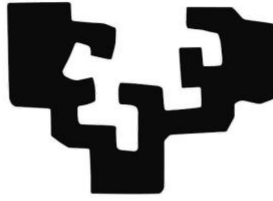


eman ta zabal zazu



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

Proiektuan egin beharreko ErrefaktORIZAZIOAK



informatika
fakultatea

facultad de
informática

EgileaK: Aritz Plazaola Cortabarria eta Aitor Velaz Nicolas

Data: 2022/10/14

Irakaslea: Jon Iturrioz


Irakasgaia: Software Ingeniaritza II

AURKIBIDEA

1. Aitorrek eginiko errefaktORIZAZIOAK.....	2
1.1 Write short units of code.....	2
1.2 Write simple units of code.....	3
1.3 Duplicate Code.....	5
1.4 Keep unit interfaces small.....	6
2. Aritzek eginiko errefaktORIZAZIOAK.....	9
2.1 Write short units of code.....	9
2.2 Write simple units of code.....	11
2.3 Duplicate Code.....	12
2.4 Keep unit interfaces small.....	14

1. "Write short units of code"

DataAccess.java

 Refactor this method to reduce its Cognitive Complexity from 17 to the 15 allowed. [+10 locations]

public boolean gertaeraEzabatu(Event ev) {} metodoa errefaktORIZATUKO dugu 15 linea kode baino gutxiago izan dezan hainbat zatitan banatuz. Horretarako query-ko kode oso hau metodo batean sartuko dugu eta metodoa banatu egingo dugu.

```
TypedQuery<Quote> Query = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEventNumber() =?1", Quote.class);
Query.setParameter(1, event.getEventNumber());
List<Quote> listQUO = Query.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);
        KirolEstatistikak ke=ap1.getUser().kirolEstatistikakLortu(spo);
        ke.setKont(ke.getKont()-1);
        db.getTransaction().commit();
    }
}
```

ezabatzekoAldaketak() metodoa sortuko dugu eta hor sartuko dugu kodea. Honek parametro bezala eventua bat emango diogu kodean event.getEventNumber() metodarako beharrezkoa izango duelako hasieran lortu dugun "event" eventua. Beraz **refactor** → **extract method** aukeratuko dugu.

```
public void ezabatzekoAldaketak(Event event) {
    TypedQuery<Quote> Query = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEventNumber() =?1", Quote.class);
    Query.setParameter(1, event.getEventNumber());
    List<Quote> listQUO = Query.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);
            KirolEstatistikak ke=ap1.getUser().kirolEstatistikakLortu(spo);
            ke.setKont(ke.getKont()-1);
            db.getTransaction().commit();
        }
    }
}
```

Azkenean gertaera ezabatu metodoa honela geratuko zaigu "smell kodea" zuzenduz:

```

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 == false) {
        return false;
    } else if(new Date().compareTo(event.getEventDate())<0) {
        ezabatzekoAldaketak(ev);
    }
    db.getTransaction().begin();
    db.remove(event);
    db.getTransaction().commit();
    return true;
}

```

2. "Write simple units of code"

public boolean ApustuaEgin(User u, Vector<Quote> quote, Double balioa, Integer apustuBikoitzaGalarazi){} metodoa zuzenduko dugu "branch fo points" kantitatea gutxituz. Horretarako **refactor** → **extract metodo** eginez kantitatea gutxituko dugu.

```

public boolean ApustuaEgin(User u, Vector<Quote> quote, Double balioa, Integer apustuBikoitzaGalarazi) {
    Registered user = (Registered) db.find(User.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 kute = db.find(Quote.class, quo.getQuoteNumber());
            Apustua ap = new Apustua(apustuAnitza, kute);
            db.persist(ap);
            apustuAnitza.addApustua(ap);
            kute.addApustua(ap);
        }
        db.getTransaction().commit();
        db.getTransaction().begin();
        if(apustuBikoitzaGalarazi!=-1) {
            apustuBikoitzaGalarazi=apustuAnitza.getApustuAnitzaNumber();
        }
        apustuAnitza.setApustuKopia(apustuBikoitzaGalarazi);
    }
}

```



```

public boolean ApustuaEgin(User u, Vector<Quote> quote, Double balioa, Integer apustuBikoitzaGalarazi) {
    Registered user = (Registered) db.find(User.class, u.getUsername());
    Boolean b;
    if(user.getDirukop()>=balioa) {
        db.getTransaction().begin();
        ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
        db.persist(apustuAnitza);
        apustuBikoitzaGalarazi = apustuaEzarri(quote, apustuBikoitzaGalarazi, apustuAnitza);
    }
    apustuAnitza.setApustuKopia(apustuBikoitzaGalarazi);
}

```

Lehenengo for eta if-a metodo batean sartuko ditugu.

```
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()==apu.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;
```

Ondoren hurrengo for-a beste metodo batean sartuko dugu:

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

        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.getApustuaNumber());
            Quote q = db.find(Quote.class, apu.getKuota().getQuoteNumber());
            Sport spo =q.getQuestion().getEvent().getSport();
            spo.setApustuKantitatea(spo.getApustuKantitatea()+1);
            if(!user.containsKirola(spo)) {
                KirolEstatistikak ke=new KirolEstatistikak(user, spo);
                ke.setKont(1);
                user.addKirolEstatistikak(ke);
                spo.addKirolEstatistikak(ke);
            }else {
                KirolEstatistikak ke=user.kirolEstatistikakLortu(spo);
                ke.eguneratuKont(1);
            }
        }
        user.addTransaction(t);
    }
}
```

eta azkeneko for barruan dagoen for-a ere bai:

```
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()==apu.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;
```

Azkenean metodoa honela geratuko zaigu:

```

public boolean ApustuaEgin(User u, Vector<Quote> quote, Double balioa, Integer apustuBikoitzaGalarazi) {
    Registered user = (Registered) db.find(User.class, u.getUsername());
    Boolean b;
    if(user.getDirukop()>=balioa) {
        db.getTransaction().begin();
        ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
        db.persist(apustuAnitza);
        apustuBikoitzaGalarazi = apustuaEzarri(quote, apustuBikoitzaGalarazi, apustuAnitza);//refactor

        apustuAnitza.setApustuKopia(apustuBikoitzaGalarazi);
        user.updateDirukontua(-balioa);
        Transaction t = new Transaction(user, balioa, new Date(), apustuaEgin);
        user.addApustuAnitza(apustuAnitza);
        apusutakLortu(user, apustuAnitza);//refactor

        user.addTransaction(t);
        db.persist(t);
        db.getTransaction().commit();
        for(Jarraitzailea reg:user.getJarraitzaileLista()) {
            Jarraitzailea erab=db.find(Jarraitzailea.class, reg.getJarraitzaileaNumber());
            b=true;
            b = apustuAnitzakKonprobatu(b, apustuAnitza, erab);//refactor
            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;
}

```

3. "Duplicate code"

DataAccess.java Define a constant instead of duplicating this literal "Zeinek irabaziko du partidua?" 3 times. [+3 locations]

Ikusi daiteke "Zeinek irabaziko du partidua?" String-a hainbat alditan errepikatzen dela kodean. Hau zuzentzeko kodea berrirabili beharko dugu kode errepikatuaren aldagai lokal bat sortuz. Honetarako **refactor** → **extract local variable** aukeratuko dugu string-ean hau zuzentzeko.

```

Duplication
q1=ev1.addQuestion( 1 "Zeinek irabaziko du partidua?",1);
q2=ev1.addQuestion("Zeinek sartuko du lehenengo gola?",2);
Duplication
q3=ev11.addQuestion( 2 "Zeinek irabaziko du partidua?",1);
q4=ev11.addQuestion("Zenbat gol sartuko dira?",2);
Duplication
q5=ev17.addQuestion( 3 "Zeinek irabaziko du partidua?",1);

```



```

String question = "Zeinek irabaziko du partidua?";
Duplication
q1=ev1.addQuestion(question,1);
q2=ev1.addQuestion("Zeinek sartuko du lehenengo gola?",2);
Duplication
q3=ev11.addQuestion(question,1);
q4=ev11.addQuestion("Zenbat gol sartuko dira?",2);
Duplication
q5=ev17.addQuestion(question,1);

```

4. "Keep unit interfaces small" → event edo mezuakrenderer klasean dago bat!!!!

```
@Override
public Component getListCellRendererComponent(JList<? extends MezuakContainer> list, MezuakContainer mezua, int index,
        boolean isSelected, boolean cellHasFocus) {
```

Aurreko kodean, MezuakRenderer klasean, ikusi genezake metodoak bost parametro dituela, 4ko limitea baino bat gehiago. Gainera horietako bi *Boolean* motako parametroak dira. Honetarako klase berri bat sortuko dut:

```
package domain;

public class SelectedOrFocus {

    Boolean isSelected;
    Boolean cellHasFocus;

    public SelectedOrFocus(boolean isSelected, boolean cellHasFocus) {
        super();
        this.isSelected=isSelected;
        this.cellHasFocus=cellHasFocus;
    }
    public Boolean getisSelected() {
        return isSelected;
    }
    public void setisSelected(Boolean isSelected) {
        this.isSelected = isSelected;
    }
    public Boolean getCellHasFocus() {
        return cellHasFocus;
    }
    public void setCellHasFocus(Boolean cellHasFocus) {
        this.cellHasFocus = cellHasFocus;
    }
}
```

Aurreko metodoko bi boolean parametro clase batean juntatzeko eta parametro kopurua 5 izan beharrean 4 izateko.

```
public Component getListCellRendererComponent(JList<? extends
MezuakContainer> list, MezuakContainer mezua, int index,
        boolean isSelected, boolean cellHasFocus) {

    setText(mezua.toString());
    String code = "m1";

    ImageIcon imageIcon = new
    ImageIcon(".\\src/main/resources\\data\\"+mezua.getMessage().isIrakurrita(
)+".png"); // load the image to a ImageIcon
    Image image = imageIcon.getImage(); // transform it
```

```

        Image newimg = image.getScaledInstance(25, 25,
Image.SCALE_SMOOTH); // scale it the smooth way
        ImageIcon imageIcon = new ImageIcon(newimg);
        setIcon(imageIcon);

        if (mezua.getMessage().isIrakurrita() == false &&
!isSelected) {

setBackground(list.getSelectionBackground().PINK);

setForeground(list.getSelectionForeground().BLACK);

        } else {
            if(mezua.getMessage().isIrakurrita()) {
                setBackground(list.getBackground());
                setForeground(list.getForeground());
            }
            if(isSelected) {

setBackground(list.getSelectionBackground().gray);

setForeground(list.getSelectionForeground().white);
            }
        }
        return this;
    }

```

Klasea sortu ondoren horrela geratzen da metodoa:

```

    public Component getListCellRendererComponent(JList<? extends
MezuakContainer> list, MezuakContainer mezua, int index,
        SelectedOrFocus SelectedOrFocus) {

        setText(mezua.toString());
        String code = "m1";

        ImageIcon imageIcon = new
ImageIcon(".\\src/main/resources\\data\\"+mezua.getMessage().isIrakurrita(
)+".png"); // load the image to a ImageIcon
        Image image = imageIcon.getImage(); // transform it
        Image newimg = image.getScaledInstance(25, 25,
Image.SCALE_SMOOTH); // scale it the smooth way
        ImageIcon imageIcon = new ImageIcon(newimg);
        setIcon(imageIcon);
    }

```



```

        if (mezua.getMessage().isIrakurrita() == false &&
!SelectedOrFocus.isSelected) {

setBackground(list.getSelectionBackground().PINK);

setForeground(list.getSelectionForeground().BLACK);

        } else {
            if(mezua.getMessage().isIrakurrita()) {
                setBackground(list.getBackground());
                setForeground(list.getForeground());
            }
            if(SelectedOrFocus.isSelected) {

setBackground(list.getSelectionBackground().gray);

setForeground(list.getSelectionForeground().white);
            }
        }
        return this;
    }

```

1. "Write short units of code"

Hasiera kodea:

```
public void ezabatzekoAldaketak(Event event) {
    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);
            KirolEstatistikak
ke=ap1.getUser().kirolEstatistikakLortu(spo);
            ke.setKont(ke.getKont()-1);
            db.getTransaction().commit();
        }
    }
}
```

Arazoa:

Lehenago Aitorrek errefaktoretzatutako **ezabatzekoAldaketak** metodoak 15 lerro baino gehiago ditu, eta hori ez dugu nahi, *Bad Smell* bat baita. Horretarako metodoko zati bat beste metodo baten bihurtu dut "Extract method" erabiliz. Horrela bi metodo desberdin lortu ditut: **ezabatzekoAldaketak** eta **ezabatzekoAldaketakGorde**, baina, hala ere, **ezabatzekoAldaketakGorde** metodoak 16 lerro ditu, beraz "Extract method" bigarrenaz aplikatu behar izan dut.

ezabatzekoAldaketakGorde metodoak if-else if bat zuen barruan, eta bigarrenko errefaktORIZAZIOAN, if-esle if blokea beste metodo baten bilakatu dut, **apustuarenEgoeraAztertu**.

Bukaerako kodea:

```
public void ezabatzekoAldaketak(Event event) {
    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));
        ezabatzekoAldaketakGorde(quo);
    }
}
```

```
private void ezabatzekoAldaketakGorde(Quote quo) {
    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();
        apustuarenEgoeraAztertu(ap1); //refactored
        db.getTransaction().begin();
        Sport spo =quo.getQuestion().getEvent().getSport();
        spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
        KirolEstatistikak
ke=ap1.getUser().kirolEstatistikakLortu(spo);
        ke.setKont(ke.getKont()-1);
        db.getTransaction().commit();
    }
}
```

```
private void apustuarenEgoeraAztertu(ApustuAnitza ap1) {
    if(ap1.getApustuak().isEmpty() &&
!ap1.getEgoera().equals("galduta")) {
        this.apustuaEzabatu(ap1.getUser(), ap1);
    }else if(!ap1.getApustuak().isEmpty() &&
ap1.irabazitaMarkatu()){
```

```

        this.ApustuaIrabazi(ap1);
    }
}

```

2. "Write simple units of code"

Hasiera 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 == false) {
        return false;
    }else if(new Date().compareTo(event.getEventDate())<0) {
        ezabatzekoAldaketak(event);
    }
    db.getTransaction().begin();
    db.remove(event);
    db.getTransaction().commit();
    return true;
}

```

Arazoa:

gertaeraEzabatu metodoaren konplexutasun ziklomatikoa 4 baino handiagoa denez gutxitu beharra dago. Horretarako metodoan dagoen for begizta metodo berri baten jartzea erabaki da.

Bukaerako kodea:

```

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

```

```

        if(resultB == false) {
            return false;
        }else if(new Date().compareTo(event.getEventDate())<0) {
            ezabatzekoAldaketak(event);
        }
        db.getTransaction().begin();
        db.remove(event);
        db.getTransaction().commit();
        return true;
    }

    private boolean emaitzaHutsa(boolean resultB, List<Question> listQ) {
        for(Question q : listQ) {
            if(q.getResult() == null) {
                resultB = false;
            }
        }
        return resultB;
    }
}

```

3. “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");
```

Arazoa:

 Define a constant instead of duplicating this literal "ApustuaEgin" 12 times. [+12 locations]

Arazoa software mantenuan ohikoa den arazo bat da, kode errepikamena. Kodea asko errepikatzea **Bad Smell** bat da eta hori ekiditeko aldagai global bat egin dut. Horrela, kodea errepikatu beharrean, kodea berrerabili egiten da.

Kode berria:

```
static final String apustuaEgin = "ApustuaEgin";

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);
```

4. "Keep unit interfaces small"

Hasierako kodea:

```
public boolean mezuaBidali(User igorlea, String hartzailea, String
titulo, String test, Elkarrizketa elkarrizketa) {
    User igorle = db.find(User.class, igorlea.getUsername());
    User hartzaile = db.find(User.class, hartzailea);
    Elkarrizketa elk=null;
    if(hartzaile==null) {
        return false;
    }else {
        db.getTransaction().begin();
        Message m = new Message(igorle, hartzaile, test);
        db.persist(m);
        if(elkarrizketa!=null) {
            elk = db.find(Elkarrizketa.class,
elkarrizketa.getElkarrizketaNumber());
        }else {
            elk= new Elkarrizketa(titulo, igorle, hartzaile);
            db.persist(elk);
            m.setElkarrizketa(elk);
            igorle.addElkarrizketak(elk);
            hartzaile.addElkarrizketak(elk);
        }
        elk.addMezua(m);
        igorle.addBidalitakoMezuak(m);
        hartzaile.addJasotakoMezuak(m);
        db.getTransaction().commit();
        return true;
    }
}
```

Arazoa:

Aurreko kodean ikusi genezake metodoak bost parametro dituela, 4ko limitea baino bat gehiago. Gainera horietako bi *String* motako parametroak dira.

Bad Smell hau kentzeko klase berri bat sortu dut **domain** paketeen Bidaltzekoa izenarekin. Klaseak bi atributu ditu izenburua eta testua, lehengo kodean zeuden bi parametroak. Horrela **mezuaBidali** metodoak lau parametro izango ditu. Ondoren, DataAccesseko eta BLFacadeko erreferentziak zuzendu ditut.

Bukaerako kodea:

```
class Bidaltzekoa{

    String izenburua;
    String testua;

    public Bidaltzekoa(String izenburua, String testua) {
        super();
        this.izenburua = izenburua;
        this.testua = testua;
    }

    public String getIzenburua() {
        return izenburua;
    }
    public void setIzenburua(String izenburua) {
        this.izenburua = izenburua;
    }
    public String getTestua() {
        return testua;
    }
    public void setTestua(String testua) {
        this.testua = testua;
    }
}

public boolean mezuaBidali(User igorlea, String hartzailea, Bidaltzekoa
bidaltzekoak, Elkarriketa elkarriketa) {
    User igorle = db.find(User.class, igorlea.getUsername());
    User hartzaile = db.find(User.class, hartzailea);
    Elkarriketa elk=null;
    if(hartzaile==null) {
        return false;
    }else {
        db.getTransaction().begin();
        Message m = new Message(igorle, hartzaile,
bidaltzekoak.getTestua());
        db.persist(m);
        if(elkarriketa!=null) {
            elk = db.find(Elkarriketa.class,
elkarriketa.getElkarriketaNumber());
        }else {
            elk= new Elkarriketa(bidaltzekoak.getIzenburua(),
igorle, hartzaile);
```



```
        db.persist(elk);
        m.setElkarrizketa(elk);
        igorle.addElkarrizketak(elk);
        hartzaile.addElkarrizketak(elk);
    }
    elk.addMezua(m);
    igorle.addBidalitakoMezuak(m);
    hartzaile.addJasotakoMezuak(m);
    db.getTransaction().commit();
    return true;
}
}
```