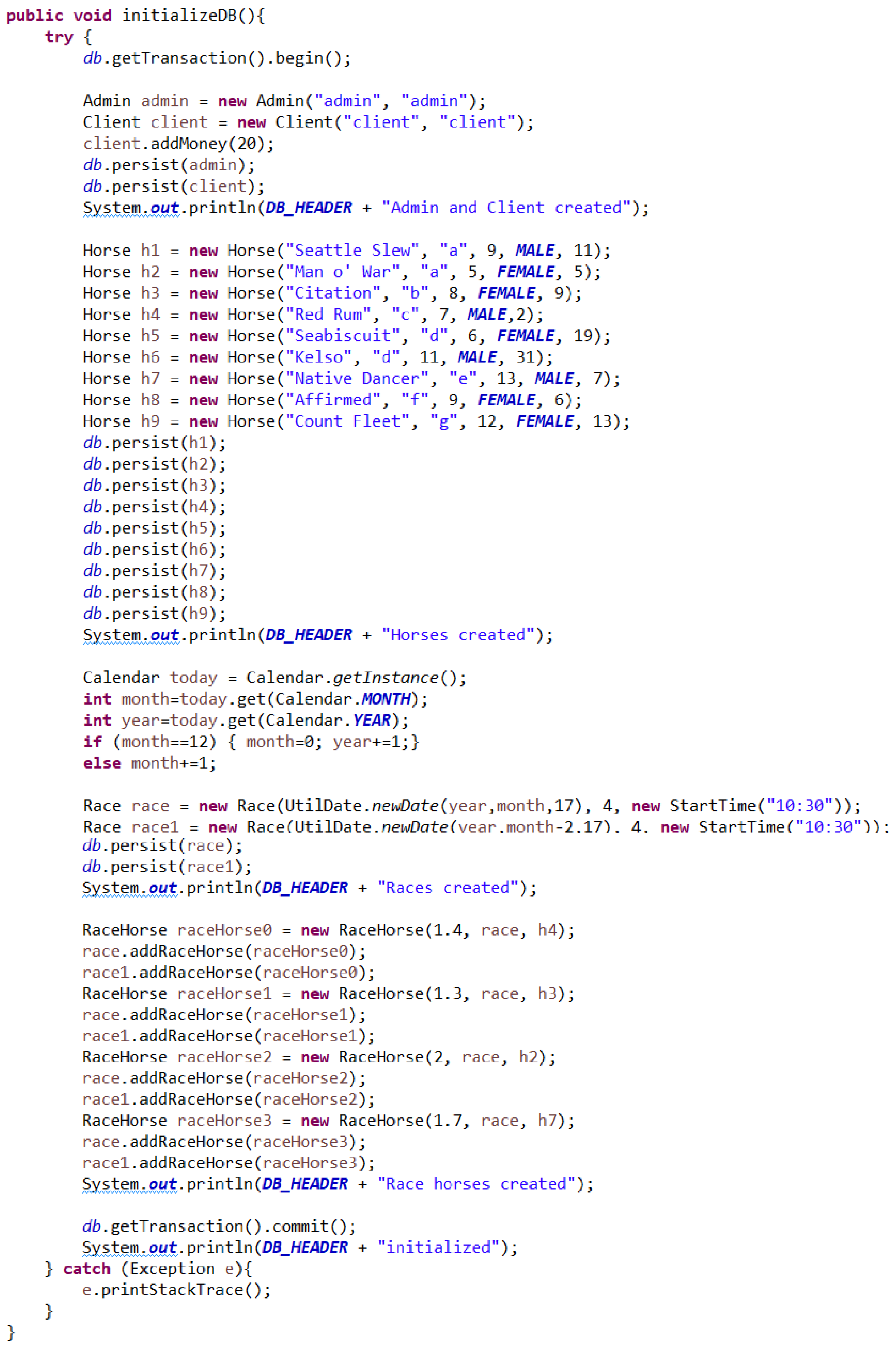
ERREFAKTORIZAZIOA(Julen Larrañaga) GITHUB

“Write short units of code”(15 lerroko metodoak gehienez)

1. **Hasierako kodea:**



```java
public void initializeDB(){
    try {
        db.getTransaction().begin();

        Admin admin = new Admin("admin", "admin");
        Client client = new Client("client", "client");
        client.addMoney(20);
        db.persist(admin);
        db.persist(client);
        System.out.println(DB_HEADER + "Admin and Client created");

        Horse h1 = new Horse("Seattle Slew", "a", 9, MALE, 11);
        Horse h2 = new Horse("Man o' War", "a", 5, FEMALE, 5);
        Horse h3 = new Horse("Citation", "b", 8, FEMALE, 9);
        Horse h4 = new Horse("Red Rum", "c", 7, MALE,2);
        Horse h5 = new Horse("Seabiscuit", "d", 6, FEMALE, 19);
        Horse h6 = new Horse("Kelso", "d", 11, MALE, 31);
        Horse h7 = new Horse("Native Dancer", "e", 13, MALE, 7);
        Horse h8 = new Horse("Affirmed", "f", 9, FEMALE, 6);
        Horse h9 = new Horse("Count Fleet", "g", 12, FEMALE, 13);
        db.persist(h1);
        db.persist(h2);
        db.persist(h3);
        db.persist(h4);
        db.persist(h5);
        db.persist(h6);
        db.persist(h7);
        db.persist(h8);
        db.persist(h9);
        System.out.println(DB_HEADER + "Horses created");

        Calendar today = Calendar.getInstance();
        int month=today.get(Calendar.MONTH);
        int year=today.get(Calendar.YEAR);
        if (month==12) { month=0; year+=1;}
        else month+=1;

        Race race = new Race(UtilDate.newDate(year,month,17), 4, new StartTime("10:30"));
        Race race1 = new Race(UtilDate.newDate(year,month-2,17), 4, new StartTime("10:30"));
        db.persist(race);
        db.persist(race1);
        System.out.println(DB_HEADER + "Races created");

        RaceHorse raceHorse0 = new RaceHorse(1.4, race, h4);
        race.addRaceHorse(raceHorse0);
        race1.addRaceHorse(raceHorse0);
        RaceHorse raceHorse1 = new RaceHorse(1.3, race, h3);
        race.addRaceHorse(raceHorse1);
        race1.addRaceHorse(raceHorse1);
        RaceHorse raceHorse2 = new RaceHorse(2, race, h2);
        race.addRaceHorse(raceHorse2);
        race1.addRaceHorse(raceHorse2);
        RaceHorse raceHorse3 = new RaceHorse(1.7, race, h7);
        race.addRaceHorse(raceHorse3);
        race1.addRaceHorse(raceHorse3);
        System.out.println(DB_HEADER + "Race horses created");

        db.getTransaction().commit();
        System.out.println(DB_HEADER + "initialized");
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

2. Errefaktorizatutako kodea:

```java
public void initializeDB(){
    initializeUsers();
    ArrayList<Horse> horses = initializeHorses();
    ArrayList<Race> races = initializeRaces();
    initializeRaceHorses(horses, races);
    System.out.println(DB_HEADER + "initialized");
}

private void initializeRaceHorses(ArrayList<Horse> horses, ArrayList<Race> races) {
    Race race = races.get(0);
    Race race1 = races.get(1);

    RaceHorse raceHorse0 = new RaceHorse(1.4, race, horses.get(4));
    race.addRaceHorse(raceHorse0);
    race1.addRaceHorse(raceHorse0);
    RaceHorse raceHorse1 = new RaceHorse(1.3, race, horses.get(3));
    race.addRaceHorse(raceHorse1);
    race1.addRaceHorse(raceHorse1);
    RaceHorse raceHorse2 = new RaceHorse(2, race, horses.get(2));
    race.addRaceHorse(raceHorse2);
    race1.addRaceHorse(raceHorse2);
    RaceHorse raceHorse3 = new RaceHorse(1.7, race, horses.get(7));
    race.addRaceHorse(raceHorse3);
    race1.addRaceHorse(raceHorse3);
    System.out.println(DB_HEADER + "Race horses created");
}

private ArrayList<Race> initializeRaces() {
    Calendar today = Calendar.getInstance();
    int month=today.get(Calendar.MONTH);
    int year=today.get(Calendar.YEAR);
    if (month==12) { month=0; year+=1;}
    else month+=1;

    ArrayList<Race> races = new ArrayList<>();
    races.add(new Race(UtilDate.newDate(year,month,17), 4, new StartTime("10:30")));
    races.add(new Race(UtilDate.newDate(year,month-2,17), 4, new StartTime("10:30")));
    for(Race r: races) {    db.persist(r);  }
    System.out.println(DB_HEADER + "Races created");
    return races;
}

private ArrayList<Horse> initializeHorses() {
    ArrayList<Horse> horses = new ArrayList<>();
    horses.add(new Horse("Seattle Slew", "a", 9, MALE, 11));
    horses.add(new Horse("Man o' War", "a", 5, FEMALE, 5));
    horses.add(new Horse("Citation", "b", 8, FEMALE, 9));
    horses.add(new Horse("Red Rum", "c", 7, MALE,2));
    horses.add(new Horse("Seabiscuit", "d", 6, FEMALE, 19));
    horses.add(new Horse("Kelso", "d", 11, MALE, 31));
    horses.add(new Horse("Native Dancer", "e", 13, MALE, 7));
    horses.add(new Horse("Affirmed", "f", 9, FEMALE, 6));
    horses.add(new Horse("Count Fleet", "g", 12, FEMALE, 13));
    for(Horse h: horses) { db.persist(h); }
    System.out.println(DB_HEADER + "Horses created");
    return horses;
}

private void initializeUsers() {
    Admin admin = new Admin("admin", "admin");
    Client client = new Client("client", "client");
    client.addMoney(20);
    db.persist(admin);
    db.persist(client);
    System.out.println(DB_HEADER + "Admin and Client created");
}
```

3. **Egindako errefaktorizazioaren deskribapena:**

Datu basearen hasieratzeko 4 zati nagusiak banatu ditut 4 metodo ezberdinetan refactor extract method eginez. Erabiltzaileak hasieratzeko, zaldiak hasieratzeko, lasterketak hasieratzeko eta lasterketa zaldiak eguneratzeko.

Metodoetan banatzeko, zaldiak eta lasterketak aldagai lokal sinpleetan gorde beharrean, arrayListetan sartu eta itzuli behar izan ditut. Lasterketa zaldiak sortzen dituen metodoak, lasterketak eta zaldiak atzigarri izan behar dituelako parametro moduan.

“Write simple units of code” (4-ko konplexutasun ziklomatikoa gehienez)

1. Hasierako kodea:

```java
public RaceHorse createRaceHorse(double winGain, Race race, Horse horse)
        throws RaceHorseAlreadyExist,WrongParameterException, RaceFullException, RaceDoesntExist, HorseDoesntExist, RaceFinished{

    if(race==null || horse==null || winGain<1) throw new WrongParameterException();

    Race newRace = db.find(race.getClass(), race.getKey());
    if(newRace==null) throw new RaceDoesntExist();

    Horse newHorse = db.find(horse.getClass(), horse.getKey());
    if(newHorse==null) throw new HorseDoesntExist();

    RaceHorse raceHorse = new RaceHorse(winGain, newRace, newHorse);
    if(newRace.getRaceHorses().contains(raceHorse)) throw new RaceHorseAlreadyExist();

    db.getTransaction().begin();
    if(!newRace.addRaceHorse(raceHorse)) throw new RaceFullException();
    db.getTransaction().commit();
    System.out.println("DB>>> New RaceHorse created and added to data base");
    return raceHorse;
}
```

2. Errefaktorizatutako kodea:

```java
public RaceHorse createRaceHorse(double winGain, Race race, Horse horse)
        throws RaceHorseAlreadyExist,WrongParameterException, RaceFullException, RaceDoesntExist, HorseDoesntExist, RaceFinished{

    if(race==null || horse==null || winGain<1) throw new WrongParameterException();

    Race newRace = getRace(race);
    Horse newHorse = getHorse(horse);

    RaceHorse raceHorse = addRaceHorse(winGain, newRace, newHorse);
    System.out.println("DB>>> New RaceHorse created and added to data base");
    return raceHorse;
}

private RaceHorse addRaceHorse(double winGain, Race newRace, Horse newHorse)
        throws RaceHorseAlreadyExist, RaceFullException {
    RaceHorse raceHorse = new RaceHorse(winGain, newRace, newHorse);
    if(newRace.getRaceHorses().contains(raceHorse)) throw new RaceHorseAlreadyExist();

    db.getTransaction().begin();
    if(!newRace.addRaceHorse(raceHorse)) throw new RaceFullException();
    db.getTransaction().commit();
    return raceHorse;
}

private Horse getHorse(Horse horse) throws HorseDoesntExist {
    Horse newHorse = db.find(horse.getClass(), horse.getKey());
    if(newHorse==null) throw new HorseDoesntExist();
    return newHorse;
}

private Race getRace(Race race) throws RaceDoesntExist {
    Race newRace = db.find(race.getClass(), race.getKey());
    if(newRace==null) throw new RaceDoesntExist();
    return newRace;
}
```

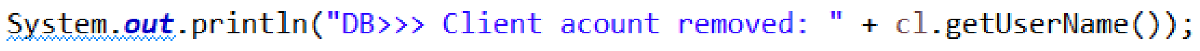
3. Egindako errefaktorizazioaren deskribapena:

3 refactor extract method egin ditut, metodo bakoitzak gauza bat egiteko:
DBtik zaldia lortu, DBtik lasterketa lortu, eta lasterketa zaldi berri bat sortu.

Duplicate code (kode bera behin bakarrik idatzi)

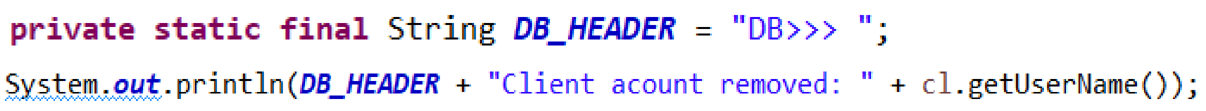
1. **Hasierako kodea:**

   Datu Baseko klase guztietan “DB>>>” -rekin hasten diren printak ditut.



```
System.out.println("DB>>> Client acount removed: " + cl.getUserName());
```

2. **Errefaktorizatutako kodea:**



```
private static final String DB_HEADER = "DB>>> ";
System.out.println(DB_HEADER + "Client acount removed: " + cl.getUserName());
```

3. **Egindako errefaktorizazioaren deskribapena:**

Refactor extract constant errefaktorizazioa erabili dut DB_HEADER atributu konstante lokala ezartzeko. Horrela, burukoa aldatu nahiko banu, behin bakarrik aldatu beharko nuke.

“Keep unit interfaces small” (4 sarrera parametro gehienez)

Kasu honetan, nire DataAccess-eko metodo guztiek 3 parametro edo gutxiago dituzte eta ez du zentzurik kopuru hori murrizten sahiatzeak. Beraz, parametroen aldetik, klasea ez dut aldatu.