PROIEKTUA 1.1

9.taldea, Ibai Oñatibia, Iker Galarraga eta Jon Olea

Proiektuaren sonnar link-a: https://sonarcloud.io/project/overview?id=9-taldea-rides24
Proiektuaren github link-a: https://github.com/IkerGI28/Rides24E9

Sonar erroreen arazoen konponketa dokumentatua: sonar.pdf (ere proiektuaren erroan)

Aukeratu ditugun metodoak:

- bookRide
- gauzatuEragiketa
- getRidesbyDriver

bookRide

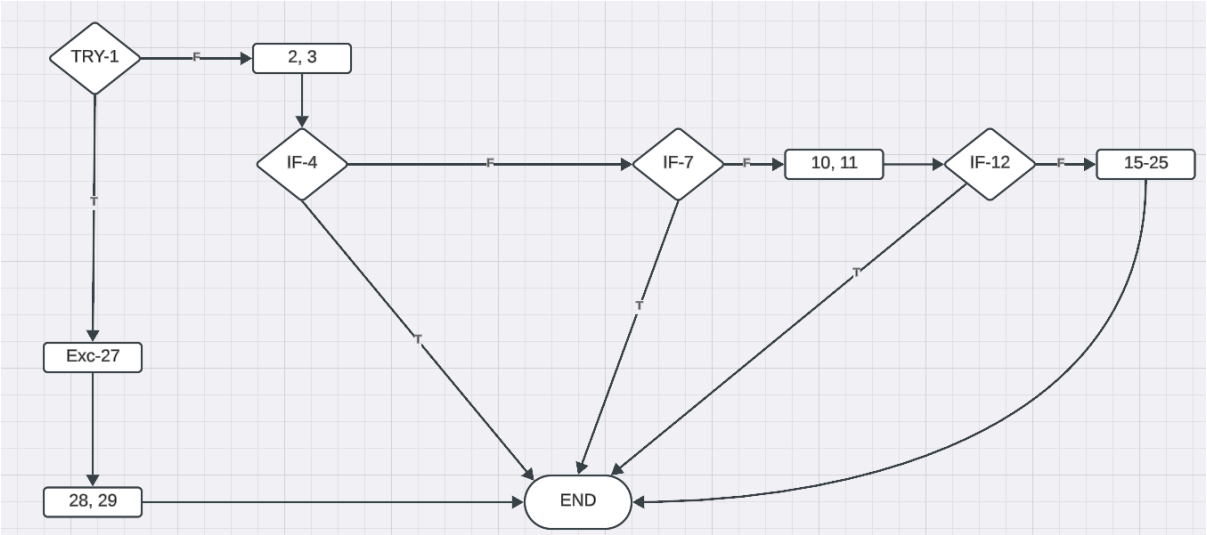
egilea = Iker Galarraga

bookRide metodoak bidai baten gainean erreserba bat egitea du helburu. Sartu beharreko parametroak biaiariaren username, bidaia, eserleku kopurua eta deskontuaren kopurua dira. Metodoak boolean balio bat bueltatzen du. TRUE bueltatuko du bidaiaria datu basean badago, eskatutako eserleku kopurua eskuragarri badaude eta bidaiariak bidaia ordaindu ahal badu eta FALSE bueltatuko salbuespen bat altxatuz gero edo aurretik esandako baldintzaren bat betetzen ez bada. Hona hemen metodoaren inplementazioa:

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

Kutxa txuriaren analisia

Grafoa



Konplexutasun ziklomatikoa = 4 + 1 = **5**
Kutxa txuriaren analisiaren taula

	Bidea	Baldintza	Sarrera		Itxaron Emaitza		
			Parametroak	**DB egoera**	**Irteera**	**DB egoera**	
1	TRY-1(T) Exc-27 28 29 END	*	username=null, ride=r, seats=2.45, desk=i	t !∈ DB	FALSE	Aldaketa gabe	
2	TRY-1(F) 2 3 IF-4(T) END	t !∈ DB	username=Pepe, ride=r,	t !∈ DB	FALSE	Aldaketa gabe	t=Traveler("Patxi")

			seats=2, desk=3				r=Ride("Donostia", "Zarautz")
3	TRY-1(F) 2 3 IF-4(F) IF-7(T) END	t ∈ DB && ride.getNPlaces()<seats	username=Patxi, ride=r, seats=2, desk=3	t ∈ DB	FALSE	Aldaketa gabe	t=Traveler("Patxi") r=Ride("Donostia", "Zarautz", 05/10/2026)
4	TRY-1(F) 2 3 IF-4(F) IF-7(F) 10 11 IF-12(T) END	t ∈ DB && ride.getNPlaces()>=seats && availableBalance<ridePriceDesk	username=Patxi, ride=r, seats=2, desk=3	t ∈ DB	FALSE	Aldaketa gabe	t=Traveler("Patxi") r=Ride("Bilbo", "Mutriku", 05/10/2026)
5	TRY-1(F) 2 3 IF-4(F) IF-7(F) 10 11 IF-12(F) 15..25 END	t ∈ DB && ride.getNPlaces()>=seats && availableBalance>=ridePriceDesk	username=Patxi, ride=r, seats=2, desk=3	t ∈ DB	TRUE	t ∈ DB && b ∈ t	t=Traveler("Patxi") r=Ride("Ordizia", "Anoeta", 05/10/2200)

Kutxa beltzaren analisia

Baldintza	Baliok. klase egokiak	Baliok. klase ez egokiak
username balioa	username != null (1)	username == null (9)
ride balioa	ride != null (2)	ride == null (10)
seats balioa	seats > 0 (3)	seats <= 0 (11)
desk balioa	desk >= 0 (4)	desk < 0 (12)
tokia dago	r.nPlaces >= seats (5)	r.nPlaces < seats (13)
ordaindu dezake	t.money >= (r.price - desk) * seats (6)	t.money < (r.price - desk) * seats (14)
traveler DB-an	traveler ∈ DB (7)	

traveler DB-an ez	traveler !∈ DB (8)	

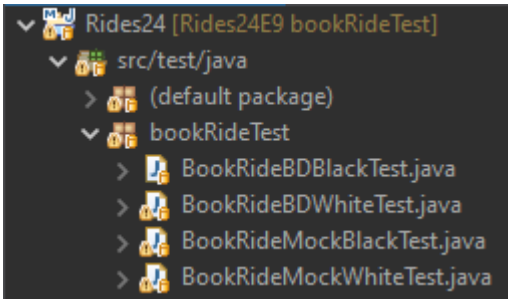
Baliokidetasun-klaseen taula

Proba kasuen taula

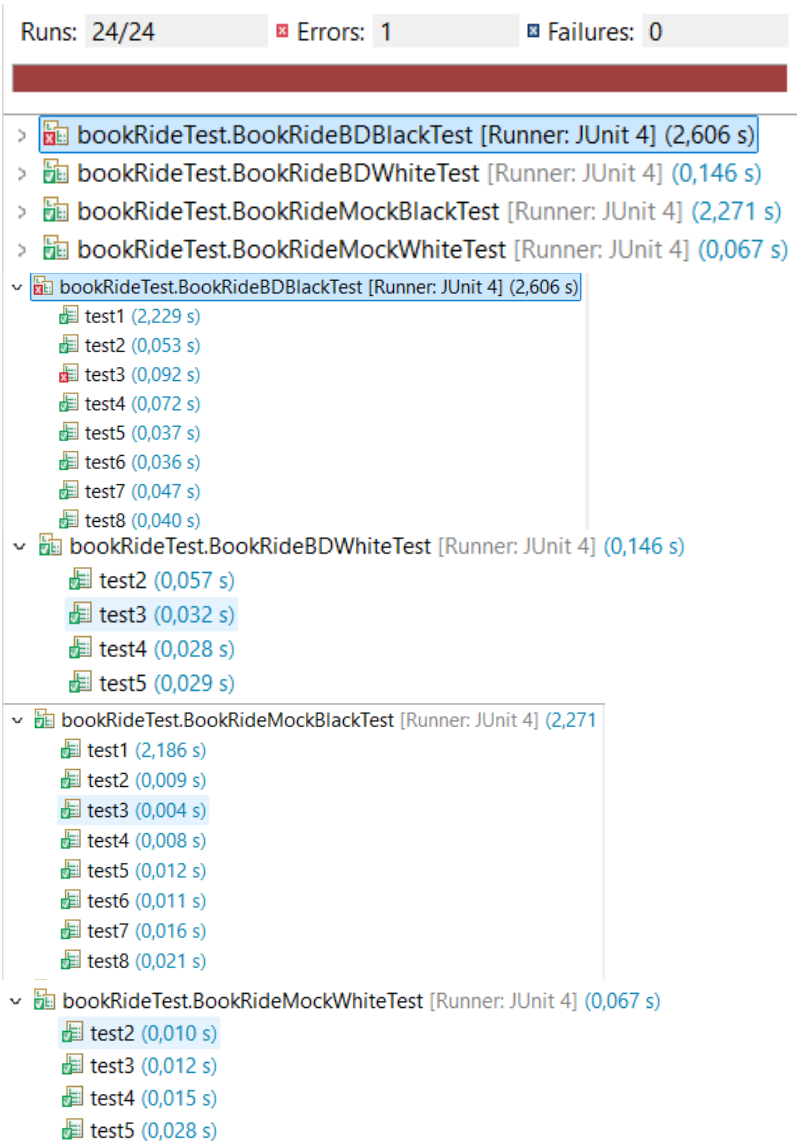
#	Estalitako klaseak	Sarrera kontestua		Irteera kontestua		
		DB Egoera	**Parametroak (username, ride, seats, desk)**	**DB Egoera**	**Irteera**	*: zutabe honetan agertzen diren parametroak BDBlackTest klaseko parametroak dira, WhiteTesteko parametroak desberdinak dira
1	1, 2, 3, 4, 5, 6, 7	t ∈ DB	username= Patxi, ride=r, seats=2, desk=3	t ∈ DB && b ∈ t	TRUE	r=Ride("Ordizia", "Anoeta", 05-10-2200, 3, 10)
2	1, 2, 3, 4, 5, 6, 8	t !∈ DB	username= Patxi, ride=r, seats=2, desk=3	t !∈ DB	FALSE	r=Ride("Milan", "Roma", 23-05-2028, 5, 6)
3	9	*	username= null, ride=r, seats=2, desk=3	Ez da aldatzen	FALSE	r=null
4	10	*	username= Patxi, ride=null, seats=2, desk=3	Ez da aldatzen	FALSE	r=null
5	11	*	username= Patxi, ride=r, seats=0, desk=3	Ez da aldatzen	FALSE	r=("Donostia", "Zarautz", 05-10-2026, 5, 6)

6	12	*	username=Patxi, ride=r, seats=2, desk=-3	Ez da aldatzen	FALSE	r=("Donostia", "Zarautz", 05-10-2026, 5, 6)
7	13	*	username=Patxi, ride=r, seats=6, desk=3	Ez da aldatzen	FALSE	r=("Donostia", "Zarautz", 05-10-2026, 5, 6)
8	14	*	username=Patxi, ride=r, seats=2, desk=1	Ez da aldatzen	FALSE	r=("Donostia", "Zarautz", 05-10-2026, 5, 6)

Testak src/test/java/BookRideTest paketean daude



Testen emaitzak (errore bat)



Metodoaren estalpena 100%

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

Aurkitutako akatsak eta izandako arazoak

Akats batzukt aurkitu ditut bookride metodoaren inplementazioan. Erreserba bat eskatzerakoan 0 eserlekuekin metodoak false bueltatu beharko luke, ez daukalako zentzurik 0 eserlekuko erreserba bat egitea, hala ere true bueltatzen du metodoak eta -1 eserleku kopuruko erreserba egitean true bueltatu dit bookRide metodoak behin baino gehiagotan. Gainera, deskontua negatiboa denean erreserba egin daiteke, hau da ez da kontrolik egiten ea deskontuaren balioa negatiboa den, zuzenean erreserba egiten da ordaintzeko dirua badaukazu eta datu basean gordetako erabiltzailea bazara.

Horretaz gain hainbat arazo izan ditut testak egokiak izaterako garaian. DataAccesseko hainbat metodoetan patroi berdina jarraitu dutelako, return egiterakoan metodo batzuetan ez dute commit egiten eta beraz, trantsakzioa oraindik aktiboa mantentzen zen.

JUnit proba unitarioak egiterakoan ondo inplementatu ditut proba kasu gehieneak baina ez nekien nola inplementatu try-catch blokeko TRY-1(T) kasua.

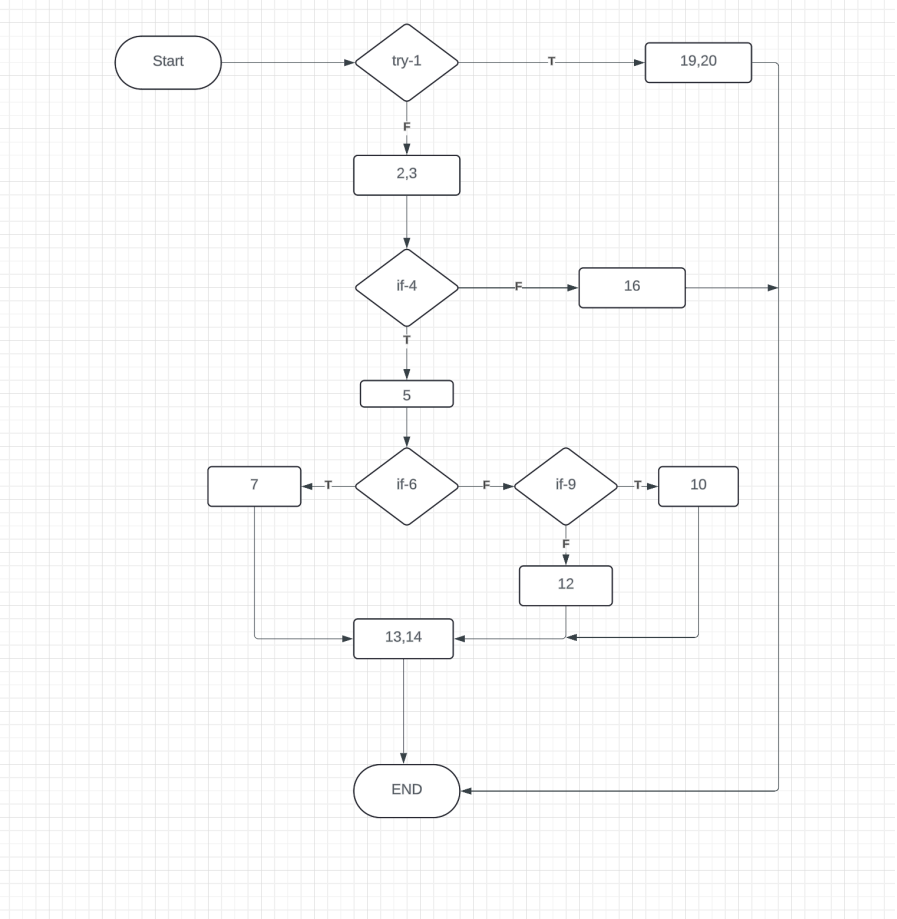
gauzatuEragiketa

egilea = Ibai Oñatibia

Erabiltzailearen kontuan dirua sartzeko edo ateratzeko erabiltzen den metodoa da, 3 atributo jasotzen ditu, erabiltzailea, sartu edo atera nahi den diru kopurua eta boolean atributu bat, true denean dirua sartzen da eta false denean dirua ateratzen da, metodoak boolean bat itzultzen du transferentzia ondo egin den ala ez adierazten duena.

```
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;
    }
}
```

Kutxa Txuria



Konplexutasun ziklomatrikoa: 4+1 =5

Proba kasuen taula:

u =gorka@gmail.com,amount=20

	Bidea	Baldintza	Sarrera		Itxaron Emaitza	
			Parametroak	**DB egoera**	**Irteera**	**DB egoera**
1	TRY-1(T),19,20,END	Errore bat egotea.	null, amount,TRUE	-	FALSE	Berdina
2	TRY-1(F), 2,3,IF-4(F),16, END	u∉ DB	u, amount , TRUE	u∉ DB	FALSE	Berdina
3	TRY-1(F), 2,3,IF-4(T),5,IF-6(T),13,14,END	u∈DB eta dirua kontuan depositatu egiten da	u, amount , TRUE	u∈DB	TRUE	Money = currentMoney+amount
4	TRY-1(F), 2,3,IF-4(T),5,IF-6(F),IF-9(T),10,13,14 ,END	u∈DB eta daukana bainan diru gehiago kontutik atera nahi du	u,40,FALSE	u∈DB eta currentMoney=30	TRUE	Money = 0
5	TRY-1(F), 2,3,IF-4(T),5,IF-6(F),IF-9(F),12,13,14 ,END	u∈DB eta daukan dirua atera nahi du	u,amount,FALSE	u∈DB eta currentMoney =30	TRUE	Money = currentMoney-amount

Kutxa Beltza:

Baliokidetasun-klasearen emaitza-taula:

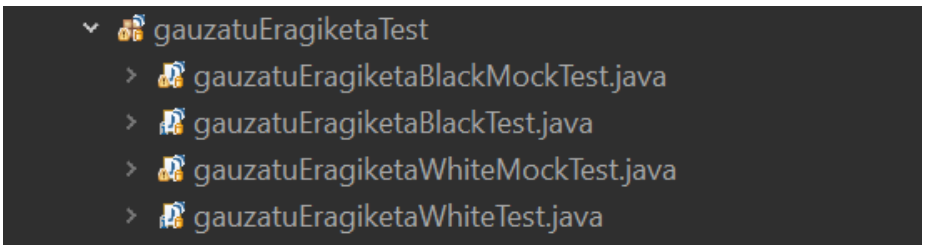
Baldintza	Baliok. klase egokiak	Baliok. klase ez egokiak
Username null?	Username ez da null(1)	Username==null (7)
amount negatiboa?	amount ez da negatiboa(2)	amount<0(8)
username datu basean dago	u∈DB (3)	
username-a ez dago datu basean	u∉ DB (4)	
atera nahi den dirua daukana baina handiagoa da	amount>currentMoney (5)	
atera nahi den dirua daukana baina gutxiago da	amount<=currentMoney(6)	

Proba kasuak

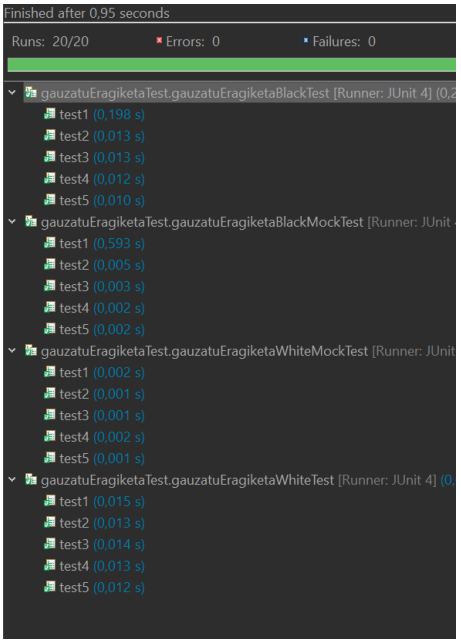
#	Estalitako klaseak	Sarrera kontestua		Irteera kontestua	
		DB Egoera	**Parametroak**	**DB Egoera**	**Irteera**
1	1,2,3,5	"Iker" datubasean dago	Iker,30,FALSE	money=0	TRUE
2	1,2,3,6	"Iker" datubasean dago	Iker,40,FALSE	money>=0	TRUE
3	7	Edozein egoera	null,10,TRUE	Aldaketarik gabe	FALSE
4	1,3,8,	"Iker" datubasean dago, currentMoney =20	Iker,-10,TRUE	10	FALSE
5	1,2,4,	"Iker" ez dago datubasean	Iker,20,TRUE	Aldaketarik gabe	FALSE

Proben Inplementazioa

src/test/java/gauzatuEragiketaTest -ean daude



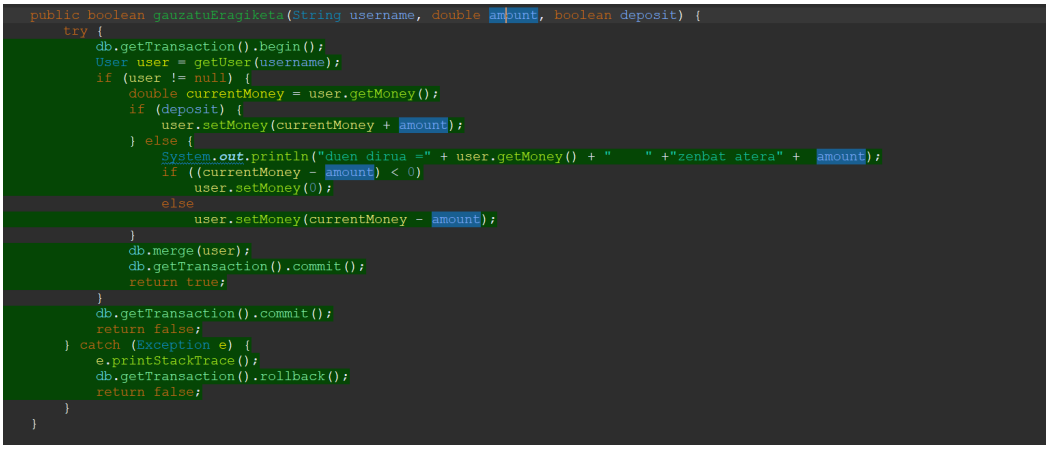
testen emaitzak:



Hasierako metodoak ez zuen estalduraren %100-a estaltzen egoera batzuetara ez zelako iristen, hau lortzeko metodoak erabiltzen duen getUser metodoa aldatu dugu, erabiltzailea null denean NullPointerException bat jaurtitzeko eta datubasean ez duenean biltatzen null itzultzeko:

```
public User getUser(String erab){
    if(erab == null) throw new NullPointerException();
    try {
        TypedQuery<User> query = db.createQuery("SELECT u FROM User u WHERE u.username = :username", User.class);
        query.setParameter("username", erab);
        return query.getSingleResult();
    } catch (Exception e) {
        return null;
    }
}
```

Aldaketa hau eginda estalduraren %100-a lortu dugu:



```
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 {
                System.out.println("duen dirua =" + user.getMoney() + "    " +"zenbat atera" +  amount);
                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;
    }
}
```

Aurkitutako akatsak:

Kodigoaren implementazioa ez zegoen guztiz ondo eginda, iristezina diren egoerak zeudelako, dena getUser metodoaren implementazio okerraren arazoa zen, ez zuelako ondo lan egiten null balioarekin eta datubasean ez dauden elementuekin

Baita ere gomendagarria iruditko litzateke, atera nahi den diru kopurua erabiltzaileak daukana baina handiagoa bada, errore bat ematea edo sortutako salbuespena bat jaurtitzea, oraingo kodigoarekin 0 balioan uzten diolako.

Azkenik, zenbaki negatiboko transakziak egiten uzten du, nire ustez aldatu beharko litzatekeen zerbait da, ez du zentzurik -10 euro sartzen uztera, dirua ateratzen arituko zinelako.

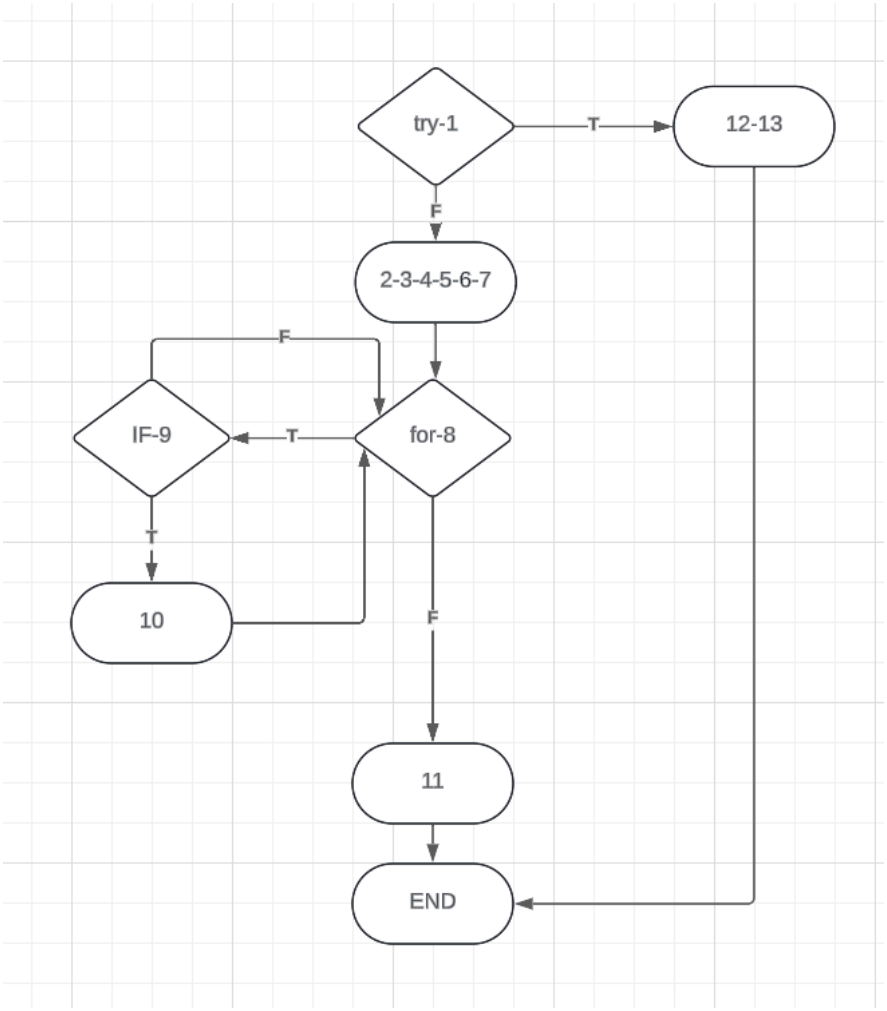
getRidesByDriver

egilea = Jon Olea

Metodo honek gidari batek sortutako eta momentuan aktibo dauden bidaien lista bat itzultzen du.
Parametro bezala, gidariaren kontu izena sartzen da.
Metodoak gidariak sortutako lista itzultzen du. Ez bada bidairik aurkitzen, lista hutsa itzuliko du, eta errore bat gertatu bada, null itzuliko du.

```java
/**
 * This method returns the rides that a driver has created and are active
 * @param username The username of the driver whose rides are to be returned
 * @return A list of active rides created by the driver. If no active rides exist or if an error occurs, returns null.
 */
public List<Ride> getRidesByDriver(String username) {
    try {
        TypedQuery<Driver> query = db.createQuery("SELECT d FROM Driver d WHERE d.username = :username",
                Driver.class);
        query.setParameter("username", username);
        Driver driver = query.getSingleResult();
        List<Ride> rides = driver.getCreatedRides();
        List<Ride> activeRides = new ArrayList<>();
        for (Ride ride : rides) {
            if (ride.isActive()) {
                activeRides.add(ride);
            }
        }
        return activeRides;
    } catch (Exception e) {
        return null;
    }
}
```

Kutxa Txuria



Konplexutasun ziklomatikoa = 3 + 1 = **4**

Proba kasuen taula

	Bidea	Baldintza	Sarrera		Itxaron Emaitza	
			Parametroak	**DB egoera**	**Irteera**	**DB egoera**
1	TRY1(T)-12-13	U !∈ DB	(User1)	User1 !∈ DB	null	Berdina
2	TRY1(F)-2-3-4-5-6-FOR7(F)-10	U -k ez du bidai-rik sortu	(User2)	U-k ez du bidairik erregistratua	activeRides (Bidai gabeko lista)	Berdina
3	TRY1(F)-2-3-4-5-6-FOR7(T,F)-IF8(F)-END	U-k ez du bidai aktiborik	(User3)	U-k ez du bidai aktiborik	activeRides (Bidai gabeko lista)	Berdina
4	TRY1(F)-2-3-4-5-6-FOR7(T,F)-IF8(T)-9-END	U-k bidai aktiboak ditu	(User4)	U-k bidai aktiboak ditu	activeRides (Bidai aktiboz osatutako lista)	Berdina

Kutxa Beltza

Baliokidetasun-klasearen emaitza-taula

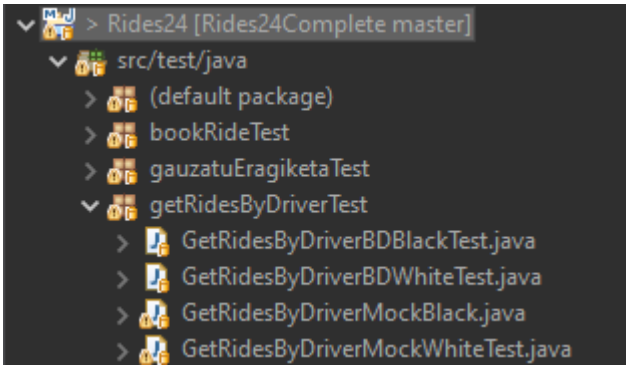
Baldintza	Baliok. klase egokiak	Baliok. klase ez egokiak
Username null?	Username ez null (1)	Username==null (6)
DB-an dago	Username ∈ DB (2)	
DB-an ez dago	Username !∈ DB (3)	
User-ek bidaiak ditu DB-an	User.rides != [] (4)	
User-ek ez ditu bidaiak DB-an	User.rides == [] (5)	

Proba kasuak

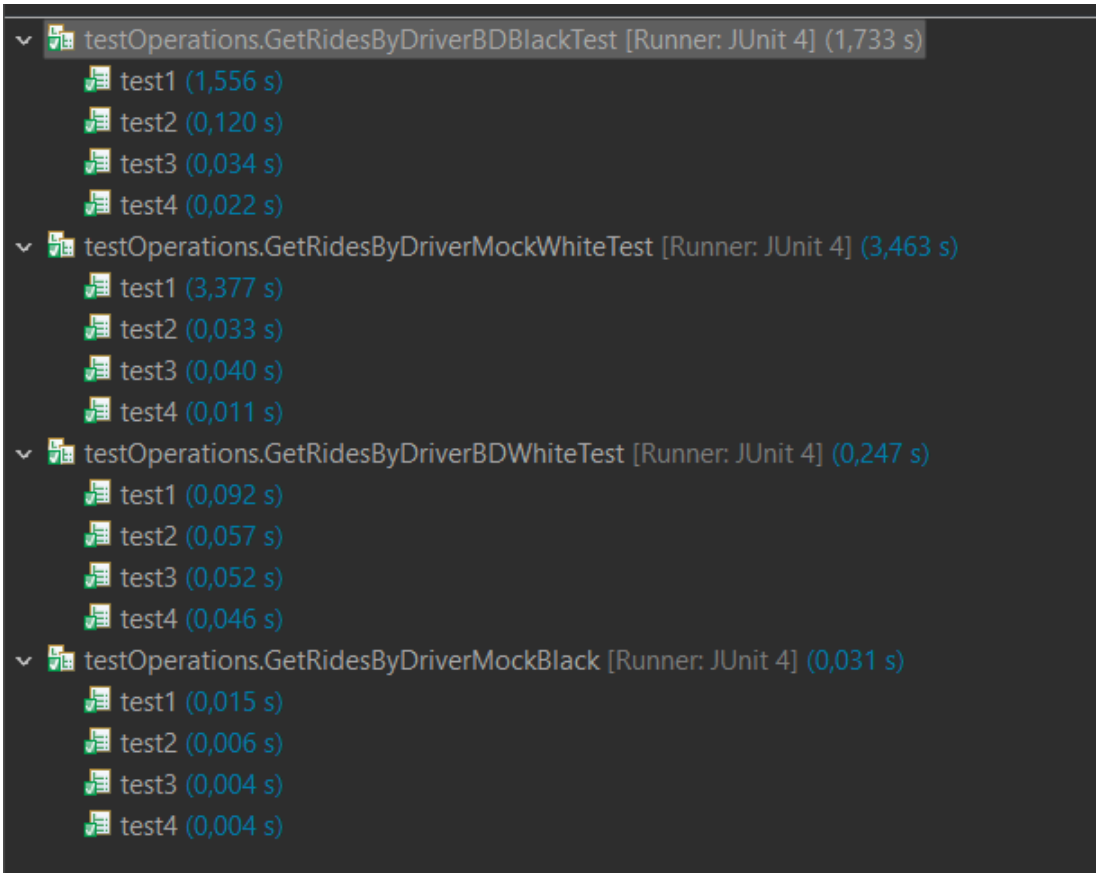
#	Estalitako klaseak	Sarrera kontestua		Irteera kontestua	
		DB Egoera	**Parametroa (username)**	**DB Egoera**	**Irteera**
1	1,3	driverTest ez dago DB-an	driverTest	Aldaketa gabe	null
2	1, 2, 4,	driverTest DB-an, bidaiak ditu	driverTest	Aldaketa gabe	rideLista
3	1, 2, 5,	driverTest DB-an, bidai gabe	driverTest	Aldaketa gabe	rideLista
4	6	Edozein	null	Aldaketa gabe	null

Proben Inplementazioa

Test-ak src/test/java/testOperations- en daude



Test guztiak zuzen



%100-eko estaldura du metodoak

```java
public List<Ride> getRidesByDriver(String username) {
    try {
        TypedQuery<Driver> query = db.createQuery("SELECT d FROM Driver d WHERE d.username = :username",
                Driver.class);
        query.setParameter("username", username);
        Driver driver = query.getSingleResult();
        List<Ride> rides = driver.getCreatedRides();
        List<Ride> activeRides = new ArrayList<>();
        for (Ride ride : rides) {
            if (ride.isActive()) {
                activeRides.add(ride);
            }
        }
        return activeRides;
    } catch (Exception e) {
        return null;
    }
}
```

Aurkitutako akatsak

Ez dut akatsik aztertutako metodoan test-ak egitean. Hasieran akatsak lortzen nituen, baina test-ak gaizki zeudelako, ez dataAcces-eko metodotik eratorriak, beraz, esan daiteke metodoa ondo inplementatuta dagoela proba kasu guztiak begiratu ondoren. Arazoa kontsideratu daitekeen bakarra, errore bat gertatzen denean edo gidaria datu basean ez dagoenean metodoak portaera bera duela da, agian, kasu bakoitzean errore ezberdin bat altxatu lezake, arazoa zein den argiago ikusteko.