LABORATORIO REFACTORIZACIÓN

Paula Recaj, Irati Cruz y Amna Rumaisa

Refactorizaciones Paula Recaj

- Código inicial (**"Write short units of code"**):

```
	public void deleteUser(User us) {
		try {
			if (us.getMota().equals("Driver")) {
				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);
					}
				}
			} else {
				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());
					}
				}
			}
			db.getTransaction().begin();
			us = db.merge(us);
			db.remove(us);
			db.getTransaction().commit();
		} catch (Exception e) {
			e.printStackTrace();
```

```
            }
        }
```

- Código refactorizado:

```
        //Método 1 para reducir líneas del método deleteUser
        public void cancelRidesByUser(User us) {
                List<Ride> rl = getRidesByDriver(us.getUsername());
                if (rl != null) {
                        for (Ride ri : rl) {
                                cancelRide(ri);
                        }
                }
        }

        //Método 2 para reducir líneas del método deleteUser
        public void deleteCarsByUser (User us) {
                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);
                        }
                }
        }

        //Método 3 para reducir líneas del método deleteUser
        public void rejectBookedRidesByUser(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());
                        }
                }
        }
        //Método 4 para reducir líneas del método deleteUser
        public void deleteAlertsByUser(User us) {
                List<Alert> la = getAlertsByUsername(us.getUsername());
                if (la != null) {
                        for (Alert lx : la) {
                                deleteAlert(lx.getAlertNumber());
                        }
                }
        }
```

```
	public void deleteUser(User us) {
		try {
			if (us.getMota().equals("Driver")) {
				cancelRidesByUser(us);
				deleteCarsByUser(us);
			} else {
				rejectBookedRidesByUser(us);
				deleteAlertsByUser(us);
			}
			db.getTransaction().begin();
			us = db.merge(us);
			db.remove(us);
			db.getTransaction().commit();

		} catch (Exception e) {

			e.printStackTrace();
		}
	}
```

- Descripción del error concreto detectado y descripción de la refactorización realizada.
  - El error detectado se trata de un método que supera las 15 líneas de código, por lo que se trata de un bad smell de "Write short units of code".
  - Para refactorizar este método, he creado 4 métodos extra para reducir el número de líneas de código del método principal. Esto seleccionando el código que quiero refactorizar, click derecho y darle a refactor → extract method.

- Código inicial (**"Write simple units of code"**):

```
	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(USER, 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;
            }
    }
```

- Código refactorizado:

```
    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(USER, 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();
                    alertFound = extracted(alertFound, alerts, rides);
                    db.getTransaction().commit();
                    return alertFound;
            } catch (Exception e) {
                    e.printStackTrace();
                    db.getTransaction().rollback();
                    return false;
            }
```

```
        }

        //Método para mejorar el updateAlertaAurkituak
        private boolean extracted(boolean alertFound, List<Alert> alerts, List<Ride> rides) {
                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);
                }
                return alertFound;
        }
```

- Descripción del error concreto detectado y descripción de la refactorización realizada.
    - El error sería tener tantos for-s e if-s en el mismo método, por lo que estaríamos hablando de un bad smell de "Write simple units of code".
    - Para refactorizar el código he creado un método seleccionando el código que quiero refactorizar, haciendo click derecho y dándole a refactor → extract method.

- Código inicial (**“Duplicate code”**):

```
                        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);
```

- Código refactorizado:

```
private static final String BOOK_FREEZE = "BookFreeze";
                Movement m1 = new Movement(traveler1, BOOK_FREEZE, 20);
                Movement m2 = new Movement(traveler1, BOOK_FREEZE, 40);
                Movement m3 = new Movement(traveler1, BOOK_FREEZE, 5);
                Movement m4 = new Movement(traveler2, BOOK_FREEZE, 4);
                Movement m5 = new Movement(traveler1, BOOK_FREEZE, 3);
```

- Descripción del error concreto detectado y descripción de la refactorización realizada.
  - El error es que se duplica 5 veces el string "BookFreeze", por lo que es un bad smell de "Duplicate code".
  - Para refactorizar el error, hacemos click derecho en "BookFreeze", le damos a refactor → extract constant. Esto lo que hace es crear una variable constante con el valor que se repetía.

- Código inicial (**"Keep unit interfaces small"**):

```
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;
        }
}
```

- Código refactorizado:

```
public boolean erreklamazioaBidali(Complaint complaint) {
        try {
                db.getTransaction().begin();
                db.persist(complaint);
                db.getTransaction().commit();
                return true;
        } catch (Exception e) {
                e.printStackTrace();
                db.getTransaction().rollback();
                return false;
        }
}
```

- Descripción del error concreto detectado y descripción de la refactorización realizada.
    - El error estaba en que había 6 parámetros en el método, por lo que podría ser un bad smell (aunque Sonar Cloud no avisaba del error). Sería del tipo "Keep unit interfaces small".
    - Para refactorizar el error he seleccionado todos los parámetros, click derecho en refactor → change method signature. Ahí he eliminado los 6 parámetros y los he sustituido por la clase que componían. También he refactorizado este método en las clases BLFacade, BLFacadeImplementation, ArazoaGUI, ErreklamazioakGUI, pues hacían referencia a la versión sin refactorizar. En las BLF* ha bastado con cambiar los parámetros del método, en las demás, he creado un Complaint con los elementos que se enviaban al método originalmente. Ejemplo:

```
appFacadeInterface.erreklamazioaBidali(nori, nork, gaur, booking, "Ez da agertu",false);
```

```
Complaint complaint = new Complaint (nori, nork, gaur, booking, "Ez da agertu",false);
appFacadeInterface.erreklamazioaBidali(complaint);
```

Refactorizaciones Irati Cruz

- **Bad Smell: Write short units of code**
  - Código inicial

```
public boolean bookRide(String username, Ride ride, int seats, double desk) {
    try {
        db.getTransaction().begin();
        Traveler traveler = getTraveler(username);
        if (traveler == null) {
            return false;
        }

        if (ride.getnPlaces() < seats) {
            return false;
        }

        double ridePriceDesk = (ride.getPrice() - desk) * seats;
        double availableBalance = traveler.getMoney();
        if (availableBalance < ridePriceDesk) {
            return false;
        }
        Booking booking = new Booking(ride, traveler, seats);
        booking.setTraveler(traveler);
        booking.setDeskontua(desk);
        db.persist(booking);
        ride.setnPlaces(ride.getnPlaces() - seats);
        traveler.addBookedRide(booking);
        traveler.setMoney(availableBalance - ridePriceDesk);
        traveler.setIzoztatutakoDirua(traveler.getIzoztatutakoDirua() + ridePriceDesk);
        db.merge(ride);
        db.merge(traveler);
        db.getTransaction().commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}
```

  - Código refactorizado

```
public boolean bookRide(String username, Ride ride, int seats, double desk) {
    try {
        db.getTransaction().begin();

        Traveler traveler = getTraveler(username);

        if(traveler == null ||!hasEnoughPlaces(ride, seats) || !hasEnoughBalance(traveler, ride, seats, desk)) {
            return false;
        }
        Booking booking = createBooking(ride, traveler, seats, desk);
        db.persist(booking);

        double ridePriceDesk = (ride.getPrice() - desk) * seats;
        ride.setnPlaces(ride.getnPlaces() - seats);

        updateRideAndTraveler(ride, traveler, ridePriceDesk, booking);
        db.getTransaction().commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}
```

```java
//Metodo 1: hasEnoughPlaces()
private boolean hasEnoughPlaces(Ride ride, int seats) {
    return ride.getnPlaces() >= seats;
}

//Metodo 2: hasEnoughBalance()
private boolean hasEnoughBalance(Traveler traveler, Ride ride, int seats, double desk) {
    double ridePriceDesk = (ride.getPrice() - desk) * seats;
    return traveler.getMoney() >= ridePriceDesk;
}

//Metodo 3: createBooking()
private Booking createBooking(Ride ride, Traveler traveler, int seats, double desk) {
    Booking booking = new Booking(ride, traveler, seats);
    booking.setTraveler(traveler);
    booking.setDeskontua(desk);
    return booking;
}
//Metodo 4: updateRideAndTraveler()
private void updateRideAndTraveler(Ride ride, Traveler traveler, double ridePriceDesk, Booking booking) {
    /*
     * double ridePriceDesk = (ride.getPrice() - desk) * seats;
     * ride.setnPlaces(ride.getnPlaces() - seats);
     */
    traveler.addBookedRide(booking);
    traveler.setMoney(traveler.getMoney() - ridePriceDesk);
    traveler.setIzoztatutakoDirua(traveler.getIzoztatutakoDirua() + ridePriceDesk);

    db.merge(ride);
    db.merge(traveler);
}
```

- Descripción del error concreto detectado y descripción de la refactorización realizada. Antes de la refactorización, la función *bookRide* tenía 31 líneas de código, lo que estaba por encima de las 15 líneas recomendadas. Para arreglarlo, refactoricé el código y creé 4 métodos nuevos. Así, pude dividir la lógica en partes más pequeñas y fáciles de entender.

- **Bad Smell: Write simple units of code**
  - Código inicial

```java
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;
    }
}
```

- Código refactorizado

```
public boolean gauzatuEragiketa(String username, double amount, boolean deposit) {
    try {
        db.getTransaction().begin();
        User user = getUser(username);
        if (user == null) {
            db.getTransaction().commit();
            return false;
        }
        updateUserMoney(user, amount, deposit);
        db.merge(user);
        db.getTransaction().commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}

//IRATI (2) BAD SMELL: Write simple units of code

//Metodo auxiliar: updateUserMoney()
private void updateUserMoney(User user, double amount, boolean deposit) {
    double currentMoney = user.getMoney();
    if (deposit) {
        user.setMoney(currentMoney + amount);
    } else {
        user.setMoney(Math.max(currentMoney - amount, 0));
    }
}
```

- Descripción del error concreto detectado y descripción de la refactorización realizada.
  El método ***gauzatuEragiketa*** tenía más de 4 puntos de bifurcación, lo que complicaba su mantenimiento y aumentaba la probabilidad de errores. Para solucionarlo, he creado un nuevo método llamado ***updateUserMoney*** que gestiona el saldo del usuario y he eliminado el anidamiento innecesario.

- **Bad Smell: Duplicate code**
  - Código inicial

```
book1.setStatus("Accepted");
book2.setStatus("Rejected");
book3.setStatus("Accepted");
book4.setStatus("Accepted");
book5.setStatus("Accepted");
```

  - Código refactorizado

```
public class DataAccess implements Serializable {
    private static final String BOOK_FREEZE = "BookFreeze";
    private static final String ACCEPTED = "Accepted";
```

```
book1.setStatus(ACCEPTED);
book2.setStatus("Rejected");
book3.setStatus(ACCEPTED);
book4.setStatus(ACCEPTED);
book5.setStatus(ACCEPTED);
```

  - Descripción del error concreto detectado y descripción de la refactorización realizada.
    El error concreto es la repetición del valor "Accepted" en varias líneas, lo que puede

generar inconsistencias si se necesita cambiar este valor en el futuro. La refactorización implicaría almacenar "Accepted" en una constante y luego usar esa constante en lugar del valor literal. Esto facilita el mantenimiento del código, ya que si el valor cambia, solo se necesita modificar en un lugar.

- **Bad Smell: Keep unit interfaces small**
  - Código inicial
  - Código refactorizado
  - Descripción del error concreto detectado y descripción de la refactorización realizada.

He estado trabajando durante dos horas en el método **createRide()**, que al principio me pareció sencillo. Comencé creando dos nuevas clases (*Location y RideAvailability*) y haciendo cambios en tres clases del paquete business_logic, añadiendo como atributos los parámetros ***from*** y ***to*** en *Location* y ***nPlaces*** y ***price*** en *RideAvailability*. Sin embargo, a medida que refactorizaba esas clases, comenzaron a surgir errores en otras como *Ride* y *Drive*, así como en las clases de prueba. Esto me llevó a tener que refactorizar también en todas las clases de testeo y en muchas más. Al final, al comprobar los tests, empecé a enfrentar numerosos errores. Después de tantos cambios y complicaciones, decidí abrir el proyecto en otro workspace para evitar comprometer tantos cambios desastrosos.

Refactorizaciones Amna Rumaisa

- **Bad Smell: Write short units of code (class AdminGUI)**

- Código inicial

```
ublic AdminGUI(String username) {
                    AdminGUI.setBussinessLogic(LoginGUI.getBusinessLogic());
                    this.setTitle(ResourceBundle.getBundle("Etiquetas").getString("AdminGUI.Admin"));
                    this.setSize(495, 290);
                    jLabelSelectOption                                  =                                  new
JLabel(ResourceBundle.getBundle("Etiquetas").getString("AdminGUI.Admin"));
                    jLabelSelectOption.setFont(new Font("Tahoma", Font.BOLD, 13));
                    jLabelSelectOption.setForeground(Color.BLACK);
                    jLabelSelectOption.setHorizontalAlignment(SwingConstants.CENTER);
                    jButtonDeskontu = new JButton();
                    jButtonDeskontu.addActionListener(new ActionListener() {
                            public void actionPerformed(ActionEvent e) {
                                    JFrame a = new DeskontuaGUI(username);
                                    a.setVisible(true);
                            }
                    });

jButtonDeskontu.setText(ResourceBundle.getBundle("Etiquetas").getString("AdminGUI.Deskontua"));
                    jButtonkude = new JButton();
                    jButtonkude.addActionListener(new ActionListener() {
                            public void actionPerformed(ActionEvent e) {
                                    JFrame a = new DeskontuKudeatuGUI(username);
                                    a.setVisible(true);
                            }
```

```
                });

jButtonkude.setText(ResourceBundle.getBundle("Etiquetas").getString("AdminGUI.Kudea"));
                jButtonEzabatu = new JButton();

jButtonEzabatu.setText(ResourceBundle.getBundle("Etiquetas").getString("AdminGUI.Ezab"));
                jButtonEzabatu.addActionListener(new java.awt.event.ActionListener() {
                        public void actionPerformed(java.awt.event.ActionEvent e) {
                                JFrame a = new EzabatuGUI();
                                a.setVisible(true);
                        }
                });
                jButtonItxi = new JButton();
                jButtonItxi.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                jButtonClose_actionPerformed(e);
                        }
                });
                jButtonItxi.setText(ResourceBundle.getBundle("Etiquetas").getString("EgoeraGUI.Close"));
                jContentPane = new JPanel();
                jContentPane.setLayout(new GridLayout(6, 1, 0, 0));
                jContentPane.add(jLabelSelectOption);
                jContentPane.add(jButtonDeskontu);
                jContentPane.add(jButtonkude);
                jContentPane.add(jButtonEzabatu);
                jContentPane.add(jButtonItxi);
                setContentPane(jContentPane);
        }
```

- Código refactorizado

```
public AdminGUI(String username) {
                AdminGUI.setBussinessLogic(LoginGUI.getBusinessLogic());
                this.setTitle(ResourceBundle.getBundle("Etiquetas").getString("AdminGUI.Admin"));
                this.setSize(495, 290);
                setjLabelSelectOption();
                JButton deskontu =setjButtonDeskontu(username);
                JButton kude= setJButtonkude(username);
                JButton ezabatu = setjButtonEzabatu();
                JButton itxi = setjButtonItxi();
                setjContentPane(itxi, kude, ezabatu, deskontu);
                setContentPane(jContentPane);
        }
        private void setjContentPane(JButton jButtonItxi, JButton jButtonkude, JButton jButtonEzabatu,
JButton jButtonDeskontu) {
                jContentPane = new JPanel();
                jContentPane.setLayout(new GridLayout(6, 1, 0, 0));
                jContentPane.add(jLabelSelectOption);
                jContentPane.add(jButtonDeskontu);
                jContentPane.add(jButtonkude);
                jContentPane.add(jButtonEzabatu);
```

```
                jContentPane.add(jButtonItxi);
        }
        private JButton setjButtonItxi() {
                JButton jButtonItxi = new JButton();
                jButtonItxi.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                jButtonClose_actionPerformed(e);
                        }
                });
                jButtonItxi.setText(ResourceBundle.getBundle("Etiquetas").getString("EgoeraGUI.Close"));

                return jButtonItxi;
        }
        private JButton setjButtonEzabatu() {
                JButton jButtonEzabatu = new JButton();

jButtonEzabatu.setText(ResourceBundle.getBundle("Etiquetas").getString("AdminGUI.Ezab"));
                jButtonEzabatu.addActionListener(new java.awt.event.ActionListener() {
                        public void actionPerformed(java.awt.event.ActionEvent e) {
                                JFrame a = new EzabatuGUI();
                                a.setVisible(true);
                        }
                });

                return jButtonEzabatu;
        }
        private JButton setJButtonkude(String username) {
                JButton jButtonkude = new JButton();
                jButtonkude.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                JFrame a = new DeskontuKudeatuGUI(username);
                                a.setVisible(true);
                        }
                });

jButtonkude.setText(ResourceBundle.getBundle("Etiquetas").getString("AdminGUI.Kudea"));

                return jButtonkude;
        }
        private JButton setjButtonDeskontu(String username) {
                JButton jButtonDeskontu = new JButton();
                jButtonDeskontu.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                JFrame a = new DeskontuaGUI(username);
                                a.setVisible(true);
                        }
                });

jButtonDeskontu.setText(ResourceBundle.getBundle("Etiquetas").getString("AdminGUI.Deskontua"));
                return jButtonDeskontu;
        }
        private void setjLabelSelectOption() {
```

```
                jLabelSelectOption = new
JLabel(ResourceBundle.getBundle("Etiquetas").getString("AdminGUI.Admin"));
                jLabelSelectOption.setFont(new Font("Tahoma", Font.BOLD, 13));
                jLabelSelectOption.setForeground(Color.BLACK);
                jLabelSelectOption.setHorizontalAlignment(SwingConstants.CENTER);
        }
```

- Descripción del error concreto detectado y descripción de la refactorización realizada.

  En este constructor habían más de 15 líneas lo que no cumplía el bad smell the "Write short units of code" por ello he extraído la creación de los componentes de la interfaz gráfica en métodos individuales.

- **Bad Smell: Write simple units of code (class AlertakKudeatuGUI)**

- Código inicial

```
public AlertakKudeatuGUI(String username) {
                setBussinessLogic(TravelerGUI.getBusinessLogic());
                this.setSize(new Dimension(600, 400));
                this.setTitle(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.Alert"));
                this.setResizable(false);
                getContentPane().setLayout(new BorderLayout(0, 0));
                buttonPanel = new JPanel();
                buttonPanel.setLayout(new FlowLayout());
                getContentPane().add(buttonPanel, BorderLayout.SOUTH);
                List<Alert> alertList = appFacadeInterface.getAlertsByUsername(username);
                table = new JTable();
                JScrollPane scrollPane = new JScrollPane(table);
                getContentPane().add(scrollPane, BorderLayout.CENTER);
                table.getTableHeader().setReorderingAllowed(false);
                table.setColumnSelectionAllowed(false);
                table.setRowSelectionAllowed(true);
                table.setDefaultEditor(Object.class, null);
                SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
                String[] columnNames = {
ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.Zenbakia"),

ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.LeavingFrom"),

ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.GoingTo"),

ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.RideDate"),
                                ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.Aurkitua"),
                                ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.Aktibo") };
                DefaultTableModel model = new DefaultTableModel(columnNames, 0);
                if (alertList != null) {
                        for (Alert alert : alertList) {
                                String formattedDate = dateFormat.format(alert.getDate());
```

```
                                Object[] rowData = { alert.getAlertNumber(), alert.getFrom(), alert.getTo(),
formattedDate,
                                                alert.isFound(), alert.isActive() };
                                model.addRow(rowData);
                        }
                }
                table.setModel(model);
                table.getColumnModel().getColumn(0).setMinWidth(0);
                table.getColumnModel().getColumn(0).setMaxWidth(0);
                table.getColumnModel().getColumn(0).setWidth(0);
                statusLabel = new JLabel();
                getContentPane().add(statusLabel, BorderLayout.NORTH);
                addButton                                       =                                       new
JButton(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AddAlert"));
                addButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                JFrame a = new AlertaSortuGUI(username);
                                a.setVisible(true);
                                closeButton_actionPerformed(e);
                        }
                });
                buttonPanel.add(addButton);
                deleteButton                                    =                                       new
JButton(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.DeleteAlert"));
                deleteButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                int selectedRow = table.getSelectedRow();
                                if (selectedRow != -1) {
                                        int alertNumber = (int) table.getValueAt(selectedRow, 0);
                                        boolean deleted = appFacadeInterface.deleteAlert(alertNumber);
                                        if (deleted) {
                                                refreshTable(username);
                                        } else {

statusLabel.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.DeleteAlertError"));
                                        }
                                } else {

statusLabel.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.SelectAlert"));
                                }
                        }
                });
                buttonPanel.add(deleteButton);
                activateButton                                  =                                       new
JButton(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.ActivateAlert"));
                activateButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                int row = table.getSelectedRow();
                                if (row != -1) {
                                        int modelRow = table.convertRowIndexToModel(row);
                                        int  alertNumber  =  (int)  table.getModel().getValueAt(modelRow,
0);
```

```
                                        Alert al = appFacadeInterface.getAlert(alertNumber);
                                        if (al.isActive()) {

statusLabel.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AlertActive"));
                                        } else {
                                                statusLabel.setText("");
                                                al.setActive(true);
                                                appFacadeInterface.updateAlert(al);
                                                refreshTable(username);
                                        }
                                }
                        }
                });
                buttonPanel.add(activateButton);
                deactivateButton = new
JButton(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.DeactivateAlert"));
                deactivateButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                int row = table.getSelectedRow();
                                if (row != -1) {
                                        int modelRow = table.convertRowIndexToModel(row);
                                        int alertNumber = (int) table.getModel().getValueAt(modelRow,
0);
                                        Alert al = appFacadeInterface.getAlert(alertNumber);
                                        if (!al.isActive()) {

statusLabel.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AlertInactive"));
                                        } else {
                                                statusLabel.setText("");
                                                al.setActive(false);
                                                appFacadeInterface.updateAlert(al);
                                                refreshTable(username);
                                        }
                                }
                        }
                });
                buttonPanel.add(deactivateButton);
                closeButton = new JButton(ResourceBundle.getBundle("Etiquetas").getString("Close"));
                closeButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                closeButton_actionPerformed(e);
                        }
                });
                buttonPanel.add(closeButton);
        }
```

- Código refactorizado

```
public AlertakKudeatuGUI(String username) {
                setBussinessLogic(TravelerGUI.getBusinessLogic());
                this.setSize(new Dimension(600, 400));
```

```
                this.setTitle(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.Alert"));
                this.setResizable(false);
                getContentPane().setLayout(new BorderLayout(0, 0));
                buttonPanel = new JPanel();
                buttonPanel.setLayout(new FlowLayout());
                getContentPane().add(buttonPanel, BorderLayout.SOUTH);
                List<Alert> alertList = appFacadeInterface.getAlertsByUsername(username);
                table = new JTable();
                JScrollPane scrollPane = new JScrollPane(table);
                getContentPane().add(scrollPane, BorderLayout.CENTER);
                table.getTableHeader().setReorderingAllowed(false);
                table.setColumnSelectionAllowed(false);
                table.setRowSelectionAllowed(true);
                table.setDefaultEditor(Object.class, null);
                SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
                String[] columnNames = {
ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.Zenbakia"),

ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.LeavingFrom"),

ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.GoingTo"),

ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.RideDate"),
                        ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.Aurkitua"),
                        ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.Aktibo") };
                DefaultTableModel model = new DefaultTableModel(columnNames, 0);

                addAlertToTable(alertList, dateFormat, model);
                table.setModel(model);
                table.getColumnModel().getColumn(0).setMinWidth(0);
                table.getColumnModel().getColumn(0).setMaxWidth(0);
                table.getColumnModel().getColumn(0).setWidth(0);
                statusLabel = new JLabel();
                getContentPane().add(statusLabel, BorderLayout.NORTH);
                addButton = new
JButton(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AddAlert"));
                addButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                JFrame a = new AlertaSortuGUI(username);
                                a.setVisible(true);
                                closeButton_actionPerformed(e);
                        }
                });
                buttonPanel.add(addButton);
                deleteButton = new
JButton(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.DeleteAlert"));
                setDeleteButtonAction(username);
                buttonPanel.add(deleteButton);
                activateButton = new
JButton(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.ActivateAlert"));
                setActiveButtonAction(username);
                buttonPanel.add(activateButton);
```

```
                deactivateButton = new
JButton(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.DeactivateAlert"));
                setDeactivateButtonAction(username);
                buttonPanel.add(deactivateButton);
                closeButton = new JButton(ResourceBundle.getBundle("Etiquetas").getString("Close"));
                closeButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                closeButton_actionPerformed(e);
                        }
                });
                buttonPanel.add(closeButton);
        }
        private void addAlertToTable(List<Alert> alertList, SimpleDateFormat dateFormat,
DefaultTableModel model) {
                if (alertList != null) {
                        for (Alert alert : alertList) {
                                String formattedDate = dateFormat.format(alert.getDate());
                                Object[] rowData = { alert.getAlertNumber(), alert.getFrom(), alert.getTo(),
formattedDate,
                                                alert.isFound(), alert.isActive() };
                                model.addRow(rowData);
                        }
                }
        }
        private void setDeactivateButtonAction(String username) {
                deactivateButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                int row = table.getSelectedRow();
                                if (row != -1) {
                                        int modelRow = table.convertRowIndexToModel(row);
                                        int alertNumber = (int) table.getModel().getValueAt(modelRow,
0);
                                        Alert al = appFacadeInterface.getAlert(alertNumber);
                                        if (!al.isActive()) {

statusLabel.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AlertInactive"));
                                        } else {
                                                statusLabel.setText("");
                                                al.setActive(false);
                                                appFacadeInterface.updateAlert(al);
                                                refreshTable(username);
                                        }
                                }
                        }
                });
        }
        private void setActiveButtonAction(String username) {
                activateButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                int row = table.getSelectedRow();
                                if (row != -1) {
                                        int modelRow = table.convertRowIndexToModel(row);
```

```
                                        int  alertNumber  =  (int)  table.getModel().getValueAt(modelRow,
0);
                                        Alert al = appFacadeInterface.getAlert(alertNumber);
                                        if (al.isActive()) {

statusLabel.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.AlertActive"));
                                        } else {
                                                statusLabel.setText("");
                                                al.setActive(true);
                                                appFacadeInterface.updateAlert(al);
                                                refreshTable(username);
                                        }
                                }
                        }
                });
        }
        private void setDeleteButtonAction(String username) {
                deleteButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                int selectedRow = table.getSelectedRow();
                                if (selectedRow != -1) {
                                        int alertNumber = (int) table.getValueAt(selectedRow, 0);
                                        boolean deleted = appFacadeInterface.deleteAlert(alertNumber);
                                        if (deleted) {
                                                refreshTable(username);
                                        } else {

statusLabel.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.DeleteAlertError"));
                                        }
                                } else {

statusLabel.setText(ResourceBundle.getBundle("Etiquetas").getString("AlertGUI.SelectAlert"));
                                }
                        }
                });
        }
```

- Descripción del error concreto detectado y descripción de la refactorización realizada.

En el código inicial habían 12 puntos de decisiones y después de refactorizar (refactor - extract method) quedan 3 los correspondientes a las llamadas a los métodos que contienen aquellos puntos de decisiones.

- **Bad Smell: Duplicate code (class BookGUI)**

- Código inicial

```
        …
        private JLabel jLabelOrigin = new JLabel(
```

```
                        ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.LeavingFrom"));
        private JLabel jLabelDestination = new JLabel(

                        ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.GoingTo"));
        private final JLabel jLabelEventDate = new JLabel(

                        ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.RideDate"));
        private final JLabel jLabelEvents = new JLabel(

                        ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.Rides"));
        private final JLabel lblEmaitza = new JLabel();

        private final JLabel lblDesk = new JLabel(ResourceBundle.getBundle("Etiquetas").getString("DeskontuaGUI.Izena"));
        private JTextArea lbldekera = new JTextArea();
        private final JTextField txtDesk = new JTextField();

        private JButton jButtonBook = new JButton(ResourceBundle.getBundle("Etiquetas").getString("Book"));

        private JButton jButtonClose = new JButton(ResourceBundle.getBundle("Etiquetas").getString("Close"));

…
private String[] columnNamesRides = new String[] {

                        ResourceBundle.getBundle("Etiquetas").getString("FindRidesGUI.Driver"),

                        ResourceBundle.getBundle("Etiquetas").getString("FindRidesGUI.NPlaces"),

                        ResourceBundle.getBundle("Etiquetas").getString("FindRidesGUI.Price"),

                        ResourceBundle.getBundle("Etiquetas").getString("Balorazioa") };
        private final JLabel lblEserlekuak = new JLabel(

                        ResourceBundle.getBundle("Etiquetas").getString("FindRidesGUI.NPlaces"));
        private JComboBox<Integer> comboBoxSeats = new JComboBox<Integer>();
        private final JLabel jLabelSaldoa = new JLabel();

…

                this.setTitle(ResourceBundle.getBundle("Etiquetas").getString("TravelerGUI.BookRides"));



…

lblEmaitza.setText(ResourceBundle.getBundle("Etiquetas").getString("BookGUI.EserlekuError"));
                                        lblEmaitza.setForeground(Color.RED);
                                        return;
```

```
                                        }
                                        double ridePriceDesk = (selectedRide.getPrice() - desk) *
seatsRequested;
                                        double availableBalance =
appFacadeInterface.getActualMoney(username);
                                        if (availableBalance < ridePriceDesk) {


lblEmaitza.setText(ResourceBundle.getBundle("Etiquetas").getString("BookGUI.PriceError"));
                                                lblEmaitza.setForeground(Color.RED);
                                                return;
                                        }
                                        boolean bookingSuccess =
appFacadeInterface.bookRide(username, selectedRide, seatsRequested, desk);
                                        if (bookingSuccess) {
                                                Traveler traveler =
appFacadeInterface.getTraveler(username);
                                                appFacadeInterface.addMovement(traveler,
"BookFreeze", ridePriceDesk);
                                                double newBalance =
appFacadeInterface.getActualMoney(username);


jLabelSaldoa.setText(ResourceBundle.getBundle("Etiquetas").getString("MoneyGUI.Erabilgarri")
                                                                + newBalance + "\u20AC");
                                                int availableSeats = selectedRide.getnPlaces();
                                                tableModelRides.setValueAt(availableSeats, selectedRow,
1);


lblEmaitza.setText(ResourceBundle.getBundle("Etiquetas").getString("BookGUI.Booked"));
                                                lblEmaitza.setForeground(Color.BLACK);
                                                desk = 0;
                                                txtDesk.setText("");
                                                lbldekera.setText("");
                                                tableModelRides.setValueAt(selectedRide.getPrice(),
selectedRow, 2);
                                                comboBoxSeats.removeAllItems();
                                                for (int i = 1; i <= selectedRide.getnPlaces(); i++) {
                                                        comboBoxSeats.addItem(i);
                                                }
                                                if (selectedRide.getnPlaces() <= 0)
                                                        jButtonBook.setEnabled(false);
                                        } else {


lblEmaitza.setText(ResourceBundle.getBundle("Etiquetas").getString("BookGUI.BookingError"));
                                                lblEmaitza.setForeground(Color.RED);
                                        }
                                } else {
```

```
lblEmaitza.setText(ResourceBundle.getBundle("Etiquetas").getString("BookGUI.NoRide"));
                                                lblEmaitza.setForeground(Color.RED);
                                        }

…
if (rides.isEmpty())


jLabelEvents.setText(ResourceBundle.getBundle("Etiquetas").getString("FindRidesGUI.NoRides")
                                                                                        +           ":          "          +
dateformat1.format(calendarAct.getTime()));
                                                        else


jLabelEvents.setText(ResourceBundle.getBundle("Etiquetas").getString("FindRidesGUI.Rides")
                                                                                        +           ":          "          +
dateformat1.format(calendarAct.getTime()));
                                                        for (domain.Ride ride : rides) {

…

.setText(ResourceBundle.getBundle("Etiquetas").getString("MoneyGUI.Erabilgarri") + diruKop + "\u20AC");
                        jLabelSaldoa.setHorizontalAlignment(SwingConstants.LEFT);


…

JButton                                 jButtonDesk                                     =                                       new
JButton(ResourceBundle.getBundle("Etiquetas").getString("DeskontuaGUI.Aplikatu"));
                        jButtonDesk.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                        int pos = tableRides.getSelectedRow();
                                        if (pos != -1) {
                                                if (txtDesk.getText() != null) {
                                                        Discount                        dis                             =
appFacadeInterface.getDiscount(txtDesk.getText());
                                                        if (dis != null && dis.isActive()) {
                                                                lbldekera.setText(


ResourceBundle.getBundle("Etiquetas").getString("DeskontuaGUI.AplikatuDa"));
                                                                double deskontua = dis.getPortzentaia();
                                                                Ride            selectedRide            =           (Ride)
tableModelRides.getValueAt(pos, 4);
                                                                desk = ((selectedRide.getPrice()) * deskontua);

tableModelRides.setValueAt(selectedRide.getPrice() - desk, pos, 2);
                                                        } else {


lbldekera.setText(ResourceBundle.getBundle("Etiquetas").getString("DeskontuaGUI.Error"));
```

```
                                                }
                                            }
                                        } else {


lbldekera.setText(ResourceBundle.getBundle("Etiquetas").getString("BookGUI.NoRide"));
                                        }

…
```

- Código refactorizado

```
…
public static final String ETIQUETA = "Etiquetas";
…
private JLabel jLabelOrigin = new JLabel(


ResourceBundle.getBundle(ETIQUETA).getString("CreateRideGUI.LeavingFrom"));
        private JLabel jLabelDestination = new JLabel(

                                ResourceBundle.getBundle(ETIQUETA).getString("CreateRideGUI.GoingTo"));
        private final JLabel jLabelEventDate = new JLabel(

                                ResourceBundle.getBundle(ETIQUETA).getString("CreateRideGUI.RideDate"));
        private final JLabel jLabelEvents = new JLabel(

                                ResourceBundle.getBundle(ETIQUETA).getString("CreateRideGUI.Rides"));
        private final JLabel lblEmaitza = new JLabel();

        private final JLabel lblDesk = new
JLabel(ResourceBundle.getBundle(ETIQUETA).getString("DeskontuaGUI.Izena"));
        private JTextArea lbldekera = new JTextArea();
        private final JTextField txtDesk = new JTextField();

        private JButton jButtonBook = new
JButton(ResourceBundle.getBundle(ETIQUETA).getString("Book"));

        private JButton jButtonClose = new
JButton(ResourceBundle.getBundle(ETIQUETA).getString("Close"));

…
private String[] columnNamesRides = new String[] {

                                ResourceBundle.getBundle(ETIQUETA).getString("FindRidesGUI.Driver"),

                                ResourceBundle.getBundle(ETIQUETA).getString("FindRidesGUI.NPlaces"),

                                ResourceBundle.getBundle(ETIQUETA).getString("FindRidesGUI.Price"),
```

```
                        ResourceBundle.getBundle(ETIQUETA).getString("Balorazioa") };
        private final JLabel lblEserlekuak = new JLabel(

                        ResourceBundle.getBundle(ETIQUETA).getString("FindRidesGUI.NPlaces"));

…

this.setTitle(ResourceBundle.getBundle(ETIQUETA).getString("TravelerGUI.BookRides"));

…

lblEmaitza.setText(ResourceBundle.getBundle(ETIQUETA).getString("BookGUI.EserlekuError"));
                                lblEmaitza.setForeground(Color.RED);
                                return;
                        }
                        double ridePriceDesk = (selectedRide.getPrice() - desk) *
seatsRequested;
                        double availableBalance =
appFacadeInterface.getActualMoney(username);
                        if (availableBalance < ridePriceDesk) {


lblEmaitza.setText(ResourceBundle.getBundle(ETIQUETA).getString("BookGUI.PriceError"));
                                lblEmaitza.setForeground(Color.RED);
                                return;
                        }
                        boolean bookingSuccess =
appFacadeInterface.bookRide(username, selectedRide, seatsRequested, desk);
                        if (bookingSuccess) {
                                Traveler traveler =
appFacadeInterface.getTraveler(username);
                                appFacadeInterface.addMovement(traveler,
"BookFreeze", ridePriceDesk);
                                double newBalance =
appFacadeInterface.getActualMoney(username);


jLabelSaldoa.setText(ResourceBundle.getBundle(ETIQUETA).getString("MoneyGUI.Erabilgarri")
                                                + newBalance + "\u20AC");
                                int availableSeats = selectedRide.getnPlaces();
                                tableModelRides.setValueAt(availableSeats, selectedRow,
1);


lblEmaitza.setText(ResourceBundle.getBundle(ETIQUETA).getString("BookGUI.Booked"));
                                lblEmaitza.setForeground(Color.BLACK);

…

lblEmaitza.setText(ResourceBundle.getBundle(ETIQUETA).getString("BookGUI.BookingError"));
                                lblEmaitza.setForeground(Color.RED);
```

```
                                        }
                                    } else {


lblEmaitza.setText(ResourceBundle.getBundle(ETIQUETA).getString("BookGUI.NoRide"));
                                        lblEmaitza.setForeground(Color.RED);

…

jLabelEvents.setText(ResourceBundle.getBundle(ETIQUETA).getString("FindRidesGUI.NoRides")
                                                                    +       ":      "       +
dateformat1.format(calendarAct.getTime()));
                                            else


jLabelEvents.setText(ResourceBundle.getBundle(ETIQUETA).getString("FindRidesGUI.Rides")
                                                                    +       ":      "       +
dateformat1.format(calendarAct.getTime()));

…

.setText(ResourceBundle.getBundle(ETIQUETA).getString("MoneyGUI.Erabilgarri") + diruKop + "\u20AC");
                    jLabelSaldoa.setHorizontalAlignment(SwingConstants.LEFT);

…

JButton                 jButtonDesk                 =                   new
JButton(ResourceBundle.getBundle(ETIQUETA).getString("DeskontuaGUI.Aplikatu"));
                    jButtonDesk.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                            int pos = tableRides.getSelectedRow();
                            if (pos != -1) {
                                if (txtDesk.getText() != null) {
                                    Discount                dis                 =
appFacadeInterface.getDiscount(txtDesk.getText());
                                    if (dis != null && dis.isActive()) {
                                        lbldekera.setText(


ResourceBundle.getBundle(ETIQUETA).getString("DeskontuaGUI.AplikatuDa"));
                                        double deskontua = dis.getPortzentaia();
                                        Ride        selectedRide        =       (Ride)
tableModelRides.getValueAt(pos, 4);
                                        desk = ((selectedRide.getPrice()) * deskontua);

tableModelRides.setValueAt(selectedRide.getPrice() - desk, pos, 2);
                                    } else {


lbldekera.setText(ResourceBundle.getBundle(ETIQUETA).getString("DeskontuaGUI.Error"));
                                    }
```

```
                                        }
                                    } else {


lbldekera.setText(ResourceBundle.getBundle(ETIQUETA).getString("BookGUI.NoRide"));
                                    }

…
```

- Descripción del error concreto detectado y descripción de la refactorización realizada.

Había un string, "Etiquetas" que aparecía duplicado 26 veces, entonces lo he extraído en una constante mediante refactor - extract constant.

- **Bad Smell: Keep unit interfaces small (class BLFacadeImplementation)**

- Código inicial

```
        public Ride createRide(String from, String to, Date date, int nPlaces, float price, String driverName)
                        throws RideMustBeLaterThanTodayException, RideAlreadyExistException {
                dbManager.open();
                Ride ride = dbManager.createRide(from, to, date, nPlaces, price, driverName);
                dbManager.close();
                return ride;
        }
```

- Código refactorizado

```
        public Ride createRide(RequestRide rr)
                        throws RideMustBeLaterThanTodayException, RideAlreadyExistException {
                dbManager.open();
                Ride ride = dbManager.createRide(rr.getFrom(), rr.getTo(), rr.getDate(), rr.getnPlaces(),
rr.getPrice(), rr.getDriverName());
                dbManager.close();
                return ride;
        }
```

- Descripción del error concreto detectado y descripción de la refactorización realizada.

  El método tomaba como parámetros de entrada más de 4 parámetros por lo que he creado una nueva clase RequestRide para agrupar aquellos parámetros y he hecho refactor - change method signature y he eliminado todos los parámetros y añadido uno nuevo el correspondiente a RequestRide.