Rides proiektuan egin beharrreko errefaktorizazioak

Egilea: **Zuriñe**

- ***"Write short units of code"***

DataAccess.java klasean: This method has 19 lines, which is greater than the 15 lines authorized. Split it into smaller methods.

1. Hasierako kodea.

```
public boolean deleteAlert(int alertNumber) {
    try {
        db.getTransaction().begin();

        TypedQuery<Alert> query = db.createQuery("SELECT a FROM Alert a WHERE a.alertNumber = :alertNumber",
                Alert.class);
        query.setParameter("alertNumber", alertNumber);
        Alert alert = query.getSingleResult();

        Traveler traveler = alert.getTraveler();
        traveler.removeAlert(alert);
        db.merge(traveler);

        db.remove(alert);

        db.getTransaction().commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}
```

2. Errefaktorizatuko kodea.

```
public boolean deleteAlert(int alertNumber) {
    try {
        db.getTransaction().begin();
        searchAlertAndRemove(alertNumber);
        db.getTransaction().commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}

private void searchAlertAndRemove(int alertNumber) {
    TypedQuery<Alert> query = db.createQuery("SELECT a FROM Alert a WHERE a.alertNumber = :alertNumber", Alert.class);
    query.setParameter("alertNumber", alertNumber);
    Alert alert = query.getSingleResult();
    Traveler traveler = alert.getTraveler();
    traveler.removeAlert(alert);
    db.merge(traveler);
    db.remove(alert);
}
```

3. Egindako errefaktorizazioren deskribapena.

“Bad smell” esaten duenez bezala, kodea hain luzea ez izateko metodoa bitan banatu daiteke. Refactor>>Extract Method eginda, beste metodo bat sortzen da, bertan kodearen zati bat definitzeko eta metodo nagusiaren luzera murrizteko.

- ***"Write simple units of code"***

DataAccess.java klasean: Refactor this method to reduce its Cognitive Complexity from 23 to the 15 allowed. [+11 locations]

1. Hasierako kodea.

```
public void deleteUser(User us) {
    try {
            +1
        1   if (us.getMota().equals("Driver")) {
            List<Ride> rl = getRidesByDriver(us.getUsername());
                +2 (incl 1 for nesting)
            2   if (rl != null) {
                    +3 (incl 2 for nesting)
                3   for (Ride ri : rl) {
                    cancelRide(ri);
                }
            }
            Driver d = getDriver(us.getUsername());
            List<Car> cl = d.getCars();
                +2 (incl 1 for nesting)
            4   if (cl != null) {
                    +3 (incl 2 for nesting)
                5   for (int i = cl.size() - 1; i >= 0; i--) {
                    Car ci = cl.get(i);
                    deleteCar(ci);
                }
            }
        +1
        } 6  else {
            List<Booking> lb = getBookedRides(us.getUsername());
                +2 (incl 1 for nesting)
            7   if (lb != null) {
                    +3 (incl 2 for nesting)
                8   for (Booking li : lb) {
                    li.setStatus("Rejected");
                    li.getRide().setnPlaces(li.getRide().getnPlaces() + li.getSeats());
                }
            }
            List<Alert> la = getAlertsByUsername(us.getUsername());
                +2 (incl 1 for nesting)
            9   if (la != null) {
                    +3 (incl 2 for nesting)
                10   for (Alert lx : la) {
                    deleteAlert(lx.getAlertNumber());
                }
            }
        }
        db.getTransaction().begin();
        us = db.merge(us);
        db.remove(us);
        db.getTransaction().commit();
    +1
    } 11   catch (Exception e) {
        e.printStackTrace();
    }
}
```

4. Errefaktorizatuko kodea.

```
public void deleteUser(User us) {
    try {
        if (us.getMota().equals("Driver")) {
            deleteDriver(us);
        } else {
            deleteTraveler(us);
        }
        removeUserDB(us);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void removeUserDB(User us) {
    db.getTransaction().begin();
    us = db.merge(us);
    db.remove(us);
    db.getTransaction().commit();
}

private void deleteTraveler(User us) {
    List<Booking> lb = getBookedRides(us.getUsername());
    if (lb != null) {
        for (Booking li : lb) {
            li.setStatus("Rejected");
            li.getRide().setnPlaces(li.getRide().getnPlaces() + li.getSeats());
        }
    }
    List<Alert> la = getAlertsByUsername(us.getUsername());
    if (la != null) {
        for (Alert lx : la) {
            deleteAlert(lx.getAlertNumber());
        }
    }
}

private void deleteDriver(User us) {
    List<Ride> rl = getRidesByDriver(us.getUsername());
    if (rl != null) {
        for (Ride ri : rl) {
            cancelRide(ri);
        }
    }
    Driver d = getDriver(us.getUsername());
    List<Car> cl = d.getCars();
    if (cl != null) {
        for (int i = cl.size() - 1; i >= 0; i--) {
            Car ci = cl.get(i);
            deleteCar(ci);
        }
    }
}
```

5. Egindako errefaktorizazioren deskribapena.

Hasieran aipatutako “Bad Smell” hori, konplexutasun kognitiboa murrizteko eskatzen du, hain zuzen ere, deleteUser() metodoak baldintza asko dituelako. Hori dela eta, kode berrian azaltzen den bezala, Refactor>>Extract Method eginda, hiru errefraktorizazio egin ditut eta kodea hirutan zatitu ondoren, “Bad smell”-a desagertu egin da.

- ***"Duplicate code"***

DataAccess.java klasean: Define a constant instead of duplicating this literal "Accepted" 5 times. [+5 locations]

1. Hasierako kodea.

```
            Duplication
127         book1.setStatus( 1  "Accepted");
128         book2.setStatus("Rejected");
            Duplication
129         book3.setStatus( 2  "Accepted");
            Duplication
130         book4.setStatus( 3  "Accepted");
            Duplication
131         book5.setStatus( 4  "Accepted");

                Duplication
664             if (booking.getStatus().equals( 5  "Accepted") || booking.getStatus().equals("NotDefined")) {
```

2. Errefaktorizatuko kodea.

```
35    private static final String ACCEPTED = "Accepted";

127             book1.setStatus(ACCEPTED);
128             book2.setStatus("Rejected");
129             book3.setStatus(ACCEPTED);
130             book4.setStatus(ACCEPTED);
131             book5.setStatus(ACCEPTED);

public void cancelRide(Ride ride) {
    try {
        db.getTransaction().begin();
        for (Booking booking : ride.getBookings()) {
            if (booking.getStatus().equals(ACCEPTED) || booking.getStatus().equals("NotDefined")) {
```

3. Egindako errefaktorizazioren deskribapena.

"Bad smell" honetan, "Accepted" String-a 5 alditan errepikatzen dela jartzen du. Kodea errepikakorra ez bihurtzeko, *accepted* aldagaia definitzea soluzio hoberena izango da. Horretarako Refactor>>Extract Constant aukeratu dut *ACCEPTED* aldagai berria sortzeko.

- ***"Keep unit interfaces small"***

DataAccess.java klasean: Method has 6 parameters, which is greater than 4 authorized.

1. Hasierako kodea.

```
public boolean erreklamazioaBidali(String nor, String nori, Date gaur, Booking booking, String textua,
        boolean aurk) {
    try {
        db.getTransaction().begin();

        Complaint erreklamazioa = new Complaint(nor, nori, gaur, booking, textua, aurk);
        db.persist(erreklamazioa);
        db.getTransaction().commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}
```

2. Errefaktorizatuko kodea.

```
public boolean erreklamazioaBidali(ErreklamazioaBidaliParameter parameterObject) {
    try {
        db.getTransaction().begin();
        Complaint erreklamazioa = new Complaint(parameterObject.getNor(), parameterObject.getNori(), parameterObject.getGaur()
        parameterObject.getBooking(), parameterObject.getTextua(), parameterObject.isAurk());
        db.persist(erreklamazioa);
        db.getTransaction().commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}
```

Klase berria:

```
package dataAccess;

import java.util.Date;

public class ErreklamazioaBidaliParameter {
    private String nor;
    private String nori;
    private Date gaur;
    private Booking booking;
    private String textua;
    private boolean aurk;

    public ErreklamazioaBidaliParameter(String nor, String nori, Date gaur, Booking booking, String textua,
            boolean aurk) {
        this.nor = nor;
        this.nori = nori;
        this.gaur = gaur;
        this.booking = booking;
        this.textua = textua;
        this.aurk = aurk;
    }

    public String getNor() {
        return nor;
    }

    public void setNor(String nor) {
        this.nor = nor;
    }

    public String getNori() {
        return nori;
    }

    public void setNori(String nori) {
        this.nori = nori;
    }

    public Date getGaur() {
        return gaur;
    }

    public void setGaur(Date gaur) {
        this.gaur = gaur;
    }

    public Booking getBooking() {
        return booking;
    }

    public void setBooking(Booking booking) {
        this.booking = booking;
    }

    public String getTextua() {
        return textua;
    }

    public void setTextua(String textua) {
        this.textua = textua;
    }

    public boolean isAurk() {
        return aurk;
    }

    public void setAurk(boolean aurk) {
        this.aurk = aurk;
    }
}
```

3. Egindako errefaktorizazioren deskribapena.

“Bad smell” honen helburua, *erreklamazioaBidali()* metodoaren parametro kopurua murriztea da, metodoaren egitura garbiago ikusteko. Horretarako, hasierako *erreklamazioaBidali()* metodoaren parametroak aukeratuz, Refactor>>Introduce Parameter Object egin eta gero, aukeraturako parametroekin klase gehigarri bat sortuko da. Klase horretan, parametro horien hasieraketak egingo dira eta *erreklamazioaBidali()-an* 6 parametro ordez, parametro bakarra egongo da.

Egilea: **Luken**

- ***"Write short units of code"***

1. Hasierako kodea.

```
public boolean gauzatuEragiketa(String username, double amount, boolean deposit) {
    try {
        db.getTransaction().begin();
        User user = getUser(username);
        if (user != null) {
            double currentMoney = user.getMoney();
            if (deposit) {
                user.setMoney(currentMoney + amount);
            } else {
                if ((currentMoney - amount) < 0)
                    user.setMoney(0);
                else
                    user.setMoney(currentMoney - amount);
            }
            db.merge(user);
            db.getTransaction().commit();
            return true;
        }
        db.getTransaction().commit();
        return false;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}
```

2. Errefaktorizatuko kodea.

```
public boolean gauzatuEragiketa(String username, double amount, boolean deposit) {
    try {
        db.getTransaction().begin();
        User user = getUser(username);
        if (user != null) {
            return userEzNull(amount, deposit, user);
        }
        db.getTransaction().commit();
        return false;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}
private boolean userEzNull(double amount, boolean deposit, User user) {
    double currentMoney = user.getMoney();
    if (deposit) {
        user.setMoney(currentMoney + amount);
    } else {
        if ((currentMoney - amount) < 0)
            user.setMoney(0);
        else
            user.setMoney(currentMoney - amount);
    }
    db.merge(user);
    db.getTransaction().commit();
    return true;
}
```

3. Egindako errefaktorizazioren deskribapena.

   GauzatuEragiketa() metodoa metodo konplexu eta astuna da ikusi daitekeen bezala. 2. kapituluan azaltzen den bezala, metodoek ez lukete 15 lerro baino gehiagoko aginduak izan behar, eta hortaz, hau ekiditeko, refactor erabiliz, hain zuzen, extract method erabiliz, metodo luze honetatik beste metodo bateri deituz, kodea murriztu dugu aginduak bi metodoen artean banatuz.

- ***"Write simple units of code"***

1. Hasierako kodea.

```
public AlertaSortuGUI(String username) {

    setBussinessLogic(TravelerGUI.getBusinessLogic());

    this.traveler = appFacadeInterface.getTraveler(username);
    this.getContentPane().setLayout(null);
    this.setSize(new Dimension(604, 370));

    this.setTitle(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AddAlert"));

    jLabelOrigin.setBounds(new Rectangle(33, 94, 92, 20));

    jCalendar.setBounds(new Rectangle(300, 50, 225, 150));
    scrollPaneEvents.setBounds(new Rectangle(25, 44, 346, 116));

    jButtonCreate.setBounds(new Rectangle(100, 263, 130, 30));

    jButtonCreate.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            jLabelMsg.setText("");
            String error = field_Errors();
               +2 (incl 1 for nesting)
             1 if (error != null)
                jLabelMsg.setText(error);
               +1
             2 else {
                Date selectedDate = jCalendar.getDate();
                Date currentDate = new Date();
                   +3 (incl 2 for nesting)
                 3 if (selectedDate.before(UtilDate.trim(currentDate))
                          +1
                       4 || UtilDate.trim(selectedDate).equals(UtilDate.trim(currentDate))) {

                    jLabelMsg.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.InvalidDate"));
               +1
                } 5 else {
                    Alert newAlert = new Alert(fieldOrigin.getText(), fieldDestination.getText(),
                            UtilDate.trim(selectedDate), traveler);
```

```
                Alert newAlert = new Alert(fieldOrigin.getText(), fieldDestination.getText(),
                        UtilDate.trim(selectedDate), traveler);
                boolean success = appFacadeInterface.createAlert(newAlert);
                   +4 (incl 3 for nesting)
                 6 if (success) {

                    jLabelMsg.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AlertCreated"));
                    traveler.addAlert(newAlert);
                    JFrame a = new AlertakKudeatuGUI(username);
                    a.setVisible(true);
                    jButtonClose_actionPerformed(e);
                +1
                } 7 else {
                    jLabelMsg.setText(

                            ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AlertCreateFail"));
                }
            }
        }
    }
});
jButtonClose.setBounds(new Rectangle(275, 263, 130, 30));
jButtonClose.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JFrame a = new AlertakKudeatuGUI(username);
        a.setVisible(true);
        jButtonClose_actionPerformed(e);
    }
});

jLabelMsg.setBounds(new Rectangle(275, 214, 305, 20));
jLabelMsg.setForeground(Color.red);

jLabelError.setBounds(new Rectangle(10, 232, 320, 20));
jLabelError.setForeground(Color.red);

this.getContentPane().add(jLabelMsg, null);
this.getContentPane().add(jLabelError, null);

this.getContentPane().add(jButtonClose, null);
this.getContentPane().add(jButtonCreate, null);
this.getContentPane().add(jButtonClose, null);
this.getContentPane().add(jButtonCreate, null);
this.getContentPane().add(jLabelOrigin, null);

this.getContentPane().add(jCalendar, null);

datesWithEventsCurrentMonth = appFacadeInterface.getThisMonthDatesWithRides("a", "b", jCalendar.getDate());

jLabRideDate.setBounds(new Rectangle(40, 15, 140, 25));
jLabRideDate.setBounds(298, 16, 140, 25);
getContentPane().add(jLabRideDate);

jLabelDestination.setBounds(33, 119, 92, 20);
getContentPane().add(jLabelDestination);

fieldOrigin.setBounds(127, 91, 130, 26);
getContentPane().add(fieldOrigin);
fieldOrigin.setColumns(10);

fieldDestination.setBounds(127, 119, 130, 26);
getContentPane().add(fieldDestination);
fieldDestination.setColumns(10);

this.jCalendar.addPropertyChangeListener(new PropertyChangeListener() {
    public void propertyChange(PropertyChangeEvent propertychangeevent) {
           +2 (incl 1 for nesting)
         8 if (propertychangeevent.getPropertyName().equals("locale")) {
            jCalendar.setLocale((Locale) propertychangeevent.getNewValue());
        +1
        } else 9 if (propertychangeevent.getPropertyName().equals("calendar")) {
            calendarAnt = (Calendar) propertychangeevent.getOldValue();
            calendarAct = (Calendar) propertychangeevent.getNewValue();
            @SuppressWarnings("unused")
            DateFormat dateformat1 = DateFormat.getDateInstance(1, jCalendar.getLocale());

            int monthAnt = calendarAnt.get(Calendar.MONTH);
            int monthAct = calendarAct.get(Calendar.MONTH);
                +3 (incl 2 for nesting)
             10 if (monthAct != monthAnt) {
                    +4 (incl 3 for nesting)
                11 if (monthAct == monthAnt + 2) {
```

```
                    // Si en JCalendar está 30 de enero y se avanza al mes siguiente, devolverá 2 de
                    // marzo (se toma como equivalente a 30 de febrero)
                    // Con este código se dejará como 1 de febrero en el JCalendar
                    calendarAct.set(Calendar.MONTH, monthAnt + 1);
                    calendarAct.set(Calendar.DAY_OF_MONTH, 1);
                }

                jCalendar.setCalendar(calendarAct);
            }
            jCalendar.setCalendar(calendarAct);
            int offset = jCalendar.getCalendar().get(Calendar.DAY_OF_WEEK);

            +3 (incl 2 for nesting)
            if (Locale.getDefault().equals(new Locale("es")))
                offset += 4;
            +1
            else
                offset += 5;
            @SuppressWarnings("unused")
            Component o = (Component) jCalendar.getDayChooser().getDayPanel()
                    .getComponent(jCalendar.getCalendar().get(Calendar.DAY_OF_MONTH) + offset);
        }
    }
});

}
```

2. Errefaktorizatuko kodea.

```
public AlertaSortuGUI(String username) {
    setBussinessLogic(TravelerGUI.getBusinessLogic());
    this.traveler = appFacadeInterface.getTraveler(username);

    setupGUI();

    setupListeners(username);

    datesWithEventsCurrentMonth = appFacadeInterface.getThisMonthDatesWithRides("a", "b", jCalendar.getDate());

    this.jCalendar.addPropertyChangeListener(new PropertyChangeListener() {
        public void propertyChange(PropertyChangeEvent propertychangeevent) {
            handleCalendarPropertyChange(propertychangeevent);
        }
    });
}

private void setupGUI() {
    this.getContentPane().setLayout(null);
    this.setSize(new Dimension(604, 370));
    this.setTitle(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AddAlert"));

    jLabelOrigin.setBounds(new Rectangle(33, 94, 92, 20));
    jCalendar.setBounds(new Rectangle(300, 50, 225, 150));
    scrollPaneEvents.setBounds(new Rectangle(25, 44, 346, 116));

    jButtonCreate.setBounds(new Rectangle(100, 263, 130, 30));
    jButtonClose.setBounds(new Rectangle(275, 263, 130, 30));

    jLabelMsg.setBounds(new Rectangle(275, 214, 305, 20));
    jLabelMsg.setForeground(Color.red);
    jLabelError.setBounds(new Rectangle(10, 232, 320, 20));
    jLabelError.setForeground(Color.red);

    this.getContentPane().add(jLabelMsg);
    this.getContentPane().add(jLabelError);
    this.getContentPane().add(jButtonClose);
    this.getContentPane().add(jButtonCreate);
    this.getContentPane().add(jLabelOrigin);
```

```
        this.getContentPane().add(jCalendar);
    }

    private void setupListeners(String username) {
        jButtonCreate.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                createAlert(username, e);
            }
        });

        jButtonClose.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JFrame a = new AlertakKudeatuGUI(username);
                a.setVisible(true);
                jButtonClose_actionPerformed(e);
            }
        });
    }

    private void createAlert(String username, ActionEvent e) {
        jLabelMsg.setText("");
        String error = field_Errors();
        if (error != null) {
            jLabelMsg.setText(error);
        } else {
            Date selectedDate = jCalendar.getDate();
            Date currentDate = new Date();
            if (selectedDate.before(UtilDate.trim(currentDate)) ||
                UtilDate.trim(selectedDate).equals(UtilDate.trim(currentDate))) {
                jLabelMsg.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.InvalidDate"));
            } else {
                Alert newAlert = new Alert(fieldOrigin.getText(), fieldDestination.getText(),
                                           UtilDate.trim(selectedDate), traveler);
                boolean success = appFacadeInterface.createAlert(newAlert);
                handleAlertCreation(success, username, e, newAlert);
            }
        }
    }
```

```
private void handleAlertCreation(boolean success, String username, ActionEvent e, Alert newAlert) {
    if (success) {
        jLabelMsg.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AlertCreated"));
        traveler.addAlert(newAlert);
        JFrame a = new AlertakKudeatuGUI(username);
        a.setVisible(true);
        jButtonClose_actionPerformed(e);
    } else {
        jLabelMsg.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AlertCreateFail"));
    }
}

private void handleCalendarPropertyChange(PropertyChangeEvent propertychangeevent) {
    if (propertychangeevent.getPropertyName().equals("locale")) {
        jCalendar.setLocale((Locale) propertychangeevent.getNewValue());
    } else if (propertychangeevent.getPropertyName().equals("calendar")) {
        calendarAnt = (Calendar) propertychangeevent.getOldValue();
        calendarAct = (Calendar) propertychangeevent.getNewValue();
        @SuppressWarnings("unused")
        DateFormat dateformat1 = DateFormat.getDateInstance(1, jCalendar.getLocale());

        int monthAnt = calendarAnt.get(Calendar.MONTH);
        int monthAct = calendarAct.get(Calendar.MONTH);
        if (monthAct != monthAnt) {
            if (monthAct == monthAnt + 2) {
                calendarAct.set(Calendar.MONTH, monthAnt + 1);
                calendarAct.set(Calendar.DAY_OF_MONTH, 1);
            }
            jCalendar.setCalendar(calendarAct);
        }
        jCalendar.setCalendar(calendarAct);
        int offset = jCalendar.getCalendar().get(Calendar.DAY_OF_WEEK);

        if (Locale.getDefault().equals(new Locale("es")))
            offset += 4;
        else
            offset += 5;
        @SuppressWarnings("unused")
        Component o = (Component) jCalendar.getDayChooser().getDayPanel()
                .getComponent(jCalendar.getCalendar().get(Calendar.DAY_OF_MONTH) + offset);
    }
}
```

3. Egindako errefaktorizazioren deskribapena.

   Kode zati handi honetan ikusi daitekeen bezala, erramifikazio asko zeuden if() eta else() beterik zegoelako eta ‘bad smell’ sortzen zuelako. Hau konpondu nahian, refactor-en bidez, Extract method eginez, metodo handi bat zena, submetodoetan zatitzen joan gara garbiago eta egokiago bihurtuz.

- ***"Duplicate code"***

1. Hasierako kodea.

```
Date gaur = new Date();

this.setSize(495, 290);

Duplication
jLabelSelectOption = new JLabel(ResourceBundle.getBundle( 1 "Etiquetas").getString("ArazoaGUI.Arazoa"));
jLabelSelectOption.setBounds(180, 11, 240, 36);
jLabelSelectOption.setFont(new Font("Tahoma", Font.BOLD, 13));
jLabelSelectOption.setForeground(Color.BLACK);
jLabelSelectOption.setHorizontalAlignment(SwingConstants.CENTER);

jButtonJun = new JButton();
jButtonJun.setBounds(40, 70, 240, 50);
Duplication
jButtonJun.setText(ResourceBundle.getBundle( 2 "Etiquetas").getString("ArazoaGUI.Eznaizaurkeztu"));
jButtonJun.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        appFacadeInterface.erreklamazioaBidali(nori, nork, gaur, booking, "Ez da agertu",false);
        jButtonClose_actionPerformed(e);
    }
});

jButtonErrekla = new JButton();
jButtonErrekla.setBounds(40, 70, 240, 50);
Duplication
jButtonErrekla.setText(ResourceBundle.getBundle( 3 "Etiquetas").getString("BezeroGUI.Erreklamatu"));
jButtonErrekla.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        JFrame a = new ErreklamazioakGUI(nork, nori, booking);
        a.setVisible(true);
        jButtonClose_actionPerformed(e);
    }
```

2. Errefaktorizatuko kodea.

```
this.setSize(495, 290);

String etiketa = "Etiquetas";
Duplication
jLabelSelectOption = new JLabel(ResourceBundle.getBundle(etiketa).getString("ArazoaGUI.Arazoa"));
jLabelSelectOption.setBounds(180, 11, 240, 36);
jLabelSelectOption.setFont(new Font("Tahoma", Font.BOLD, 13));
jLabelSelectOption.setForeground(Color.BLACK);
jLabelSelectOption.setHorizontalAlignment(SwingConstants.CENTER);

jButtonJun = new JButton();
jButtonJun.setBounds(40, 70, 240, 50);
Duplication
jButtonJun.setText(ResourceBundle.getBundle(etiketa).getString("ArazoaGUI.Eznaizaurkeztu"));
jButtonJun.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        appFacadeInterface.erreklamazioaBidali(nori, nork, gaur, booking, "Ez da agertu",false);
        jButtonClose_actionPerformed(e);
    }
});

jButtonErrekla = new JButton();
jButtonErrekla.setBounds(40, 70, 240, 50);
Duplication
jButtonErrekla.setText(ResourceBundle.getBundle(etiketa).getString("BezeroGUI.Erreklamatu"));
jButtonErrekla.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        JFrame a = new ErreklamazioakGUI(nork, nori, booking);
        a.setVisible(true);
        jButtonClose_actionPerformed(e);
    }
```

3. Egindako errefaktorizazioren deskribapena.

   Kasu honetan, errepikatutako kodea ekiditeko, sonarlink erabiliz, ikusi dut programa bloke honetan 'Etiquetas' string-a hiru aldiz erabiltzen dela. Hortaz, refactor erabiliz, extract local variable, string-a aldagai lokal bezela definitu dut, programa hobetuz.

- ***"Keep unit interfaces small"***

1. Hasierako kodea.

```
 */
public Ride createRide(String from, String to, Date date, int nPlaces, float price, String driverName)
        throws RideAlreadyExistException, RideMustBeLaterThanTodayException {
    System.out.println(
            ">> DataAccess: createRide=> from= " + from + " to= " + to + " driver=" + driverName + " date " + date);
    if (driverName==null) return null;
    try {
        if (new Date().compareTo(date) > 0) {
            System.out.println("ppppp");
            throw new RideMustBeLaterThanTodayException(
                    ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.ErrorRideMustBeLaterThanToday"));
        }

        db.getTransaction().begin();
        Driver driver = db.find(Driver.class, driverName);
        if (driver.doesRideExists(from, to, date)) {
            db.getTransaction().commit();
            throw new RideAlreadyExistException(
                    ResourceBundle.getBundle("Etiquetas").getString("DataAccess.RideAlreadyExist"));
        }
        Ride ride = driver.addRide(from, to, date, nPlaces, price);
        // next instruction can be obviated
        db.persist(driver);
        db.getTransaction().commit();

        return ride;
    } catch (NullPointerException e) {
        // TODO Auto-generated catch block
        return null;
    }
```

2. Errefaktorizatuko kodea.

```
public Ride createRide(CreateRideParameter parameterObject)
        throws RideAlreadyExistException, RideMustBeLaterThanTodayException {
    System.out.println(
            ">> DataAccess: createRide=> from= " + parameterObject.from + " to= " + parameterObject.to + " driver=" + parameterObject.driver
    if (parameterObject.driverName==null) return null;
    try {
        if (new Date().compareTo(parameterObject.date) > 0) {
            System.out.println("ppppp");
            throw new RideMustBeLaterThanTodayException(
                    ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.ErrorRideMustBeLaterThanToday"));
        }

        db.getTransaction().begin();
        Driver driver = db.find(Driver.class, parameterObject.driverName);
        if (driver.doesRideExists(parameterObject.from, parameterObject.to, parameterObject.date)) {
            db.getTransaction().commit();
            throw new RideAlreadyExistException(
                    ResourceBundle.getBundle("Etiquetas").getString("DataAccess.RideAlreadyExist"));
        }
        Ride ride = driver.addRide(parameterObject.from, parameterObject.to, parameterObject.date, parameterObject.nPlaces, parameterObject.
        // next instruction can be obviated
        db.persist(driver);
        db.getTransaction().commit();

        return ride;
    } catch (NullPointerException e) {
        // TODO Auto-generated catch block
        return null;
    }


}
```

```java
package dataAccess;

import java.util.Date;

public class CreateRideParameter {
    public String from;
    public String to;
    public Date date;
    public int nPlaces;
    public float price;
    public String driverName;

    public CreateRideParameter(String from, String to, Date date, int nPlaces, float price, String driverName) {
        this.from = from;
        this.to = to;
        this.date = date;
        this.nPlaces = nPlaces;
        this.price = price;
        this.driverName = driverName;
    }
}
```

3. Egindako errefaktorizazioren deskribapena.

CreateRide() metodoan seis parametro sartzen dira hasieratzen denean, ‘code smell’ sortuz kantitate handiegia delako. Refactor erabiliz kasu honetan, lortzen duguna da sei parametro hauek bakarra bihurtzea. Hau egin dezakegu ‘Introduce Parameter Object’ kasuarekin, seis parametroak objetu berri batean sortuz eta sarrera sinplifikatuz.

Egilea: **Unax**

- ***"Write short units of code"***

1. Hasierako kodea.

```
public Ride createRide(CreateRideParameter parameterObject)
        throws RideAlreadyExistException, RideMustBeLaterThanTodayException {
    System.out.println(
            ">> DataAccess: createRide=> from= " + parameterObject.from + " to= " + parameterObject.to + " driver=" + parameterObject.driverName + "
    if (parameterObject.driverName==null) return null;
    try {
        if (new Date().compareTo(parameterObject.date) > 0) {
            System.out.println("ppppp");
            throw new RideMustBeLaterThanTodayException(
                    ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.ErrorRideMustBeLaterThanToday"));
        }

        db.getTransaction().begin();
        Driver driver = db.find(Driver.class, parameterObject.driverName);
        if (driver.doesRideExists(parameterObject.from, parameterObject.to, parameterObject.date)) {
            db.getTransaction().commit();
            throw new RideAlreadyExistException(
                    ResourceBundle.getBundle("Etiquetas").getString("DataAccess.RideAlreadyExist"));
        }
        Ride ride = driver.addRide(parameterObject.from, parameterObject.to, parameterObject.date, parameterObject.nPlaces, parameterObject.price);
        // next instruction can be obviated
        db.persist(driver);
        db.getTransaction().commit();

        return ride;
    } catch (NullPointerException e) {
        // TODO Auto-generated catch block
        return null;
    }


}
```

2. Errefaktorizatuko kodea.

```
public Ride createRide(CreateRideParameter parameterObject)throws RideAlreadyExistException, RideMustBeLaterThanTodayException {
    System.out.println(">> DataAccess: createRide=> from= " + parameterObject.from + " to= " + parameterObject.to + " driver=" + parameterObject.driverN
    if (parameterObject.driverName == null) {
        return null;
    }
    try {
        validateRideDate(parameterObject.date);
        db.getTransaction().begin();
        Driver driver = findDriver(parameterObject.driverName);
        validateRideExistence(driver, parameterObject);
        Ride ride = createNewRide(driver, parameterObject);
        db.getTransaction().commit();
        return ride;
    } catch (NullPointerException e) {
        return null;
    }
}

private void validateRideDate(Date date) throws RideMustBeLaterThanTodayException {
    if (new Date().compareTo(date) > 0) {
        System.out.println("ppppp");
        throw new RideMustBeLaterThanTodayException(
                ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.ErrorRideMustBeLaterThanToday"));
    }
}

private Driver findDriver(String driverName) {
    return db.find(Driver.class, driverName);
}

private void validateRideExistence(Driver driver, CreateRideParameter parameterObject)
        throws RideAlreadyExistException {
    if (driver.doesRideExists(parameterObject.from, parameterObject.to, parameterObject.date)) {
        db.getTransaction().commit();
        throw new RideAlreadyExistException(
                ResourceBundle.getBundle("Etiquetas").getString("DataAccess.RideAlreadyExist"));
    }
}

private Ride createNewRide(Driver driver, CreateRideParameter parameterObject) {
    Ride ride = driver.addRide(parameterObject.from, parameterObject.to, parameterObject.date,
            parameterObject.nPlaces, parameterObject.price);
    db.persist(driver);
    return ride;
}
```

3. Egindako errefaktorizazioren deskribapena.

“Bad smell” esaten duenez bezala, kodea hain luzea ez izateko metodoa bitan banatu daiteke. Refactor>>Extract Method eginda, beste metodo bat sortzen da, bertan kodearen zati bat definitzeko eta metodo nagusiaren luzera murrizteko.

- ***"Write simple units of code"***

1. Hasierako kodea.

```
public boolean updateAlertaAurkituak(String username) {
	try {
		db.getTransaction().begin();

		boolean alertFound = false;
		TypedQuery<Alert> alertQuery = db.createQuery("SELECT a FROM Alert a WHERE a.traveler.username = :username",
				Alert.class);
		alertQuery.setParameter("username", username);
		List<Alert> alerts = alertQuery.getResultList();

		TypedQuery<Ride> rideQuery = db
				.createQuery("SELECT r FROM Ride r WHERE r.date > CURRENT_DATE AND r.active = true", Ride.class);
		List<Ride> rides = rideQuery.getResultList();

		for (Alert alert : alerts) {
			boolean found = false;
			for (Ride ride : rides) {
				if (UtilDate.datesAreEqualIgnoringTime(ride.getDate(), alert.getDate())
						&& ride.getFrom().equals(alert.getFrom()) && ride.getTo().equals(alert.getTo())
						&& ride.getnPlaces() > 0) {
					alert.setFound(true);
					found = true;
					if (alert.isActive())
						alertFound = true;
					break;
				}
			}
			if (!found) {
				alert.setFound(false);
			}
			db.merge(alert);
		}

		db.getTransaction().commit();
		return alertFound;
	} catch (Exception e) {
		e.printStackTrace();
		db.getTransaction().rollback();
		return false;
	}
}
```

2. Errefaktorizatuko kodea.

```
public boolean updateAlertaAurkituak(String username) {
    try {
        db.getTransaction().begin();

        boolean alertFound = false;

        List<Alert> alerts = getAlertsByUsername(username);
        List<Ride> rides = getActiveRides();

        for (Alert alert : alerts) {
            boolean found = checkIfAlertMatchesRides(alert, rides);
            updateAlert(alert);

            if (alert.isActive() && found) {
                alertFound = true;
            }
        }

        db.getTransaction().commit();
        return alertFound;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}

public List<Ride> getActiveRides() {
    TypedQuery<Ride> query = db.createQuery("SELECT r FROM Ride r WHERE r.date > CURRENT_DATE AND r.active = true", Ride.class);
    return query.getResultList();
}

private boolean checkIfAlertMatchesRides(Alert alert, List<Ride> rides) {
    for (Ride ride : rides) {
        if (UtilDate.datesAreEqualIgnoringTime(ride.getDate(), alert.getDate())
                && ride.getFrom().equals(alert.getFrom())
                && ride.getTo().equals(alert.getTo())
                && ride.getnPlaces() > 0) {
            alert.setFound(true);
            return true;
        }
    }
    return false;
}
```

3. Egindako errefaktorizazioren deskribapena.

Hasieran aipatutako “Bad Smell” hori, konplexutasun kognitiboa murrizteko eskatzen du, hain zuzen ere, updateAlertaAurkituak() metodoak baldintza asko dituelako. Hori dela eta, kode berrian azaltzen den bezala, Refactor>>Extract Method eginda, hiru errefraktorizazio egin ditut eta kodea hirutan zatitu ondoren, “Bad smell”-a desagertu egin da.

- ***"Duplicate code"***

1. Hasierako kodea.

```
Movement m1 = new Movement(traveler1, "BookFreeze", 20);
Movement m2 = new Movement(traveler1, "BookFreeze", 40);
Movement m3 = new Movement(traveler1, "BookFreeze", 5);
Movement m4 = new Movement(traveler2, "BookFreeze", 4);
Movement m5 = new Movement(traveler1, "BookFreeze", 3);
Movement m6 = new Movement(driver1, "Deposit", 15);
Movement m7 = new Movement(traveler1, "Deposit", 168);
```

2. Errefaktorizatuko kodea.

```
Movement m1 = new Movement(traveler1, BF, 20);

Movement m2 = new Movement(traveler1, BF, 40);

Movement m3 = new Movement(traveler1, BF, 5);

Movement m4 = new Movement(traveler2, BF, 4);

Movement m5 = new Movement(traveler1, BF, 3);
```

3. Egindako errefaktorizazioren deskribapena.

"Bad smell" honetan, "`BookFreeze`" String-a 5 alditan errepikatzen dela jartzen du. Kodea errepikakorra ez bihurtzeko, *BF* aldagaia definitzea soluzio hoberena izango da. Horretarako Refactor>>Extract Constant aukeratu dut *BF* aldagai berria sortzeko.

- ***"Keep unit interfaces small"***

1. Hasierako kodea.

```
@WebMethod
public Ride createRide(String from, String to, Date date, int nPlaces, float price, String driverName)
        throws RideMustBeLaterThanTodayException, RideAlreadyExistException;
```

2. Errefaktorizatuko kodea.

```
@WebMethod
public Ride createRide(CreateRideParameter parameterObject)
        throws RideMustBeLaterThanTodayException, RideAlreadyExistException;
```

3. Egindako errefaktorizazioren deskribapena.

"Bad smell" honen helburua, *erreklamazioaBidali()* metodoaren definizioan parametro kopurua murriztea da, metodoaren egitura garbiago ikusteko. Horretarako, hasierako *erreklamazioaBidali()* metodoaren parametroak aukeratuz, Refactor>>Introduce Parameter Object egin eta gero, aukeraturako parametroekin klase gehigarri bat sortu beharko litzake, baina dagoeneko sortua dago. Klase horretan, parametro horien hasieraketak egingo dira eta *erreklamazioaBidali()-an* 6 parametro ordez, parametro bakarra egongo da.