzaGalarazi) {
                Registered user = (Registered) db.find(Registered.class,
u.getUsername());
                Boolean b;
                if(user.getDirukop()>=balioa) {
                        db.getTransaction().begin();
                        ApustuAnitza apustuAnitza = ApustuaGehitu(quote, balioa,
user);
                        db.getTransaction().commit();
                        db.getTransaction().begin();
                        if(apustuBikoitzaGalarazi==-1) {

apustuBikoitzaGalarazi=apustuAnitza.getApustuAnitzaNumber();
                        }
                        apustuAnitza.setApustuKopia(apustuBikoitzaGalarazi);
                        UpdateApustuAnitza(balioa, user, apustuAnitza);
                        db.getTransaction().commit();
                        JarraitzaileakApustuEgin(quote, new
JarraitzaileakApustuEginParameter(balioa, apustuBikoitzaGalarazi, apustuAnitza),
user);
                        return true;
                }else{
                        return false;
                }
        }
        private void UpdateApustuAnitza(Double balioa, Registered user,
ApustuAnitza apustuAnitza) {
                user.updateDiruKontua(-balioa);
                Transaction t = new Transaction(user, balioa, new Date(),
"ApustuaEgin");
                user.addApustuAnitza(apustuAnitza);
                UpdateSport(apustuAnitza);
                user.addTransaction(t);
```

```
			db.persist(t);
		}
		private void JarraitzaileakApustuEgin(Vector<Quote> quote,
JarraitzaileakApustuEginParameter ApustuInfo, Registered user) {
			Boolean b;
			for(Jarraitzailea reg:user.getJarraitzaileLista()) {
				Jarraitzailea erab=db.find(Jarraitzailea.class,
reg.getJarraitzaileaNumber());
				b=true;
				for(ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {

if(apu.getApustuKopia()==ApustuInfo.apustuAnitza.getApustuKopia()) {
					b=false;
				}
				}
				if(b) {
					if(erab.getNork().getDiruLimitea()<ApustuInfo.balioa) {
					this.ApustuaEgin(erab.getNork(), quote,
erab.getNork().getDiruLimitea(), ApustuInfo.apustuBikoitzaGalarazi);
					}else{
					this.ApustuaEgin(erab.getNork(), quote,
ApustuInfo.balioa, ApustuInfo.apustuBikoitzaGalarazi);
					}
				}
			}
		}
		private void UpdateSport(ApustuAnitza apustuAnitza) {
			for(Apustua a: apustuAnitza.getApustuak()) {
				Apustua apu = db.find(Apustua.class, a.getApostuaNumber());
				Quote q = db.find(Quote.class,
apu.getKuota().getQuoteNumber());
				Sport spo =q.getQuestion().getEvent().getSport();
				spo.setApustuKantitatea(spo.getApustuKantitatea()+1);

			}
		}
		private ApustuAnitza ApustuaGehitu(Vector<Quote> quote, Double balioa,
Registered user) {
			ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
			db.persist(apustuAnitza);
			for(Quote quo: quote) {
				Quote kuote = db.find(Quote.class, quo.getQuoteNumber());
				Apustua ap = new Apustua(apustuAnitza, kuote);
				db.persist(ap);
				apustuAnitza.addApustua(ap);
				kuote.addApustua(ap);
			}
			return apustuAnitza;
		}
```

Apustuak egin funtzioa simplifikatzeko funtzioa 5 zatitan banatu dut: Apustuak egin funtzioan funtzioarentzat espezifikoak diren atalak utzi ditut, gainontzekoak apustuak eginen garatutako atal espezifikoetako informazioa prozesatzen duten atalaetan banandu ditut commit tartean egiten diren atalak eta balio garrantzitsuak itzultzen dituztela kontuan hartuz.
Aldaketa honen ondoren kodea sinplifikatzea eta motzago izatea lortu dut.

- Keep unit interfaces small:
- kode originala:

```
private void JarraitzaileakApustuEgin(Vector<Quote> quote, Double balioa, Integer
apustuBikoitzaGalarazi,
                    Registered user, ApustuAnitza apustuAnitza) {
             Boolean b;
             for(Jarraitzailea reg:user.getJarraitzaileLista()) {
                    Jarraitzailea erab=db.find(Jarraitzailea.class,
reg.getJarraitzaileaNumber());
                    b=true;
                    for(ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {
                           if(apu.getApustuKopia()==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);
                           }
                    }
             }
      }
```

- Errefaktorizatutako kodea:

```
      private void JarraitzaileakApustuEgin(Vector<Quote> quote,
JarraitzaileakApustuEginParameter ApustuInfo, Registered user) {
             Boolean b;
             for(Jarraitzailea reg:user.getJarraitzaileLista()) {
                    Jarraitzailea erab=db.find(Jarraitzailea.class,
reg.getJarraitzaileaNumber());
                    b=true;
                    for(ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {

if(apu.getApustuKopia()==ApustuInfo.apustuAnitza.getApustuKopia()) {
                                  b=false;
                           }
```

```
                    }
                    if(b) {
                          if(erab.getNork().getDiruLimitea()<ApustuInfo.balioa) {
                                this.ApustuaEgin(erab.getNork(), quote,
erab.getNork().getDiruLimitea(), ApustuInfo.apustuBikoitzaGalarazi);
                          }else{
                                this.ApustuaEgin(erab.getNork(), quote,
ApustuInfo.balioa, ApustuInfo.apustuBikoitzaGalarazi);
                          }
                    }
              }
        }
```

Hasieran ApustuakEgin funtziotik errefaktorizaziotik sortutako funtzio batean parametro kopurua 5 ekoa den funtzio bat lortu dut, parametro kopurua jaisteko errefaktorazio bidez  balioa, apustuBikoitzaGalarazi eta user parametroak, apustuari erlazionatutako paramtroak, ApustuInfo parametroan gorde ditut.

- Duplicate code:
- Kode originala(kodea oso luzea denez aldaketak jasotako atala bakarrik erakutsiko dut):

```
      public void initializeDB(){
...

                    this.DiruaSartu(new DiruaSartuParameter(reg1, new Date(),
"DiruaSartu"), 50.0);

                    this.DiruaSartu(new DiruaSartuParameter(reg2, new Date(),
"DiruaSartu"), 50.0);

                    this.DiruaSartu(new DiruaSartuParameter(reg3, new Date(),
"DiruaSartu"), 50.0);
                    this.DiruaSartu(new DiruaSartuParameter(reg4, new Date(),
"DiruaSartu"), 50.0);


...
```

- Errefaktorizatutako kodea:

```
      public void initializeDB(){
...

                    String Dirua = "DiruaSartu";
```

```
                this.DiruaSartu(new DiruaSartuParameter(reg1, new Date(),
Dirua), 50.0);

                this.DiruaSartu(new DiruaSartuParameter(reg2, new Date(),
Dirua), 50.0);

                this.DiruaSartu(new DiruaSartuParameter(reg3, new Date(),
Dirua), 50.0);
                this.DiruaSartu(new DiruaSartuParameter(reg4, new Date(),
Dirua), 50.0);
```

…

Hasierako kodean ikus daiteke transakzio bat sortu behar den bakoitzean string berdina erabiltzen dela, hori dela eta, kodea txukuntzeko stringa Dirua bariablean sartu dut transakzioak hau erabili dezan.

- Ekaitz Murillo
    - Write Short units of code
        - Hasierako Kodea:

```
public boolean gertaerakSortu(String description,Date eventDate, String sport)
{
        boolean b = true;
        db.getTransaction().begin();
        Sport spo =db.find(Sport.class, sport);
        if(spo!=null) {
            TypedQuery<Event> Equery = db.createQuery("SELECT e
FROM Event e WHERE e.getEventDate() =?1 ",Event.class);
            Equery.setParameter(1, eventDate);
            for(Event ev: Equery.getResultList()) {
                if(ev.getDescription().equals(description)) {
                    b = false;
                }
            }
            if(b) {
                String[] taldeak = description.split("-");
                Team lokala = new Team(taldeak[0]);
                Team kanpokoa = new Team(taldeak[1]);
```

```
                        Event e = new Event(description, eventDate, lokala,
kanpokoa);
                        e.setSport(spo);
                        spo.addEvent(e);
                        db.persist(e);
                    }
                }else {
                    return false;
                }
                db.getTransaction().commit();
                return b;
        }
```

- Errefaktoritatutako kodea:

```
public boolean gertaerakSortu(String description,Date eventDate, String sport)
{
                boolean b = true;
                db.getTransaction().begin();
                Sport spo =db.find(Sport.class, sport);
                if(spo!=null) {
                    TypedQuery<Event> Equery = db.createQuery("SELECT e
FROM Event e WHERE e.getEventDate() =?1 ",Event.class);
                    Equery.setParameter(1, eventDate);
                    for(Event ev: Equery.getResultList()) {
                        if(ev.getDescription().equals(description)) {
                            b = false;
                        }
                    }
                    if(b) {
                        gertaerakGehitu(description, eventDate, spo);
                    }
                }else {
                    return false;
                }
                db.getTransaction().commit();
                return b;
        }

        public void gertaerakGehitu(String description, Date eventDate, Sport
spo) {
```

```
            String[] taldeak = description.split("-");
            Team lokala = new Team(taldeak[0]);
            Team kanpokoa = new Team(taldeak[1]);
            Event e = new Event(description, eventDate, lokala, kanpokoa);
            e.setSport(spo);
            spo.addEvent(e);
            db.persist(e);
        }
```

Metodo honek hainbeste lerro ez izateko metodo berri bat sortu dut (gertaerakGehitu), horrela bi metodoek ez dute 15 lerro baino gehiago izango

- Write simple units of code
  - Hasierako kodea

```
public boolean gertaeraEzabatu(Event ev) {
            Event event  = db.find(Event.class, ev);
            boolean resultB = true;
            List<Question> listQ = event.getQuestions();

            for(Question q : listQ) {
                if(q.getResult() == null) {
                    resultB = false;
                }
            }
            if(!resultB) {
                return false;
            }else if(new Date().compareTo(event.getEventDate())<0) {
                TypedQuery<Quote> Qquery = db.createQuery("SELECT q
FROM Quote q WHERE q.getQuestion().getEvent().getEventNumber() =?1",
Quote.class);
                Qquery.setParameter(1, event.getEventNumber());
                List<Quote> listQUO = Qquery.getResultList();
                for(int j=0; j<listQUO.size(); j++) {
                    Quote quo = db.find(Quote.class, listQUO.get(j));
                    for(int i=0; i<quo.getApustuak().size(); i++) {
                        ApustuAnitza apustuAnitza =
quo.getApustuak().get(i).getApustuAnitza();
```

```
                                    ApustuAnitza ap1 =
db.find(ApustuAnitza.class, apustuAnitza.getApustuAnitzaNumber());
                                    db.getTransaction().begin();
                                    ap1.removeApustua(quo.getApustuak().get(i));
                                    db.getTransaction().commit();
                                    if(ap1.getApustuak().isEmpty() &&
!ap1.getEgoera().equals("galduta")) {
                                            this.apustuaEzabatu(ap1.getUser(),
ap1);
                                    }else if(!ap1.getApustuak().isEmpty() &&
ap1.irabazitaMarkatu()){
                                            this.ApustuaIrabazi(ap1);
                                    }
                                    db.getTransaction().begin();
                                    Sport spo
=quo.getQuestion().getEvent().getSport();

spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
                                    db.getTransaction().commit();
                            }
                    }

            }
            db.getTransaction().begin();
            db.remove(event);
            db.getTransaction().commit();
            return true;
    }
```

- Amaierako kodea

```
public boolean gertaeraEzabatu(Event ev) {
            Event event  = db.find(Event.class, ev);
            boolean resultB = true;
            List<Question> listQ = event.getQuestions();

            resultB = galderaNull(listQ);
            if(!resultB) {
                    return false;
            }else if(new Date().compareTo(event.getEventDate())<0) {
```

```
                    TypedQuery<Quote> Qquery = db.createQuery("SELECT q
FROM Quote q WHERE q.getQuestion().getEvent().getEventNumber() =?1",
Quote.class);
                    Qquery.setParameter(1, event.getEventNumber());
                    List<Quote> listQUO = Qquery.getResultList();
                    for(int j=0; j<listQUO.size(); j++) {
                        Quote quo = db.find(Quote.class, listQUO.get(j));
                        for(int i=0; i<quo.getApustuak().size(); i++) {
                            ApustuAnitza apustuAnitza =
quo.getApustuak().get(i).getApustuAnitza();
                            ApustuAnitza ap1 =
db.find(ApustuAnitza.class, apustuAnitza.getApustuAnitzaNumber());
                            db.getTransaction().begin();
                            ap1.removeApustua(quo.getApustuak().get(i));
                            db.getTransaction().commit();
                            if(ap1.getApustuak().isEmpty() &&
!ap1.getEgoera().equals("galduta")) {
                                this.apustuaEzabatu(ap1.getUser(),
ap1);
                            }else if(!ap1.getApustuak().isEmpty() &&
ap1.irabazitaMarkatu()){
                                this.ApustuaIrabazi(ap1);
                            }
                            db.getTransaction().begin();
                            Sport spo
=quo.getQuestion().getEvent().getSport();

spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
                            db.getTransaction().commit();
                        }
                    }

                }
                db.getTransaction().begin();
                db.remove(event);
                db.getTransaction().commit();
                return true;
        }

        public boolean galderaNull(List<Question> listQ) {
```

```
            for(Question q : listQ) {
                if(q.getResult() == null) {
                    return  false;
                }

            }
            return true;
        }
```

Komplexutasun ziklomatikoa jaizteko hainbat bifurkazio metodotik atera ditut

- Duplicate code
  - Hasierako Kodea

```
Transaction t1 = new Transaction(reg1, apA1.getBalioa(), new Date(),
"ApustuaEgin");
                    Transaction t3 = new Transaction(reg2, apA4.getBalioa(),
new Date(), "ApustuaEgin");
                    Transaction t4 = new Transaction(reg3, apA5.getBalioa(),
new Date(), "ApustuaEgin");
                    Transaction t5 = new Transaction(reg4, apA3.getBalioa(),
new Date(), "ApustuaEgin");
                    Transaction t6 = new Transaction(reg4, apA6.getBalioa(),
new Date(), "ApustuaEgin");
                    Transaction t7 = new Transaction(reg1, apA7.getBalioa(),
new Date(), "ApustuaEgin");
                    Transaction t8 = new Transaction(reg1, apA8.getBalioa(),
new Date(), "ApustuaEgin");
                    Transaction t9 = new Transaction(reg2, apA9.getBalioa(),
new Date(), "ApustuaEgin");
                    Transaction t10 = new Transaction(reg2,
apA10.getBalioa(), new Date(), "ApustuaEgin");
                    Transaction t11 = new Transaction(reg3, apA11.getBalioa(),
new Date(), "ApustuaEgin");
                    Transaction t12 = new Transaction(reg3,
apA12.getBalioa(), new Date(), "ApustuaEgin");
```

  - Errefaktorizatutako kodea

```
String apeg= “ApustuaEgin”
Transaction t1 = new Transaction(reg1, apA1.getBalioa(), new Date(),
"ApustuaEgin");
                    Transaction t3 = new Transaction(reg2, apA4.getBalioa(),
new Date(), apeg");
                    Transaction t4 = new Transaction(reg3, apA5.getBalioa(),
new Date(), apeg);
                    Transaction t5 = new Transaction(reg4, apA3.getBalioa(),
new Date(), apeg);
                    Transaction t6 = new Transaction(reg4, apA6.getBalioa(),
new Date(), apeg);
                    Transaction t7 = new Transaction(reg1, apA7.getBalioa(),
new Date(), apeg);
                    Transaction t8 = new Transaction(reg1, apA8.getBalioa(),
new Date(), apeg);
                    Transaction t9 = new Transaction(reg2, apA9.getBalioa(),
new Date(), apeg);
                    Transaction t10 = new Transaction(reg2,
apA10.getBalioa(), new Date(), apeg);
                    Transaction t11 = new Transaction(reg3, apA11.getBalioa(),
new Date(), apeg);
                    Transaction t12 = new Transaction(reg3,
apA12.getBalioa(), new Date(), apeg);
```

12 aldiz ez errepikatzeko “ApustuaEgin” aldagai bat definitzen dugu, bertan testu hori jartzen duena

- Keep unit interface small
  - Hasierako kodea:

```
public void DiruaSartu(Registered u, Double dirua, Date data, String mota) {
        Registered user = (Registered) db.find(Registered.class,
u.getUsername());
        db.getTransaction().begin();
        Transaction t = new Transaction(user, dirua, data, mota);
        System.out.println(t.getMota());
        user.addTransaction(t);
        user.updateDiruKontua(dirua);
        db.persist(t);
        db.getTransaction().commit();
    }
```

        - Amaierako kodea:

```
public void DiruaSartu(DiruaSartuParameter parameterObject, Double dirua) {
        Registered user = (Registered) db.find(Registered.class,
parameterObject.u.getUsername());
        db.getTransaction().begin();
        Transaction t = new Transaction(user, dirua,
parameterObject.data, parameterObject.mota);
        System.out.println(t.getMota());
        user.addTransaction(t);
        user.updateDiruKontua(dirua);
        db.persist(t);
        db.getTransaction().commit();
    }
```

Hainbeste parametro ez izateko 3 parametro parametro bakar batean fusionatu ditut

- Iñigo Gil:
    - Hasierako kodea.

```
if (Locale.getDefault().equals(new Locale("es"))) {
   q1 = ev1.addQuestion("¿Quién ganará el partido?", 1);
   q2 = ev1.addQuestion("¿Quién meterá el primer gol?", 2);
   q3 = ev11.addQuestion("¿Quién ganará el partido?", 1);
   q4 = ev11.addQuestion("¿Cuántos goles se marcarán?", 2);
   q5 = ev17.addQuestion("¿Quién ganará el partido?", 1);
   q6 = ev17.addQuestion("¿Habrá goles en la primera parte?", 2);
   }
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);
   } 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);

    }
```

- Errefaktorizatuko kodea.

```
final String ganarPartidoString = "¿Quién ganará el partido?";
final String winMatchString = "Who will win the match?";
final String partiduaIrabaziString = "Zeinek irabaziko du partidua?";
if (Locale.getDefault().equals(new Locale("es"))) {
  q1 = ev1.addQuestion(ganarPartidoString, 1);
  q2 = ev1.addQuestion("¿Quién meterá el primer gol?", 2);
  q3 = ev11.addQuestion(ganarPartidoString, 1);
  q4 = ev11.addQuestion("¿Cuántos goles se marcarán?", 2);
  q5 = ev17.addQuestion(ganarPartidoString, 1);
  q6 = ev17.addQuestion("¿Habrá goles en la primera parte?", 2);
} else if (Locale.getDefault().equals(new Locale("en"))) {
  q1 = ev1.addQuestion(winMatchString, 1);
  q2 = ev1.addQuestion("Who will score first?", 2);
  q3 = ev11.addQuestion(winMatchString, 1);
  q4 = ev11.addQuestion("How many goals will be scored in the match?",
2);
  q5 = ev17.addQuestion(winMatchString, 1);
  q6 = ev17.addQuestion("Will there be goals in the first half?", 2);
  } else {
      q1 = ev1.addQuestion(partiduaIrabaziString, 1);
      q2 = ev1.addQuestion("Zeinek sartuko du lehenengo gola?", 2);
      q3 = ev11.addQuestion(partiduaIrabaziString, 1);
      q4 = ev11.addQuestion("Zenbat gol sartuko dira?", 2);
      q5 = ev17.addQuestion(partiduaIrabaziString, 1);
      q6 = ev17.addQuestion("Golak sartuko dira lehenengo zatian?", 2);
}
```

- Egindako errefaktorizazioren deskribapena.
  “Duplicate code”
  Lerro horietan bat baino gehiagotan erabiltzen zen String bat zegoen, hortaz, egin dena String horiek aldagaietan jartzea da. Horri esker, String-a aldatu nahiko bagenu bakarrik behin egin beharko genuke. Aldagai horiek ‘ganarPartidoString’, ‘winMatchString’ eta ‘partiduaIrabaziString’ dira.
- Egilea: Iñigo Gil

- Hasierako kodea.

```
public void initializeDB(){


db.getTransaction().begin();
            try {

              . . .


                    db.persist(team1);
                    db.persist(team2);
                    db.persist(team3);
                    db.persist(team4);
                    db.persist(team5);
                    db.persist(team6);
                    db.persist(team7);
                    db.persist(team8);
                    db.persist(team9);

db.persist(team10);

db.persist(team11);

db.persist(team12);

db.persist(team13);

db.persist(team14);

db.persist(team15);

db.persist(team16);

db.persist(team17);

db.persist(team18);

db.persist(team19);

db.persist(team20);

db.persist(team21);

db.persist(team22);

db.persist(team23);

db.persist(team24);
```

```
db.persist(team25);

db.persist(team26);

db.persist(team27);

db.persist(team28);

db.persist(team29);

db.persist(team30);

db.persist(team31);

db.persist(team32);

                    db.persist(apA1);
                    db.persist(apA3);
                    db.persist(apA4);
                    db.persist(apA5);
                    db.persist(apA6);
                    db.persist(apA7);
                    db.persist(apA8);
                    db.persist(apA9);
                    db.persist(apA10);
                    db.persist(apA11);
                    db.persist(apA12);
                    db.persist(apA13);

                    db.persist(q1);
                    db.persist(q2);
                    db.persist(q3);
                    db.persist(q4);
                    db.persist(q5);
                    db.persist(q6);
                    db.persist(q7);
                    db.persist(q8);
                    db.persist(q9);
                    db.persist(q10);
                    db.persist(q11);



                    db.persist(ev1);
                    db.persist(ev2);
                    db.persist(ev3);
                    db.persist(ev4);
                    db.persist(ev5);
                    db.persist(ev6);
                    db.persist(ev7);
                    db.persist(ev8);
```

```
                    db.persist(ev9);
                    db.persist(ev10);
                    db.persist(ev11);
                    db.persist(ev12);
                    db.persist(ev13);
                    db.persist(ev14);
                    db.persist(ev15);
                    db.persist(ev16);
                    db.persist(ev17);
                    db.persist(ev18);
                    db.persist(ev19);
                    db.persist(ev20);

                    db.persist(ev21);
                    db.persist(ev22);
                    db.persist(ev23);
                    db.persist(ev24);
                    db.persist(ev25);
                    db.persist(ev26);
                    db.persist(ev27);

                    db.persist(sp1);
                    db.persist(sp2);
                    db.persist(sp3);

                    db.persist(ad1);
                    db.persist(reg1);
                    db.persist(reg2);
                    db.persist(reg3);
                    db.persist(reg4);


db.persist(quote3);

db.persist(quote2);

db.persist(quote1);

db.persist(quote4);

db.persist(quote5);

db.persist(quote6);

db.persist(quote7);

db.persist(quote8);

db.persist(quote9);

db.persist(quote10);
```

```
db.persist(quote11);

db.persist(quote12);

                    db.persist(ap1);
                    db.persist(ap2);
                    db.persist(ap3);
                    db.persist(ap4);
                    db.persist(ap5);
                    db.persist(ap6);
                    db.persist(ap7);
                    db.persist(ap8);
                    db.persist(ap9);
                    db.persist(ap10);
                    db.persist(ap11);
                    db.persist(ap12);
                    db.persist(ap13);
                    db.persist(ap14);

                    db.persist(t1);
                    db.persist(t3);
                    db.persist(t4);
                    db.persist(t5);
                    db.persist(t6);
                    db.persist(t7);
                    db.persist(t8);
                    db.persist(t9);
                    db.persist(t10);
                    db.persist(t11);
                    db.persist(t12);



db.getTransaction().commit();


this.DiruaSartu(reg1, 50.0, new
Date(), "DiruaSartu");

this.DiruaSartu(reg2, 50.0, new
Date(), "DiruaSartu");

this.DiruaSartu(reg3, 50.0, new
Date(), "DiruaSartu");

this.DiruaSartu(reg4, 50.0, new
Date(), "DiruaSartu");


System.out.println("Db
initialized");
```

```
            }
            catch (Exception e){

e.printStackTrace();
            }
      }
```

- Errefaktorizatuko kodea.

```
public void initializeDB(){

            db.getTransaction().begin();
            try {

             . . .

                  Team[] gordetzekoTeams = {
                              team1, team2, team3, team4, team5, team6,
team7, team8, team9, team10,
                              team11, team12, team13, team14, team15,
team16, team17, team18, team19, team20,
                              team21, team22, team23, team24, team25,
team26, team27, team28, team29, team30,
                              team31, team32
                  };
                  Event[] gordetzekoEvents = {
                              ev1, ev2, ev3, ev4, ev5, ev6, ev7, ev8, ev9,
ev10,
                              ev11, ev12, ev13, ev14, ev15, ev16, ev17,
ev18, ev19, ev20,
                              ev21, ev22, ev23, ev24, ev25, ev26, ev27
                  };

                  Question[] gordetzekoQuestions = {
                         q1, q2, q3, q4, q5, q6, q7, q8, q9, q10,
                         q11};

                  this.dbGorde(gordetzekoEvents, gordetzekoQuestions,
gordetzekoTeams);


                  db.getTransaction().commit();

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

                  System.out.println("Db initialized");
            }
```

```
            catch (Exception e){
                  e.printStackTrace();
            }
      }

      private void dbGorde(Event[] pGordetzekoEvents, Question[]
pGordetzekoQuestions, Team[] pGordetzekoTeams) {
            for(Question q:pGordetzekoQuestions) db.persist(q);
            for(Event ev:pGordetzekoEvents) db.persist(ev);
            for(Team t:pGordetzekoTeams) db.persist(t);
      }
```

- Egindako errefaktorizazioren deskribapena.

  “Write short units of code”
  Metodo hau luzeegia zen. Lerro asko erabiltzen ziren datu-basean “db.persist()” metodoari deitzeko. Hortaz, dei horiek beste metodo batera atera dira eta metodo originalaren lerro kopurua txikitu da.
- Egilea: Iñigo Gil

  - Hasierako kodea.

```
public void EmaitzakIpini(Quote quote) throws EventNotFinished{

            Quote q = db.find(Quote.class, quote);
            String result = q.getForecast();

            if(new
Date().compareTo(q.getQuestion().getEvent().getEventDate())<0)
                  throw new EventNotFinished();
            Vector<Apustua> listApustuak = q.getApustuak();
            db.getTransaction().begin();
            Question que = q.getQuestion();
            Question question = db.find(Question.class, que);
            question.setResult(result);
            for(Quote quo: question.getQuotes()) {
```

```
                    for(Apustua apu: quo.getApustuak()) {

                          Boolean b=apu.galdutaMarkatu(quo);
                          if(b) {
                                apu.getApustuAnitza().setEgoera("galduta");
                          }else {
                                apu.setEgoera("irabazita");
                          }
                    }
              }
              db.getTransaction().commit();
              for(Apustua a : listApustuak) {
                    db.getTransaction().begin();
                    Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
                    db.getTransaction().commit();
                    if(bool) {
                          this.ApustuaIrabazi(a.getApustuAnitza());
                    }
              }
        }
```

- Errefaktorizatuko kodea.

```
public void EmaitzakIpini(Quote quote) throws EventNotFinished{

              Quote q = db.find(Quote.class, quote);
              String result = q.getForecast();

              if(new
Date().compareTo(q.getQuestion().getEvent().getEventDate())<0)
                    throw new EventNotFinished();
              Vector<Apustua> listApustuak = q.getApustuak();
              db.getTransaction().begin();
              Question que = q.getQuestion();
              Question question = db.find(Question.class, que);
              question.setResult(result);
              this.apustuakMarkatu(question);
              db.getTransaction().commit();
              for(Apustua a : listApustuak) {
                    db.getTransaction().begin();
                    Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
                    db.getTransaction().commit();
                    if(bool) {
                          this.ApustuaIrabazi(a.getApustuAnitza());
                    }
              }
        }
        private void apustuakMarkatu(Question question) {
              for(Quote quo: question.getQuotes()) {
                    for(Apustua apu: quo.getApustuak()) {

                          Boolean b=apu.galdutaMarkatu(quo);
                          if(b) {
```

```
                        apu.getApustuAnitza().setEgoera("galduta");
                    }else {
                        apu.setEgoera("irabazita");
                    }
                }
            }
        }
```

- Egindako errefaktorizazioren deskribapena.
  "Write simple units of code"
  Metodo honen konplexutasun ziklomatikoa nahiko altua zen. Metodo auxiliar bat sortu da hau txikitzeko. "apustuakMarkatu()" metodo auxiliarraren bitartez konplexutasun originala txikitu da.
- Egilea: Iñigo Gil

    - Hasierako kodea.

```
public boolean jarraitu(Registered jabea, Registered jarraitua, Double
limit) {
            Boolean b=false;
            Registered jarraitu = (Registered) db.find(Registered.class,
jarraitua.getUsername());
            Registered harpideduna = (Registered)
db.find(Registered.class, jabea.getUsername());
            if(!harpideduna.getJarraitutakoLista().contains(jarraitu)) {
                db.getTransaction().begin();
                Jarraitzailea jar = new Jarraitzailea(harpideduna,
jarraitu);
                harpideduna.addJarraitutako(jar);
                jarraitu.addJarraitzailea(jar);
                b=true;
                db.persist(jar);
                harpideduna.setDiruLimitea(limit);
                db.getTransaction().commit();
            }
            return b;
      }
```

- Errefaktorizatuko kodea.

```
public boolean jarraitu(JarraituParameter parameterObject) {
            Boolean b=false;
            Registered jarraitu = (Registered) db.find(Registered.class,
parameterObject.jarraitua.getUsername());
```

```
            Registered harpideduna = (Registered)
db.find(Registered.class, parameterObject.jabea.getUsername());
            if(!harpideduna.getJarraitutakoLista().contains(jarraitu)) {
                  db.getTransaction().begin();
                  Jarraitzailea jar = new Jarraitzailea(harpideduna,
jarraitu);
                  harpideduna.addJarraitutako(jar);
                  jarraitu.addJarraitzailea(jar);
                  b=true;
                  db.persist(jar);
                  harpideduna.setDiruLimitea(parameterObject.limit);
                  db.getTransaction().commit();
            }
            return b;
      }
```

- Egindako errefaktorizazioren deskribapena.
  “Keep unit interfaces small”
  Erabili dugun klasean 4 parametro baino gehiago erabiltzen dituen metodorik ez egon arren, hiru parametro jasotzen zuen metodo hau parametro bakarrera gutxitu dugu.
- Egilea: Iñigo Gil