LABORATORIO DE REFACTORIZACIÓN

Autores: David Murguialday, Haritz Gomez e Igor Da Silva

createRide : "Write short units of code”

Código antes:

```
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;
    }

}
```

Código refactorizado:

```
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 {
        validateRideDate(date);
        db.getTransaction().begin();
        Driver driver = db.find(Driver.class, driverName);
        doesRideExistInDriver(driver, from, to, date);
        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;
    }

}
```

Descripción del error y de la refactorización:

Función con muchas líneas de código. He creado dos funciones que realizan las funcionalidades extras que hacia la función create ride. Extract Method

```
public 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"));
    }
}

public void doesRideExistInDriver(Driver driver, String from, String to, Date date) throws RideAlreadyExistException{
    if (driver.doesRideExists(from, to, date)) {
        db.getTransaction().commit();
        throw new RideAlreadyExistException(
                ResourceBundle.getBundle("Etiquetas").getString("DataAccess.RideAlreadyExist"));
    }
}
```

Autor: David

GauzatuErakigeta: "Write simple units of code"

Código antes:

```
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) {
            doDeposit(user, amount, deposit);
            db.merge(user);
            db.getTransaction().commit();
            return true;
        }
        db.getTransaction().commit();
        return false;
    } catch (Exception e) {
        e.printStackTrace();
        db.getTransaction().rollback();
        return false;
    }
}
```

Descripción del error y de la refactorización:

Función con muchas ramas, ifs . He creado una función que se encarga de realizar el depósito, que a su vez llama a una función que se encarga de hacer el retiro de dinero. Así en la función original solo tenemos un if, y en las nuevas funciones un if y un else.

```
public void doDeposit(User user, double amount, boolean deposit, double currentMoney) {
    if (deposit) {
        user.setMoney(currentMoney + amount);
    } else {
        withdrawMoney(user, amount, currentMoney);
    }

}

public void withdrawMoney(User user, double amount, double currentMoney) {
    if ((currentMoney - amount) < 0) {
        user.setMoney(0);
    }
    else {
        user.setMoney(currentMoney - amount);
    }
}
```

Autor: David

erreklamazioaBidali: "Keep unit interfaces small "

Código antes:

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

Descripción del error y de la refactorización:

Función con muchos parámetros (6). En lugar de pasarle todos los parámetros necesarios para después crear un objeto de tipo complaint, he hecho que se le pase como parámetro directamente el objeto de tipo complaint.

Autor: David

BezeroGUI “Duplicate code”

Código antes:

```
    if (TravelsList != null) {
        for (Booking bo : TravelsList) {

            String status;
            switch (bo.getStatus()) {

            case "Completed":
                status = ResourceBundle.getBundle("Etiquetas").getString("Completed");
                break;
            case "Accepted":
                status = ResourceBundle.getBundle("Etiquetas").getString("Accepted");
                break;
            case "Rejected":
                status = ResourceBundle.getBundle("Etiquetas").getString("Rejected");
                break;
            case "NotCompleted":
                status = ResourceBundle.getBundle("Etiquetas").getString("NotCompleted");
                break;
            case "Complained":
                status = ResourceBundle.getBundle("Etiquetas").getString("Complained");
                break;
            Duplication
            case 1 "Valued":
                Duplication
                status = ResourceBundle.getBundle("Etiquetas").getString( 2 "Valued");
                break;
```

```
            }

        Duplication
        } else if (bo.getStatus().equals("Completed") || bo.getStatus().equals( 3 "Valued")

                || bo.getStatus().equals("Complained")) {
            Object[] rowData = { bo.getBookNumber(), dateFormat.format(bo.getRide().getDate()),
                    bo.getTraveler().getUsername(), status, "" };
            model.addRow(rowData);
            BezeroLista.add(bo);
        }

    }
```

```
// Baloratu botoia
jButtonBaloratu = new JButton(ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.Baloratu"));
jButtonBaloratu.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int pos = taula.getSelectedRow();
        if (pos != -1) {
            Booking bo = BezeroLista.get(pos);

            if (bo.getStatus().equals("Completed")) {
                Duplication
                bo.setStatus( 4 "Valued");
                appFacadeInterface.updateBooking(bo);
                JFrame a = new BaloraGUI(bo.getTraveler().getUsername());
                a.setVisible(true);
                Duplication
                model.setValueAt(ResourceBundle.getBundle("Etiquetas").getString( 5 "Valued"), pos, 3);
            Duplication
            } else if (bo.getStatus().equals(ResourceBundle.getBundle("Etiquetas").getString( 6 "Valued"))) {
                lblErrorea.setForeground(Color.RED);
                lblErrorea.setText(
                        ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.BezeroaJadanikBaloratuta"));
            } else {
                lblErrorea.setForeground(Color.RED);
                lblErrorea.setText(
                        ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.AukeratuOsatutakoBidaia"));
            }
        } else {
            lblErrorea.setForeground(Color.RED);
            lblErrorea.setText(ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.Erroraukera"));
```

Código refactorizado:

```
public class BezeroGUI extends JFrame {

    private static final long serialVersionUID = 1L;
    private static BLFacade appFacadeInterface;
    private JTable taula;
    private JButton jButtonBaloratu;
    private JButton jButtonErreklamatu;
    private JButton jButtonClose;
    private JScrollPane scrollPane;
    private static String valued = "Valued";
```

```
        Duplication
        case "valued":
            Duplication
            status = ResourceBundle.getBundle("Etiquetas").getString(valued);
            break;
        default:
```

```
    } else if (bo.getStatus().equals("Completed") || bo.getStatus().equals(valued)

            || bo.getStatus().equals("Complained")) {
        Object[] rowData = { bo.getBookNumber(), dateFormat.format(bo.getRide().getDate()),
                bo.getTraveler().getUsername(), status, "" };
        model.addRow(rowData);
        BezeroLista.add(bo);
    }
```

```
    if (bo.getStatus().equals("Completed")) {

        bo.setStatus(valued);
        appFacadeInterface.updateBooking(bo);
        JFrame a = new BaloraGUI(bo.getTraveler().getUsername());
        a.setVisible(true);

        model.setValueAt(ResourceBundle.getBundle("Etiquetas").getString(valued), pos, 3);

    } else if (bo.getStatus().equals(ResourceBundle.getBundle("Etiquetas").getString(valued))) {
        lblErrorea.setForeground(Color.RED);
        lblErrorea.setText(
                ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.BezeroaJadanikBaloratu
    } else {
        lblErrorea.setForeground(Color.RED);
        lblErrorea.setText(
                ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.AukeratuOsatutakoBidai
    }
```

Descripción del error y de la refactorización:

Aparece un literal repetido 6 veces por toda la clase. He definido una constante que contenga el valor del literal y la he sustituido en el código.

Autor: David

CancelRide: "Write short units of code”

Codigo antes de refactorizar:

```
public void cancelRide(Ride ride) {
    try {
        db.getTransaction().begin();

        for (Booking booking : ride.getBookings()) {
            if (booking.getStatus().equals("Accepted") || booking.getStatus().equals("NotDefined")) {
                double price = booking.prezioaKalkulatu();
                Traveler traveler = booking.getTraveler();
                double frozenMoney = traveler.getIzoztatutakoDirua();
                traveler.setIzoztatutakoDirua(frozenMoney - price);

                double money = traveler.getMoney();
                traveler.setMoney(money + price);
                db.merge(traveler);
                db.getTransaction().commit();
                addMovement(traveler, "BookDeny", price);
                db.getTransaction().begin();
            }
            booking.setStatus("Rejected");
            db.merge(booking);
        }

        ride.setActive(false);
        db.merge(ride);

        db.getTransaction().commit();

    } catch (Exception e) {
        if (db.getTransaction().isActive()) {
            db.getTransaction().rollback();
        }

        e.printStackTrace();
    }
}
```

Código después de la refactorización:

```java
public void cancelRide(Ride ride) {
    try {
        db.getTransaction().begin();

        for (Booking booking : ride.getBookings()) {
            manageBooking(booking);
            booking.setStatus("Rejected");
            db.merge(booking);
        }

        ride.setActive(false);
        db.merge(ride);

        db.getTransaction().commit();

    } catch (Exception e) {
        if (db.getTransaction().isActive()) {
            db.getTransaction().rollback();
        }

        e.printStackTrace();
    }
}
```

```java
public void manageBooking(Booking booking) {
    if (booking.getStatus().equals("Accepted") || booking.getStatus().equals("NotDefined")) {
        double price = booking.prezioaKalkulatu();
        Traveler traveler = booking.getTraveler();
        double frozenMoney = traveler.getIzoztatutakoDirua();
        traveler.setIzoztatutakoDirua(frozenMoney - price);

        double money = traveler.getMoney();
        traveler.setMoney(money + price);
        db.merge(traveler);
        db.getTransaction().commit();
        addMovement(traveler, "BookDeny", price);
        db.getTransaction().begin();
    }
}
```

Descripción del error y refactorización:

La función original tiene más de 15 líneas de código. Para corregirlo, se pasa la parte que gestiona la cancelación de las reservas del viaje a otro método que se especialice en ello (manageBooking).

Autor: Haritz

DeleteUser: "Write simple units of code”

Código antes:

```
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 después:

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

public void deleteOtherUser(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());
        }
    }
}

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

Descripción del error y refactorización:

La función tenía demasiadas ramas y por lo tanto una complejidad ciclomática muy elevada. Para corregirlo, se crean dos funciones, que eliminaran el usuario dependiendo del tipo de este (Driver u otro tipo distinto), esto hará que la complejidad ciclomática se divida en cada uno de los métodos

Autor: Haritz

InitializeBD: “Duplicate code”

Codigo antes de refactorizar:

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

Código después de refactorizar:

```
private static final String STATUS_ACCEPTED = "Accepted";
```

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

Descripción del error y refactorización:

El código repite 4 veces setStatus(“Accepted”), en vez de eso, es más recomendable inicializar una constante en la clase y utilizarla cada vez que se quiera poner como “Accepted” el valor. (También se debería hacer lo mismo para “Rejected”).

Autor: Haritz

createRide: "Keep unit interfaces small "

Código antes:

```
public Ride createRide(String from, String to, Date date, int nPlaces, float price, String driverName)
```

Código después:

```
public class ValoresViaje {
    private String from;
    private String to;
    private Date date;
    private int nPlaces;
    private float price;
    private String driverName;

    public ValoresViaje(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;
    }
}
```

```
public Ride createRide(ValoresViaje vals)
        throws RideAlreadyExistException, RideMustBeLaterThanTodayException {
    System.out.println(
            ">> DataAccess: createRide=> from= " + vals.getFrom() + " to= " + vals.getTo() + " driver=" + vals.getDrive
    if (vals.getDriverName()==null) return null;
    try {
        if (new Date().compareTo(vals.getDate()) > 0) {
            System.out.println("ppppp");
            throw new RideMustBeLaterThanTodayException(
                    ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.ErrorRideMustBeLaterThanToday"));
        }

        db.getTransaction().begin();
        Driver driver = db.find(Driver.class, vals.getDriverName());
        if (driver.doesRideExists(vals.getFrom(), vals.getTo(), vals.getDate())) {
            db.getTransaction().commit();
            throw new RideAlreadyExistException(
                    ResourceBundle.getBundle("Etiquetas").getString("DataAccess.RideAlreadyExist"));
        }
        Ride ride = driver.addRide(vals.getFrom(), vals.getTo(), vals.getDate(), vals.getNPlaces(), vals.getPrice());
        // next instruction can be obviated
        db.persist(driver);
        db.getTransaction().commit();

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

Descripción del error y refactorización:

El código tenía demasiados parámetros. Para corregirlo, se crea una clase llamada ValoresViaje que contiene todos los parámetros necesarios para que el método siga funcionando y se pasa una instancia de esa clase como parámetro.

Autor: Haritz

BezeroGUI: Código duplicado

```
String[] columnNames = { ResourceBundle.getBundle("Etiquetas").getString("EgoeraGUI.BookingNumber"),
        ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.RideDate"),
        ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.Bezeroa"),
        ResourceBundle.getBundle("Etiquetas").getString("EgoeraGUI.Egoera"),
        ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.Zergatia") };
DefaultTableModel model = new DefaultTableModel(columnNames, 0);
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");

if (TravelsList != null) {
    for (Booking bo : TravelsList) {

        String status;
        switch (bo.getStatus()) {
        Duplication
        case 1 "Completed":
            Duplication
            status = ResourceBundle.getBundle("Etiquetas").getString( 2 "Completed");
            break;
        case "Accepted":
            status = ResourceBundle.getBundle("Etiquetas").getString("Accepted");
            break;
        case "Rejected":
            status = ResourceBundle.getBundle("Etiquetas").getString("Rejected");
            break;
        case "NotCompleted":
            status = ResourceBundle.getBundle("Etiquetas").getString("NotCompleted");
            break;
        case "Complained":
            status = ResourceBundle.getBundle("Etiquetas").getString("Complained");
            break;
        case "Valued":
            status = ResourceBundle.getBundle("Etiquetas").getString("Valued");
            break;
        default:
            status = ResourceBundle.getBundle("Etiquetas").getString("NotDefined");
            break;
```

```
        if (bo.getStatus().equals("NotCompleted")) {
            Complaint er = appFacadeInterface.getComplaintsByBook(bo);
            if (er != null) {
                if (er.getAurkeztua()) {
                    er.setEgoera("Erreklamazioa");
                } else {
                    er.setEgoera("Ez aurkeztua");
                }

                Object[] rowData = { bo.getBookNumber(), dateFormat.format(bo.getRide().getDate()),
                        bo.getTraveler().getUsername(), status, er.getEgoera() };
                model.addRow(rowData);
                BezeroLista.add(bo);
            }

        Duplication
        } else if (bo.getStatus().equals("Completed") || bo.getStatus().equals("Valued")
                || bo.getStatus().equals("Complained")) {
            Object[] rowData = { bo.getBookNumber(), dateFormat.format(bo.getRide().getDate()),
                    bo.getTraveler().getUsername(), status, "" };
            model.addRow(rowData);
            BezeroLista.add(bo);
        }

    }
}
taula.setModel(model);

taula.getTableHeader().setReorderingAllowed(false);
taula.setColumnSelectionAllowed(false);
taula.setRowSelectionAllowed(true);
taula.setDefaultEditor(Object.class, null);
taula.setModel(model);

// Erroreen textua
JLabel lblErrorea = new JLabel();
```

```
        // Erroreen textua
        JLabel lblErrorea = new JLabel();
        this.getContentPane().add(lblErrorea, BorderLayout.CENTER);

        this.setTitle(ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.Bezeroak"));

        // Baloratu botoia
        jButtonBaloratu = new JButton(ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.Baloratu"));
        jButtonBaloratu.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int pos = taula.getSelectedRow();
                if (pos != -1) {
                    Booking bo = BezeroLista.get(pos);
                    Duplication
                    if (bo.getStatus().equals("Completed")) {
                        bo.setStatus("Valued");
                        appFacadeInterface.updateBooking(bo);
                        JFrame a = new BaloraGUI(bo.getTraveler().getUsername());
                        a.setVisible(true);
                        model.setValueAt(ResourceBundle.getBundle("Etiquetas").getString("Valued"), pos, 3);
                    } else if (bo.getStatus().equals(ResourceBundle.getBundle("Etiquetas").getString("Valued"))) {
                        lblErrorea.setForeground(Color.RED);
                        lblErrorea.setText(
                                ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.BezeroaJadanikBaloratuta"));
                    } else {
                        lblErrorea.setForeground(Color.RED);
                        lblErrorea.setText(
                                ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.AukeratuOsatutakoBidaia"));
                    }
                } else {
                    lblErrorea.setForeground(Color.RED);
                    lblErrorea.setText(ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.Erroraukera"));
                }

            }
        });
        // Erreklamazio botoia
        jButtonErreklamatu = new JButton(ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.Onartu")
                + " / " + ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.Erreklamatu"));
        jButtonErreklamatu.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int pos = taula.getSelectedRow();
                if (pos != -1) {
                    Booking booking = BezeroLista.get(pos);
                    if (!taula.getValueAt(pos, 4).equals("")) {
                        double prez = booking.prezioaKalkulatu();
                        if (taula.getValueAt(pos, 4).equals("Erreklamazioa")) {
                            Traveler traveler = booking.getTraveler();

                            booking.setStatus("Complained");
                            appFacadeInterface.updateBooking(booking);

                            traveler.setIzoztatutakoDirua(traveler.getIzoztatutakoDirua() - prez);
                            appFacadeInterface.updateTraveler(traveler);

                            appFacadeInterface.gauzatuEragiketa(traveler.getUsername(), prez, true);
                            appFacadeInterface.addMovement(traveler, "UnfreezeNotComplete", prez);

                            Driver driver = appFacadeInterface.getDriver(username);
                            driver.setErreklamaKop(driver.getErreklamaKop() + 1);

                            lblErrorea.setText(
                                    ResourceBundle.getBundle("Etiquetas").getString("BezeroGUI.ComplaintAccepted"));
                            lblErrorea.setForeground(Color.BLACK);

                            model.setValueAt(ResourceBundle.getBundle("Etiquetas").getString("Complained"), pos, 3);
                            model.setValueAt("", pos, 4);
                        } else if (taula.getValueAt(pos, 4).equals("Ez aurkeztua")) {
                            Driver driver = booking.getRide().getDriver();
                            Traveler traveler = booking.getTraveler();

                            booking.setStatus("Complained");
```

Resultado de la refactorización (centrado en la cadena de caracteres “Etiquetas”)

```
final String completed = "Completed";
final String accepted = "Accepted";
final String rejected = "Rejected";
final String notCompleted = "NotCompleted";
final String complained = "Complained";
final String valued = "Valued";
final String notDefined = "NotDefined";
final String etiquetas = "Etiquetas";
```

```
if (TravelsList != null) {

    for (Booking bo : TravelsList) {

        String status;

        switch (bo.getStatus()) {
        case completed:
            status = ResourceBundle.getBundle(etiquetas).getString(completed);
            break;
        case accepted:
            status = ResourceBundle.getBundle(etiquetas).getString(accepted);
            break;
        case rejected:
            status = ResourceBundle.getBundle(etiquetas).getString(rejected);
            break;
        case notCompleted:
            status = ResourceBundle.getBundle(etiquetas).getString(notCompleted);
            break;
        case complained:
            status = ResourceBundle.getBundle(etiquetas).getString(complained);
            break;
        case valued:
            status = ResourceBundle.getBundle(etiquetas).getString(valued);
            break;
        default:
            status = ResourceBundle.getBundle(etiquetas).getString(notDefined);
            break;
        }
```

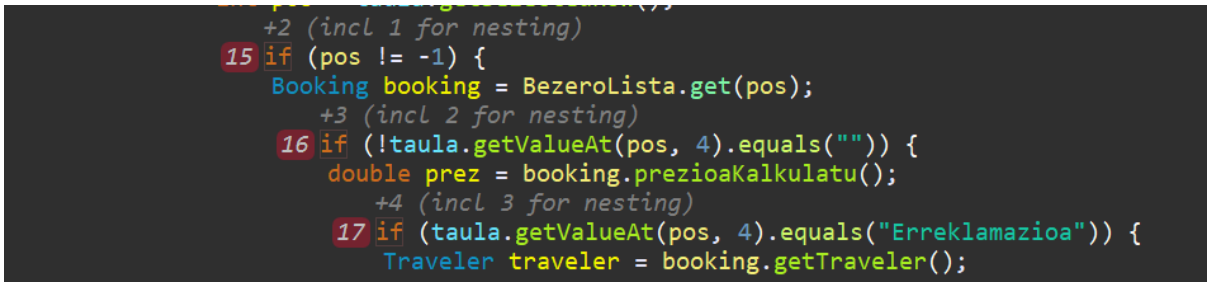
Descripción de la refactorización: Esta refactorización optimiza el código eliminando código duplicado (la cadena de caracteres “Etiquetas”) y lo sustituye por una constante en la que se declara dicho código duplicado

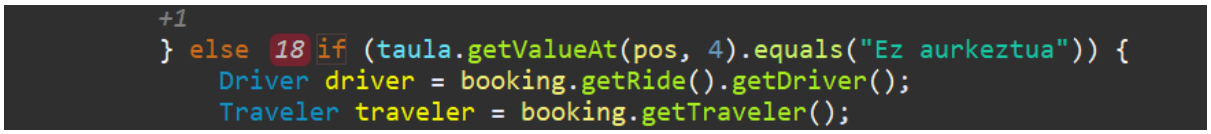
Autor: Igor

BezeroGUI: Write simple units of code

```
+1
1 if (TravelsList != null) {
    +2 (incl 1 for nesting)
  2 for (Booking bo : TravelsList) {

      String status;
        +3 (incl 2 for nesting)
       3 switch (bo.getStatus()) {
      case "Completed":
          status = ResourceBundle.getBundle(etiquetas).getString(completed);
          break;
      case "Accepted":
          status = ResourceBundle.getBundle(etiquetas).getString(accepted);
          break;
      case "Rejected":
          status = ResourceBundle.getBundle(etiquetas).getString(rejected);
          break;
      case "NotCompleted":
          status = ResourceBundle.getBundle(etiquetas).getString(notCompleted);
          break;
      case "Complained":
          status = ResourceBundle.getBundle(etiquetas).getString(complained);
          break;
      case "Valued":
          status = ResourceBundle.getBundle(etiquetas).getString(valued);
          break;
      default:
          status = ResourceBundle.getBundle(etiquetas).getString(notDefined);
          break;
      }

        +3 (incl 2 for nesting)
       4 if (bo.getStatus().equals(notCompleted)) {
          Complaint er = appFacadeInterface.getComplaintsByBook(bo);
          +4 (incl 3 for nesting)
          if (er != null){
```

```
        +3 (incl 2 for nesting)
      4 if (bo.getStatus().equals(notCompleted)) {
          Complaint er = appFacadeInterface.getComplaintsByBook(bo);
          +4 (incl 3 for nesting)
          if (er != null)
                +5 (incl 4 for nesting)
             6 if (er.getAurkeztua()) {
                er.setEgoera("Erreklamazioa");
             +1
             } 7 else {
                er.setEgoera("Ez aurkeztua");
             }
```



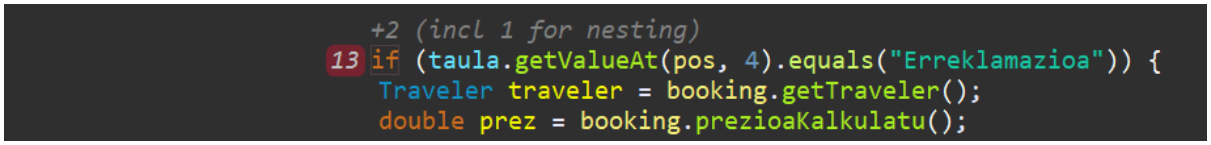


Después de haber tratado de reducir la complejidad cognitiva (esperado de 43 a 15):

```
                +1
        1 if (TravelsList != null) {
                +2 (incl 1 for nesting)
            2 for (Booking bo : TravelsList) {

                String status;
                  +3 (incl 2 for nesting)
                3 switch (bo.getStatus()) {
                case "Completed":
                    status = ResourceBundle.getBundle(etiquetas).getString(completed);
                    break;
                case "Accepted":
                    status = ResourceBundle.getBundle(etiquetas).getString(accepted);
                    break;
                case "Rejected":
                    status = ResourceBundle.getBundle(etiquetas).getString(rejected);
                    break;
                case "NotCompleted":
                    status = ResourceBundle.getBundle(etiquetas).getString(notCompleted);
                    break;
                case "Complained":
                    status = ResourceBundle.getBundle(etiquetas).getString(complained);
                    break;
                case "Valued":
```

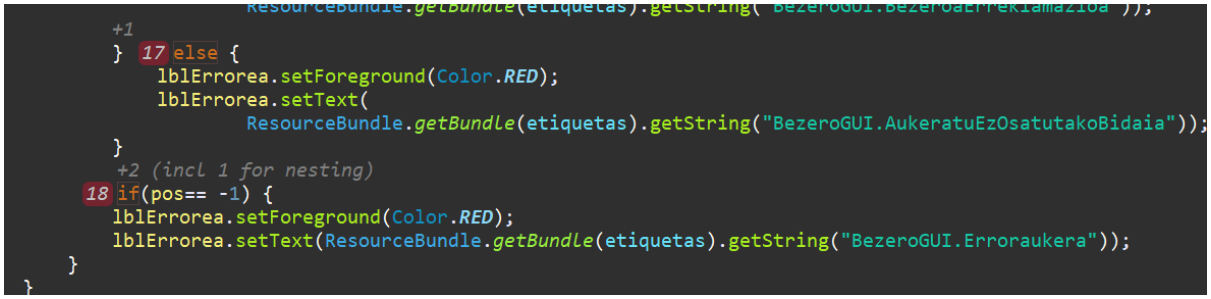
```
                +3 (incl 2 for nesting)
            4 if (bo.getStatus().equals(notCompleted)) {
                Complaint er = appFacadeInterface.getComplaintsByBook(bo);
                      +4 (incl 3 for nesting)
                    5 if (er.getAurkeztua()) {
                        er.setEgoera("Erreklamazioa");
                    +1
                    } 6 else {
                        er.setEgoera("Ez aurkeztua");
                    }

                    Object[] rowData = { bo.getBookNumber(), dateFormat.format(bo.getRide().getDate()),
                            bo.getTraveler().getUsername(), status, er.getEgoera() };
                    model.addRow(rowData);
                    BezeroLista.add(bo);

            +1 | +1
            } else 7 if (bo.getStatus().equals(completed) || bo.getStatus().equals(valued)
                    || bo.getStatus().equals(complained)) {
                Object[] rowData = { bo.getBookNumber(), dateFormat.format(bo.getRide().getDate()),
                        bo.getTraveler().getUsername(), status, "" };
```

```
                +2 (incl 1 for nesting)
            9 if (bo.getStatus().equals(completed)) {
                bo.setStatus(valued);
                appFacadeInterface.updateBooking(bo);
                JFrame a = new BaloraGUI(bo.getTraveler().getUsername());
                a.setVisible(true);
                model.setValueAt(ResourceBundle.getBundle(etiquetas).getString(valued), pos, 3);
            +1
            } else 10 if (bo.getStatus().equals(ResourceBundle.getBundle(etiquetas).getString(valued))) {
                lblErrorea.setForeground(Color.RED);
                lblErrorea.setText(
                        ResourceBundle.getBundle(etiquetas).getString("BezeroGUI.BezeroaJadanikBaloratuta"));
            +1
            } 11 else {
                lblErrorea.setForeground(Color.RED);
                lblErrorea.setText(
                        ResourceBundle.getBundle(etiquetas).getString("BezeroGUI.AukeratuOsatutakoBidaia"));
            }
        +2 (incl 1 for nesting)
    12 if(pos == -1) {
```



```
                    +2 (incl 1 for nesting)
                13 if (taula.getValueAt(pos, 4).equals("Erreklamazioa")) {
                    Traveler traveler = booking.getTraveler();
                    double prez = booking.prezioaKalkulatu();
```

```
            } else 14 if (taula.getValueAt(pos, 4).equals("Ez aurkeztua")) {
                Driver driver = booking.getRide().getDriver();
                Traveler traveler = booking.getTraveler();
                double prez = booking.prezioaKalkulatu();

                booking.setStatus(complained);
                appFacadeInterface.updateBooking(booking);

                traveler.setIzoztatutakoDirua(traveler.getIzoztatutakoDirua() - prez);
                traveler.setErreklamaKop(traveler.getErreklamaKop() + 1);
                appFacadeInterface.updateTraveler(traveler);

                appFacadeInterface.gauzatuEragiketa(driver.getUsername(), prez, true);
                appFacadeInterface.addMovement(traveler, "UnfreezeCompleteT", 0);
                appFacadeInterface.addMovement(driver, "UnfreezeCompleteD", prez);
                lblErrorea.setText(
                        ResourceBundle.getBundle(etiquetas).getString("BezeroGUI.ComplaintComplete"));
                lblErrorea.setForeground(Color.BLACK);

                model.setValueAt(ResourceBundle.getBundle(etiquetas).getString(complained), pos, 3);
                model.setValueAt("", pos, 4);
            +1
            } 15 else {
                lblErrorea.setForeground(Color.RED);
                lblErrorea.setText(ResourceBundle.getBundle(etiquetas)
                        .getString("BezeroGUI.AukeratuEzOsatutakoBidaia"));
            }
             +2 (incl 1 for nesting)
           16 if (booking.getStatus()
                .equals(ResourceBundle.getBundle(etiquetas).getString(complained))) {
            lblErrorea.setForeground(Color.RED);
            lblErrorea.setText(
                    ResourceBundle.getBundle(etiquetas).getString("BezeroGUI.BezeroaErreklamazioa"));
```



```
                ResourceBundle.getBundle(etiquetas).getString("BezeroGUI.BezeroaErreklamazioa"));
        +1
        } 17 else {
            lblErrorea.setForeground(Color.RED);
            lblErrorea.setText(
                    ResourceBundle.getBundle(etiquetas).getString("BezeroGUI.AukeratuEzOsatutakoBidaia"));
        }
         +2 (incl 1 for nesting)
      18 if(pos== -1) {
        lblErrorea.setForeground(Color.RED);
        lblErrorea.setText(ResourceBundle.getBundle(etiquetas).getString("BezeroGUI.Erroraukera"));
    }
}
```

Resultado: Complejidad cognitiva reducida de 43 a 31, posiblemente habiendo supuesto el deterioro del programa

Descripción de la refactorización: A mayor complejidad de código mayor complejidad a la hora de mantenerlo, por lo que simplificar la complejidad del código facilitará los futuros mantenimientos, modificaciones y revisiones de éste

Autor: Igor

bookRide: Write short units of code

```java
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;
```

Resultado:

```java
public boolean puedoPagar(String username, Ride ride, int seats, double desk) {
    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;}return true;}
public boolean bookRide(String username, Ride ride, int seats, double desk) {
    boolean resultado;
    try {
        db.getTransaction().begin();

        Traveler traveler = getTraveler(username);
        double ridePriceDesk = (ride.getPrice() - desk) * seats;
        double availableBalance = traveler.getMoney();
        resultado = puedoPagar(username, ride, seats, desk);


        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 resultado;
```

Descripción de la refactorización: Para no saturar un método/función con código, creamos otro método/función para dividir los bloques de código en unidades menores, así su mantenimiento es más sencillo e incluso se podría pensar que se usa uno de los principios de SOLID (concretamente el de SRP, Single-Responsibility Principle)

Autor: Igor