Ingeniería del Software II

Refactorización

Informe

Alexandra Aleina Pelipian
Ignacio Ayaso Hernández
Josu Lorenzo Hernández

Facultad de Informática
Grado de Ingeniería Informática

14 de Octubre, 2022

"Write short units of code"

1. Código Inicial

```java
/**
 * Se copia un evento existente en una fecha dada
 * @param e el evento a copiar.
 * @param date la fecha (distinta a la del evento 'e') en la que se copia el evento 'e'.
 * @return devuelve true si se ha copiado con éxito el evento 'e' en la fecha 'date'.
 */
    public boolean gertaerakKopiatu(Event e, Date date) {
        Boolean b=false;
        Event gertaera = db.find(Event.class, e.getEventNumber());
        db.getTransaction().begin();


        TypedQuery<Event> query = db.createQuery("SELECT ev FROM Event ev WHERE ev.getDescription()=?1 and ev.getEventDate()=?2",Event.class);
        query.setParameter(1,gertaera.getDescription());
        query.setParameter(2, date);
        if(query.getResultList().isEmpty()) {
            b=true;
            String[] taldeak = gertaera.getDescription().split("-");
            Team lokala = new Team(taldeak[0]);
            Team kanpokoa = new Team(taldeak[1]);
            Event gertKopiatu = new Event(gertaera.getDescription(), date, lokala, kanpokoa);
            gertKopiatu.setSport(gertaera.getSport());
            gertaera.getSport().addEvent(gertKopiatu);
            db.persist(gertKopiatu);
            for(Question q : gertaera.getQuestions()) {
                Question que= new Question(q.getQuestion(), q.getBetMinimum(), gertKopiatu);
                gertKopiatu.listaraGehitu(que);
                Question galdera = db.find(Question.class, q.getQuestionNumber());
                db.persist(que);
                for(Quote k: galdera.getQuotes()) {
                    Quote kuo= new Quote(k.getQuote(), k.getForecast(), que);
                    que.listaraGehitu(kuo);
                    db.persist(kuo);
                }
            }
        }
        db.getTransaction().commit();
        return b;
    }
```

2. Código refactorizado

```java
/**
 * Se copia un evento existente en una fecha dada
 * @param e el evento a copiar.
 * @param date la fecha (distinta a la del evento 'e') en la que se copia el evento 'e'.
 * @return devuelve true si se ha copiado con éxito el evento 'e' en la fecha 'date'.
 */
    public boolean gertaerakKopiatu(Event e, Date date) {
        Boolean b=false;
        Event gertaera = db.find(Event.class, e.getEventNumber());
        db.getTransaction().begin();


        TypedQuery<Event> query = db.createQuery("SELECT ev FROM Event ev WHERE ev.getDescription()=?1 and ev.getEventDate()=?2",Event.class);
        query.setParameter(1,gertaera.getDescription());
        query.setParameter(2, date);
        if(query.getResultList().isEmpty()) {
            b=true;
            eventCopy(date, gertaera);
        }
        db.getTransaction().commit();
        return b;
    }
```

3. Descripción del error y explicación de la refactorización

Dado el código inicial, el número de líneas del método supera con creces el límite descrito en el libro “"Building Maintainable Software”. Para facilitar la legibilidad del código, se han extraído bloques de código, refactorizándolos de la siguiente manera:
La llamada al método ‘eventCopy’ engloba los bucles anidados que, a su vez, están refactorizados en dos bloques más pequeños, de tal forma que el código sea más legible.

```
private void eventCopy(Date date, Event gertaera) {
    String[] taldeak = gertaera.getDescription().split("-");
    Team lokala = new Team(taldeak[0]);
    Team kanpokoa = new Team(taldeak[1]);
    Event gertKopiatu = new Event(gertaera.getDescription(), date, lokala, kanpokoa);
    gertKopiatu.setSport(gertaera.getSport());
    gertaera.getSport().addEvent(gertKopiatu);
    db.persist(gertKopiatu);
    questionsCopy(gertaera, gertKopiatu);
}

private void questionsCopy(Event gertaera, Event gertKopiatu) {
    for(Question q : gertaera.getQuestions()) {
        Question que= new Question(q.getQuestion(), q.getBetMinimum(), gertKopiatu);
        gertKopiatu.listaraGehitu(que);
        Question galdera = db.find(Question.class, q.getQuestionNumber());
        db.persist(que);
        quoteCopy(que, galdera);
    }
}

private void quoteCopy(Question que, Question galdera) {
    for(Quote k: galdera.getQuotes()) {
        Quote kuo= new Quote(k.getQuote(), k.getForecast(), que);
        que.listaraGehitu(kuo);
        db.persist(kuo);
    }
}
```

4. Miembro que ha realizado la refactorización

Alexandra Aleina Pelipian

---

1. Código Inicial

```
public void apustuaEzabatu(Registered user1, ApustuAnitza ap) {
    Registered user = (Registered) db.find(Registered.class, user1.getUsername());
    ApustuAnitza apustuAnitza = db.find(ApustuAnitza.class, ap.getApustuAnitzaNumber());
    db.getTransaction().begin();
    user.updateDiruKontua(apustuAnitza.getBalioa());
    Transaction t = new Transaction(user, apustuAnitza.getBalioa(), new Date(), "ApustuaEzabatu");
    user.addTransaction(t);
    db.persist(t);
    user.removeApustua(apustuAnitza);
    int i;
    for(i=0; i<apustuAnitza.getApustuak().size(); i++) {
        apustuAnitza.getApustuak().get(i).getKuota().removeApustua(apustuAnitza.getApustuak().get(i));
        Sport spo =apustuAnitza.getApustuak().get(i).getKuota().getQuestion().getEvent().getSport();
        spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
    }
    db.remove(apustuAnitza);
    db.getTransaction().commit();
}
```

2. Código refactorizado

```
public void apustuaEzabatu(Registered user1, ApustuAnitza ap) {
    Registered user = (Registered) db.find(Registered.class, user1.getUsername());
    ApustuAnitza apustuAnitza = db.find(ApustuAnitza.class, ap.getApustuAnitzaNumber());
    db.getTransaction().begin();
    user.updateDiruKontua(apustuAnitza.getBalioa());
    Transaction t = new Transaction(user, apustuAnitza.getBalioa(), new Date(), "ApustuaEzabatu");
    user.addTransaction(t);
    db.persist(t);
    user.removeApustua(apustuAnitza);
    auxApustuaEzabatu(apustuAnitza);
    db.remove(apustuAnitza);
    db.getTransaction().commit();
}

private static void auxApustuaEzabatu(ApustuAnitza apustuAnitza) {
    for(int i=0; i<apustuAnitza.getApustuak().size(); i++) {
        apustuAnitza.getApustuak().get(i).getKuota().removeApustua(apustuAnitza.getApustuak().get(i));
        Sport spo = apustuAnitza.getApustuak().get(i).getKuota().getQuestion().getEvent().getSport();
        spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
    }
}
```

3. Descripción del error y explicación de la refactorización

El error indica que el método excede la cantidad de líneas máximas recomendadas que tenga el programa, lo mejor es partir el programa en dos creando un método auxiliar.

4. Miembro que ha realizado la refactorización

Ignacio Ayaso Hernández

---

1. Código Inicial

```
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;
}
```

2. Código refactorizado

```
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) gertaerakSortuAuxiliar(description, eventDate, spo);

    } else
        return false;

    db.getTransaction().commit();
    return b;
}

private void gertaerakSortuAuxiliar(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);
}
```

3. Descripción del error y explicación de la refactorización

El código original presentaba un rama if en la que se realizaba un proceso que podría ser externalizado, de cara a mejorar la legibilidad del código. Por lo tanto, dado que lo que se hace dentro del if no altera la ejecución del método, se ha extraído y se ha llevado a un método auxiliar.

4. Miembro que ha realizado la refactorización

Josu Lorenzo Hernández

"Write simple units of code"

1. Código Inicial

```
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());
        }
    }
}
```

2. Código refactorizado

```
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);
    galdutaMarkatuRefact(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());
        }
    }
}
```

3. Descripción del error y explicación de la refactorización

Ya que el método ‘gertaerakKopiatu’ tenía complejidad ciclomática de 4, he optado por refactorizar el método ‘EmaitzakIpini’, cuya complejidad ciclomática era de 7.

Tras extraer el primer bucle en un nuevo método ‘galdutaMarkatuRefact’, la complejidad ciclomática del método principal ‘EmaitzakIpini’ es de 4. El método extraído es el siguiente:

```
private void galdutaMarkatuRefact(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");
            }
        }
    }
}
```

4. **Miembro que ha realizado la refactorización**

   Alexandra Aleina Pelipian

---

1. Código Inicial

```
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;
    }

}
```

2. Código refactorizado

```
public boolean ApustuaEgin(Registered u, Vector<Quote> quote, Double balioa, Integer apustuBikoitzaGalarazi) {
    Registered user = (Registered) db.find(Registered.class, u.getUsername());
    if(user.getDirukop()>=balioa) {
        ApustuAnitza apustuAnitza = auxFTApustuaEgin(user, balioa, quote);
        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);
        auxFor1(apustuAnitza);
        user.addTransaction(t);
        db.persist(t);
        db.getTransaction().commit();

        auxFors(user,apustuAnitza, balioa, quote, apustuBikoitzaGalarazi);
        return true;
    }else{
        return false;
    }

}
```

```
private ApustuAnitza auxFTApustuaEgin(Registered user, Double balioa, Vector<Quote> quote) {
    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();

    return apustuAnitza;
}

private void auxFor1(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 void auxFors(Registered user, ApustuAnitza apustuAnitza, Double balioa, Vector<Quote> quote, Integer apustuBikoitzaGalarazi) {
    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);
            }
        }
    }
}
```

3. Descripción del error y explicación de la refactorización

Esta vez el error se trata de un exceso de instrucciones tanto de “if”s como de “for”s, aunque claro cada método tiene su función, así que en vez de borrarlos, lo mejor es añadir métodos auxiliares que liberen cierta carga de trabajo al método “apustuaEgin”

4. **Miembro que ha realizado la refactorización**

   Ignacio Ayaso Hernández

---

1. **Código Inicial**

```
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) gertaerakSortuAuxiliar(description, eventDate, spo);

    } else
        return false;

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

2. **Código refactorizado**

```
    public boolean gertaerakSortu(String description,Date eventDate, String sport) {
        db.getTransaction().begin();
        Sport spo =db.find(Sport.class, sport);
        if(spo!=null) {
            if(gertaerakSortuAuxiliar_LookForDuplicatedEvents(eventDate, description)) {
                gertaerakSortuAuxiliar(description, eventDate, spo);

                db.getTransaction().commit();
                return true;
            } else {
                db.getTransaction().commit();
                return false;
            }
        }
        db.getTransaction().commit();
        return false;
    }

    private boolean gertaerakSortuAuxiliar_LookForDuplicatedEvents(Date eventDate, String description) {
        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))
                return false;

        return true;
    }
```

3. **Descripción del error y explicación de la refactorización**

De cara a tener menos branches en el método “gertaerakSortu”, se ha creado un método auxiliar para simplificar y minimizar el número de ramas dentro del método original. Gracias a esto, se ha pasado de 4 a 2 branch points. Además, se ha eliminado una variable local llamada “b”, la cual era simplemente un flag a devolver mediante la instrucción return.

4. **Miembro que ha realizado la refactorización**

   Josu Lorenzo Hernández

"Duplicate code"

1. Código Inicial

```
private JButton getBoton3() {
    if (jButtonQueryQueries == null) {
        jButtonQueryQueries = new JButton();
        jButtonQueryQueries.setText(ResourceBundle.getBundle("Etiquetas").getString("QueryQueries"));
        jButtonQueryQueries.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent e) {
                JFrame a = new FindQuestionsGUI();

                a.setVisible(true);
            }
        });
    }
    return jButtonQueryQueries;
}


private JLabel getLblNewLabel() {
    if (jLabelSelectOption == null) {
        jLabelSelectOption = new JLabel(ResourceBundle.getBundle("Etiquetas").getString("SelectOption"));
        jLabelSelectOption.setFont(new Font("Tahoma", Font.BOLD, 13));
        jLabelSelectOption.setForeground(Color.BLACK);
        jLabelSelectOption.setHorizontalAlignment(SwingConstants.CENTER);
    }
    return jLabelSelectOption;
}
private void redibujar() {
    jLabelSelectOption.setText(ResourceBundle.getBundle("Etiquetas").getString("SelectOption"));
    jButtonQueryQueries.setText(ResourceBundle.getBundle("Etiquetas").getString("QueryQueries"));
    jButtonCreateQuery.setText(ResourceBundle.getBundle("Etiquetas").getString("CreateQuery"));
    this.setTitle(ResourceBundle.getBundle("Etiquetas").getString("MainTitle"));
}
```

2. Código refactorizado

```
private static final String ETIQUETAS = "Etiquetas";


private void redibujar() {
    jLabelSelectOption.setText(ResourceBundle.getBundle(ETIQUETAS).getString("SelectOption"));
    jButtonQueryQueries.setText(ResourceBundle.getBundle(ETIQUETAS).getString("QueryQueries"));
    jButtonCreateQuery.setText(ResourceBundle.getBundle(ETIQUETAS).getString("CreateQuery"));
    this.setTitle(ResourceBundle.getBundle(ETIQUETAS).getString("MainTitle"));
}
```

3. Descripción del error y explicación de la refactorización

En la clase ‘DataAccess’ no había más código duplicado que se pueda refactorizar, con lo cual he encontrado en la clase ‘MainGUI’ un detalle a mejorar, y es que el texto ‘Etiqueta’ aparece dentro de varias llamadas a la clase ResourceBundle. Para mejorar el código, he optado por crear una variable constante con el valor ‘Etiqueta’ y utilizarla en cada una de las llamadas.

4. Miembro que ha realizado la refactorización

Alexandra Aleina Pelipian

---

1. **Código Inicial**

```
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);

}
```

2. **Código refactorizado**

```
if (Locale.getDefault().equals(new Locale("es"))) {
    final String sus = "Â¿QuiÃ©n ganarÃ¡ el partido?";
    q1=ev1.addQuestion(sus, 1);
    q2=ev1.addQuestion("Â¿QuiÃ©n meterÃ¡ el primer gol?", 2);
    q3=ev11.addQuestion(sus, 1);
    q4=ev11.addQuestion("Â¿CuÃ¡ntos goles se marcarÃ¡n?", 2);
    q5=ev17.addQuestion(sus, 1);
    q6=ev17.addQuestion("Â¿HabrÃ¡ goles en la primera parte?", 2);

}
```

3. **Descripción del error y explicación de la refactorización**

Por último, este code Smell se basa en que estamos copiando y pegando la misma cadena de carácteres hasta un máximo de dos ocasiones, la solución a este code Smell es crear una variable que almacene este String e intercambiar el código repetido por la variable .

4. **Miembro que ha realizado la refactorización**

Ignacio Ayaso Hernández

---

1. **Código Inicial**

```
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");
```

2. **Código refactorizado**

```
final String diruaSartu = "DiruaSartu";
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);
```

3. **Descripción del error y explicación de la refactorización**

El método “DiruaSartu” consta de 4 parámetros, siendo el último un String. En el código original, se tenían strings literales hardcodeados en la llamada al método. Para mejorar el mantenimiento, lectura y funcionamiento del código, se ha refactorizado cambiando en cada llamada el parámetro por una variable constante que contiene el valor original.

4. **Miembro que ha realizado la refactorización**

Josu Lorenzo Hernández

"Keep unit interfaces small"

1. Código Inicial

```
public boolean gertaerakKopiatu(Event e, Date date) {
    Boolean b=false;
    Event gertaera = db.find(Event.class, e.getEventNumber());
    db.getTransaction().begin();


    TypedQuery<Event> query = db.createQuery("SELECT ev FROM Event ev WHERE ev.getDescription()=?1 and ev.getEventDate()=?2",Event.class);
    query.setParameter(1,gertaera.getDescription());
    query.setParameter(2, date);
    if(query.getResultList().isEmpty()) {
        b=true;
        eventCopy(date, gertaera);
    }
    db.getTransaction().commit();
    return b;
}
```

2. Código refactorizado

```
/**
 * Se copia un evento existente en una fecha dada
 * @param parameterObject parámetros de entrada del método (un evento y una fecha)
 * @return devuelve true si se ha copiado con éxito el evento 'e' en la fecha 'date'.
 */
   public boolean gertaerakKopiatu(GertaerakKopiatuParameter parameterObject) {
       Boolean b=false;
       Event gertaera = db.find(Event.class, parameterObject.e.getEventNumber());
       db.getTransaction().begin();


       TypedQuery<Event> query = db.createQuery("SELECT ev FROM Event ev WHERE ev.getDescription()=?1 and ev.getEventDate()=?2",Event.class);
       query.setParameter(1,gertaera.getDescription());
       query.setParameter(2, parameterObject.date);
       if(query.getResultList().isEmpty()) {
           b=true;
           eventCopy(parameterObject.date, gertaera);
       }
       db.getTransaction().commit();
       return b;
   }

public class GertaerakKopiatuParameter {
    public Event e;
    public Date date;

    public GertaerakKopiatuParameter(Event e, Date date) {
        this.e = e;
        this.date = date;
    }
}
```

3. Descripción del error y explicación de la refactorización

Para dar un ejemplo de una corrección adecuada de los parámetros que un método vaya a recibir, se ha creado una clase pública cuyos atributos representan los parámetros que inicialmente el método necesitaba. En nuestro caso, el objeto GertaerakKopiatuParameter tiene, como atributos, un objeto de tipo Event y una fecha Date.

4. Miembro que ha realizado la refactorización

Alexandra Aleina Pelipian

---

1. Código Inicial

```
private void auxFors(Registered user, ApustuAnitza apustuAnitza, Double balioa, Vector<Quote> quote, Integer apustuBikoitzaGalarazi) {
    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);
            }
        }
    }
}
```

2. Código refactorizado

```
private void auxFors(Registered user, Double balioa, Vector<Quote> quote, Integer apustuBikoitzaGalarazi) {
    Boolean b;
    ApustuAnitza apustuAnitza = auxFTApustuaEgin(user, balioa, quote);
    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);
            }
        }
    }
}
```

3. Descripción del error y explicación de la refactorización

El error se nos indica como un exceso de parámetros de entrada, ya que de entrada podemos ver que hay cinco parámetros, la solución ha sido que el parámetro “apustuAnitza” obtenga su valor dentro del método “auxFors” y así no tener que usarlo como un parámetro de entrada

4. Miembro que ha realizado la refactorización

   Ignacio Ayaso Hernández

---

1. Código Inicial

```
public boolean gertaerakSortu(String description,Date eventDate, String sport) {
    db.getTransaction().begin();
    Sport spo =db.find(Sport.class, sport);
    if(spo!=null) {
        if(gertaerakSortuAuxiliar_LookForDuplicatedEvents(eventDate, description)) {
            gertaerakSortuAuxiliar(description, eventDate, spo);

            db.getTransaction().commit();
            return true;
        } else {
            db.getTransaction().commit();
            return false;
        }
    }
    db.getTransaction().commit();
    return false;
}
```

2. Código refactorizado

```
public boolean gertaerakSortu(EventParam params) {
    db.getTransaction().begin();
    Sport spo =db.find(Sport.class, params.sport);
    if(spo!=null) {
        if(gertaerakSortuAuxiliar_LookForDuplicatedEvents(params.eventDate, params.description)) {
            gertaerakSortuAuxiliar(params.description, params.eventDate, spo);

            db.getTransaction().commit();
            return true;
        } else {
            db.getTransaction().commit();
            return false;
        }
    }
    db.getTransaction().commit();
    return false;
}
```

```
public class EventParam {
    public String description;
    public Date eventDate;
    public String sport;

    public EventParam(String pdescription, Date peventDate, String psport) {
        this.description = pdescription;
        this.eventDate = peventDate;
        this.sport = psport;
    }
}
```

3. Descripción del error y explicación de la refactorización

Se ha detectado que al método “gertaerakSortu” le llegan los parámetros que definirían un posible evento. Por lo tanto, en vez de pasar los tres parámetros independientemente, se ha creado una clase auxiliar de paso de parámetros. De esta forma, se pasa un objeto con la información, y la signatura y llamada al método se simplifica.

4. Miembro que ha realizado la refactorización

Josu Lorenzo Hernández