IKER MAYORDOMO

"Write short units of code" && "Write simple units of code"

HASIERAKO KODEA:

```
public void priorityReservation(int ridenumber) {
                        Ride ride = db.find(Ride.class, ridenumber);
                        TypedQuery<Request> query = db.createQuery("SELECT r FROM Request r WHERE r.origin = :origin AND r.destination = :destination AND
r.rideDate = :date ORDER BY r.requestDate", Request.class);
                        query.setParameter("origin", ride.getFrom());
                        query.setParameter("destination", ride.getTo());
                        query.setParameter("date", ride.getDate());
                        int kant = ride.getnPlaces();
                        for (Request request : query.getResultList()) {
                                if (request.getKant() <= kant) {
                                        if(request.getValoration() <= ride.getDriver().getVal()) {
                                                if(request.isAutoBuy()) {
                                                        if((ride.getPrice()*request.getKant()) <= request.getPassenger().getMoney()) {
                                                                reservate(request.getPassenger().getEmail(), ridenumber,
request.getKant());
                                                                createNotification(ride, request.getPassenger().getEmail(), 'C');
                                                                deleteRequest(request.getRequestId());
                                                                kant -= request.getKant();
                                                        }else {
                                                                createNotification(ride, request.getPassenger().getEmail(), 'N');
                                                        }
                                                }
                                        }
                                }
                        }
                }
```

ERREFAKTORIZATUTAKO KODEA:

```
public void priorityReservation(int rideNumber) {
                Ride ride = db.find(Ride.class, rideNumber);
                List<Request> query = getMatchingRequests(ride);
                int availableSeats = ride.getnPlaces();
                for (Request request : query) {
                        if (!canReserve(request, ride, availableSeats)) {
                                continue;
                         }
                processReservation(request, ride, rideNumber);
                availableSeats -= request.getKant();
                }
}

public List<Request> getMatchingRequests(Ride ride) {
                TypedQuery<Request> query = db.createQuery(
                  "SELECT r FROM Request r WHERE r.origin = :origin AND r.destination = :destination AND r.rideDate = :date ORDER BY
r.requestDate",
                    Request.class
                );
                query.setParameter("origin", ride.getFrom());
                query.setParameter("destination", ride.getTo());
                query.setParameter("date", ride.getDate());
                return query.getResultList();
                }

                public boolean canReserve(Request request, Ride ride, int availableSeats) {
                  if (request.getKant() > availableSeats) return false;
                  if (request.getValoration() > ride.getDriver().getVal()) return false;
                  if (!request.isAutoBuy()) return false;
                  double totalCost = ride.getPrice() * request.getKant();
                  return totalCost <= request.getPassenger().getMoney();
                }

                public void processReservation(Request request, Ride ride, int rideNumber) {
                  reservate(request.getPassenger().getEmail(), rideNumber, request.getKant());
                  createNotification(ride, request.getPassenger().getEmail(), 'C');
                  deleteRequest(request.getRequestId());
                }
```

DESKRIBAPENA:

Metodo batekin 2 bad smell desberdin kenduko ditugu. Alde batetik kode honek 15 lerro baino gehiago zituen, eta hori ekiditzeko azpimetodotan banatu dugu. Bestetik, metodoaren konplexutasun ziklomatikoa 6ekoa zen; hortaz, murrizteko, baldintzen zatia ere banatu egin dugu.

Kode berriaren azalpena: lehenik, priorityreservation oinarrizko metodoa dugu, eta orain getMatchingRequest metodo bat jarri diogo hasieran datu baseko bidaia horren eskaerak lortzeko. Ondoren canReserve metodo bat gehitu diogu konprobatzeko ea posible den erreserba egitea lortutako erreserba listetatik, if asko segidan ez edukitzeko. Eta, azkenik, erreserba posiblea bada, processReservation azpimetodo bat, erreserbatzeko prozesu guztiak egiten dituena (notifikazio sortu, erreserba borratu, erreserbatu).

“Duplicate code”

HASIERAKO KODEA:

```
public void addMoney(String email, double money, String reason, Booking b) {
                db.getTransaction().begin();
                User us = db.find(User.class, email);
                us.setMoney(us.getMoney() + money);
                addTransaction(us, money, true, new Date(),reason, b);
                db.getTransaction().commit();
                if(us instanceof Driver) {
                        MainDriverGUI.setDriver((Driver)us);
                }else {
                        MainPassengerGUI.setPassenger((Passenger)us);
                }
        }
        public void removeMoney(String email, double money, String reason, Booking b) {
                db.getTransaction().begin();

                User us = db.find(User.class, email);
                if(us.getMoney() - money >= 0) {
                        us.setMoney(us.getMoney() - money);
                        addTransaction(us, money, false, new Date(),reason, b);
                }
                db.getTransaction().commit();
                if(us instanceof Driver) {
                        MainDriverGUI.setDriver((Driver)us);
                }else {
                        MainPassengerGUI.setPassenger((Passenger)us);
                }
        }
```

ERREFAKTORIZATUTAKO KODEA:

```
private void updateMoneyWrapper(String email, double money, String reason, Booking b, boolean isAdding) {
        db.getTransaction().begin();
        User us = db.find(User.class, email);
        if (us != null) {
           double finalAmount = isAdding ? us.getMoney() + money : us.getMoney() - money;
           us.setMoney(finalAmount);
           addTransaction(us, isAdding ? money : -money, isAdding, new Date(), reason, b);
        }
        db.getTransaction().commit();
        if(us instanceof Driver) {
                        MainDriverGUI.setDriver((Driver)us);
                }else {
```

```
                    MainPassengerGUI.setPassenger((Passenger)us);
                }
        }
        public void addMoney(String email, double money, String reason, Booking b) {
            updateMoneyWrapper(email, money, reason, b, true);
        }
        public void removeMoney(String email, double money, String reason, Booking b) {
            updateMoneyWrapper(email, money, reason, b, false);
        }
```

DESKRIBAPENA:

Kodearen errepikapena sahiesteko, addMoney eta removeMoney motodoak metodo berdin batean batu ditugu Wrapper modeloarekin, eta este klaseko kodea ez aldatzen egoteko aurreko metodoak mantentzen ditugu batura honi deitzeko.
Gainera, aprobetxatu dugu if-ak operatzaile ternario batzuengatik aldatzeko geroz eta kode gehiago sahiesteko.

"Keep unit interfaces small"

HASIERAKO KODEA:

```
public User createUser(String email, String username, String password, double money, boolean type) {
        User us;
        if(type) {
                us = new Driver(email, username, password, money);
        }else {
                us = new Passenger(email, username, password, money);
        }
        if (! isDriver(us.getEmail()) && ! isPassenger(us.getEmail())) {
                db.getTransaction().begin();
                db.persist(us);
                db.getTransaction().commit();
                return us;
        }
        return null;
}
```

ERREFAKTORIZATUTAKO KODEA:

```
private class UserData {
    private String email;
    private String username;
    private String password;
    private double money;
    private boolean isDriver;
    public UserData(String email, String username, String password, double money, boolean isDriver) {
        this.email = email;
        this.username = username;
        this.password = password;
        this.money = money;
        this.isDriver = isDriver;
    }
    public String getEmail() { return email; }
    public String getUsername() { return username; }
    public String getPassword() { return password; }
    public double getMoney() { return money; }
    public boolean isDriver() { return isDriver; }
}

public User createUser(String email, String username, String password, double money, boolean type) {
    UserData data = new UserData(email, username, password, money, type);
    return createUser(data);
}

/**
```

```
 * This method enters a new user if it is not there
 */
public User createUser(UserData data) {
   User us = data.isDriver()
      ? new Driver(data.getEmail(), data.getUsername(), data.getPassword(), data.getMoney())
      : new Passenger(data.getEmail(), data.getUsername(), data.getPassword(), data.getMoney());
   if (!isDriver(us.getEmail()) && !isPassenger(us.getEmail())) {
      db.getTransaction().begin();
      db.persist(us);
      db.getTransaction().commit();
      return us;
   }
   return null;
}
```

DESKRIBAPENA:

5 parametro genituenez createUser funtzioan, DataAcess klasean azpiklase pribatu bat sortu dugu Useraren datuak gordeko dituena, eta metodo berriak UserData objetu berri hau jasoko duena. Hala ere, kodea mantentzeko eta aldaketa asko ekiditzeko, aurreko metodoa utzi dugu objetu berri hau sortzen duena eta metodo berriari deitzen diona, baina berez aldaketa gehiago egin beharko lirateke beste kodean.

AIMAR ESPARZA

"Write short units of code"

HASIERAKO KODEA:

```
public List<Valoration> getValorations(String email){
	List<Valoration> valList = new ArrayList<Valoration>();
	if (isDriver(email)) {
		Driver d = db.find(Driver.class, email);
		TypedQuery<Ride> query = db.createQuery("SELECT r FROM Ride r where r.dirver = :driver", Ride.class);
		query.setParameter("driver", d);
		for(Ride r : query.getResultList()) {
			for (Booking b : r.getBookings()) {
				if(b.getDriverValoration() != null) {
					valList.add(b.getDriverValoration());
				}
			}
		}
		return valList;
	}else {
		Passenger p = db.find(Passenger.class, email);
		TypedQuery<Booking>query=db.createQuery("SELECT r FROM Booking r where r.passenger=:passenger", Booking.class);
		query.setParameter("passenger", p);
		for(Booking b : query.getResultList()) {
			if(b.getDriverValoration() != null) {
				valList.add(b.getDriverValoration());
			}
		}
		return valList;
	}
}
```

ERREFAKTORIZATUTAKO KODEA:

```
public List<Valoration> getValorations(String email){
	List<Valoration> valList = new ArrayList<Valoration>();
	if (isDriver(email)) {
		return driverGetValorations(email, valList);
	}else {
		return passengerGetValorations(email, valList);
	}
}
public List<Valoration> driverGetValorations(String email, List<Valoration> valList){
	Driver d = db.find(Driver.class, email);
	TypedQuery<Ride> query = db.createQuery("SELECT r FROM Ride r where r.dirver = :driver", Ride.class);
	query.setParameter("driver", d);
	for(Ride r : query.getResultList()) {
		for (Booking b : r.getBookings()) {
			if(b.getDriverValoration() != null) {
				valList.add(b.getDriverValoration());
			}
		}
	}
	return valList;
}
public List<Valoration> passengerGetValorations(String email, List<Valoration> valList){
	Passenger p = db.find(Passenger.class, email);
	TypedQuery<Booking> query = db.createQuery("SELECT r FROM Booking r where r.passenger = :passenger", Booking.class);
	query.setParameter("passenger", p);
	for(Booking b : query.getResultList()) {
		if(b.getDriverValoration() != null) {
			valList.add(b.getDriverValoration());
		}
	}
	return valList;
}
```

DESKRIBAPENA:

Hasierako kodeak 24 lerro zituen, eta beraz hainbat funtzio txikiagotan banatu dut kodea, horrela kodea ulergarriagoa eginez.

"Write simple units of code"

HASIERAKO KODEA:

```
public User login(String email, String password) {

		User us = null;
		Driver dr = db.find(Driver.class, email);
		Passenger pa = db.find(Passenger.class, email);
		Admin ad = db.find(Admin.class, email);
		if (pa != null) {
			if (pa.getPassword().equals(password)) {
				us = pa;
			}
		}else if (dr != null){
			if (dr.getPassword().equals(password)) {
					us = dr;
			}
		}else if(ad != null) {
			if(ad.getPassword().equals(password)) {
				us = ad;
			}
		}
		return us;
}
```

ERREFAKTORIZATUTAKO KODEA:

```
public User login(String email, String password) {
		User us = null;
		Driver dr = db.find(Driver.class, email);
		Passenger pa = db.find(Passenger.class, email);
		Admin ad = db.find(Admin.class, email);
		if (pa != null && pa.getPassword().equals(password)) {
				us = pa;
		}else if (dr != null && dr.getPassword().equals(password)){
			us = dr;
		}else if(ad != null && ad.getPassword().equals(password)) {
			us = ad;
		}
		return us;
}
```

DESKRIBAPENA:

Funzioak hainbat kasutan if bat bestearen barruan zeukan, honi komplexutasuna murrizteko bi if horiek bakar batean ipini ditut, horrela, V(G)=7 tik V(G)=4 ra pasa da.

“Duplicate code”

HASIERAKO KODEA:

FindRidesGUI klasean:

```
tableRides.getColumnModel().getColumn(0).setPreferredWidth(170);
tableRides.getColumnModel().getColumn(1).setPreferredWidth(30);
tableRides.getColumnModel().getColumn(1).setPreferredWidth(30);
```

```
tableRides.getColumnModel().removeColumn(tableRides.getColumnModel().getColumn(3));
```

ERREFAKTORIZATUTAKO KODEA:

```
public void tableRidesConf() {
	tableRides.getColumnModel().getColumn(0).setPreferredWidth(170);
	tableRides.getColumnModel().getColumn(1).setPreferredWidth(30);
	tableRides.getColumnModel().getColumn(1).setPreferredWidth(30);
	tableRides.getColumnModel().removeColumn(tableRides.getColumnModel().getColumn(3));
}
```

DESKRIBAPENA:

Hasierako kodea hiru aldiz zegoen errepikatuta, klase honetan, horregatik kode errepikapena ekiditzeko tableRidesConf() funtzioa sortu dut, horrela mantenimendua hobetzeko.

"Keep unit interfaces small"

HASIERAKO KODEA:

```
public void addValoration(int bookingId, boolean done, int val, String com, String email) {
		if(isDriver(email)){
			addDriverValoration(bookingId, done, val, com);
		}else if ( isPassenger(email)) {
			addPassengerValoration(bookingId, done, val, com);
		}
}
```

ERREFAKTORIZATUTAKO KODEA:

```
private class ValorationPack {
		private int bookingId;
		private boolean done;
		private int val;
		private String com;
		private String email;

		public ValorationPack (int bookingId, boolean done, int val, String com, String email) {
			this.bookingId = bookingId;
			this.done = done;
			this.val = val;
			this.com = com;
			this.email = email;
		}
		public int getBookingId() {return bookingId;}
		public boolean getDone() {return done;}
		public int getVal() {return val;}
		public String getCom() {return com;}
		public String getEmail() {return email;}
	}
	public void addValoration(int bookingId, boolean done, int val, String com, String email) {
		addValoration(new ValorationPack(bookingId, done, val, com, email));
	}

	public void addValoration(ValorationPack pack) {
		if(isDriver(pack.getEmail())){
			addDriverValoration(pack.getBookingId(), pack.getDone(), pack.getVal(), pack.getCom());
		}else if ( isPassenger(pack.getEmail())) {
			addPassengerValoration(pack.getBookingId(), pack.getDone(), pack.getVal(), pack.getCom());
		}
	}
```

DESKRIBAPENA:

5 parametro genituenez addValoration funtzioan, DataAcess klasean azpiklase pribatu bat sortu dugu Balorazioentzarako behar diren datuak gordeko dituena, Hala ere, kodea mantentzeko eta aldaketa asko ekiditzeko, aurreko metodoa utzi dugu objetu berri hau sortzen duena eta metodo berriari deitzen diona, baina berez aldaketa gehiago egin beharko lirateke beste kodean.