TXOSTENA

1.- Taldekideen aurkezpena eta emandako denbora

Proiektu honen parte Oier Elola eta Unax Lazkanotegi izan gara, eta aurreko urtean Software Ingenieritza I irakasgaian garatu genuen Bets21 proiektuaren gainean egindako lanak (Sonar analisia + zuzenketa eta Probak) azaltzeko txostena da hau.

Gutxi gora behera esan dezakegu taldekide bakoitzak 18 bat ordu igaro dituela proiektu honetan egin beharreko atazetan garapenak, azterketak eta ikerketak egiten. Txosten hau, lan horren islada da, non bertan ikusi daitekeen bakoitzak landu duen funtzioaren analisi zehatza.

2. Hautatutako metodoa + analisia:

2.1. register() - Unax Lazkanotegi

Haututako funtzioa register() da eta bere kodea honakoa da:

```
public Pertsona register(String izena, String abizena1, String abizena2, String erabiltzaileIzena,
                        String pasahitza, String telefonoZbkia, String emaila, Date jaiotzeData,
                        String mota) throws UserAlreadyExist{

        TypedQuery<Pertsona> query = db.createQuery("SELECT p FROM Pertsona p WHERE
                                                p.erabiltzaileIzena=?1", Pertsona.class);
        query.setParameter(1, erabiltzaileIzena);
        List<Pertsona> pertsona = query.getResultList();
        if(!pertsona.isEmpty()) {
                throw new UserAlreadyExist();
        }else {
                Pertsona berria = null;
                if(mota.equals("admin")) {
                        berria = new Admin(izena, abizena1, abizena2, erabiltzaileIzena, pasahitza,
                                        telefonoZbkia, emaila, jaiotzeData);
                }else if (mota.equals("langilea")) {
                        berria = new Langilea(izena, abizena1, abizena2, erabiltzaileIzena, pasahitza,
                                        telefonoZbkia, emaila, jaiotzeData);
                }else if (mota.equals("bezeroa")) {
                        berria = new Bezeroa(izena, abizena1, abizena2, erabiltzaileIzena, pasahitza,
                                        telefonoZbkia, emaila, jaiotzeData);
                }
                db.getTransaction().begin();
                db.persist(berria);
                db.getTransaction().commit();
                return berria;
        }
}
```

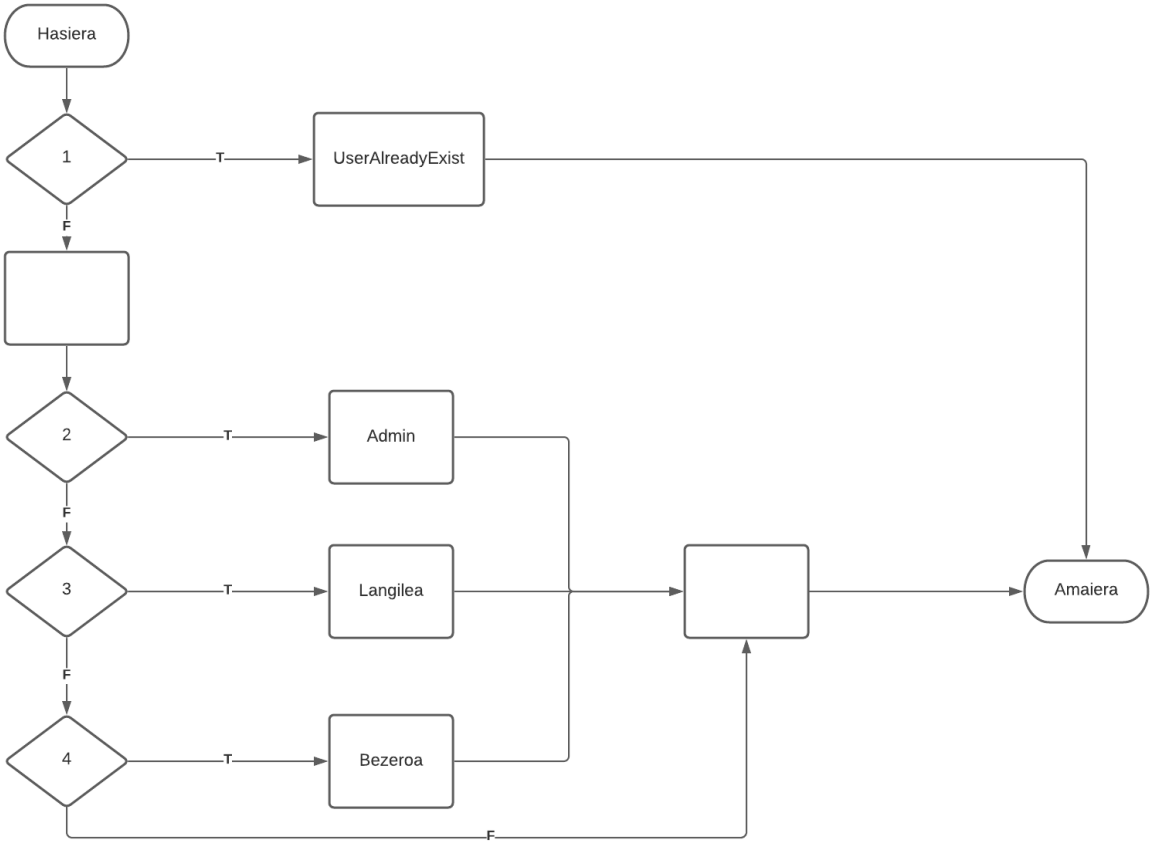
Metodo honen funtzionamendua aztertzeko bi motatako probak egin behar dizkiogu, bateragarriak eta integrazioak.

PROBA BATERAGARRIAK:

Proba bateragarrian, bi motatakoak bereizi ditzazkegu: kutxa txuriko eta kutxa beltzeko probak.

KUTXA TXURIA:

Hasteko, ikus dezakegun irudian register() metodoaren fluxu diagramen grafoa:



Honek, metodoaren funtzionamendua adierazten du, baldintza kasuak adieraziz eta dituen biderapen posibleak. If baldintza bat duen bakoitzean, bidea bitan banatzen da, true edo false kontuan hartuta eta honek aukera kasu ezberdinak existitzea ahalbidetzen du.

Azterketa honen ondoren erraza da jakitea zein den konplexutasun ziklomatikoa:

V(G)= #erabaki-nodo + 1= 4 + 1= 5

Behin Konplexutasun-ziklomatikoa zenbatekoa den jakinda, proba kasuak aztertu ditzazkegu:

#	BALDINTZA	PROBA KONTEXTUA		ITXARON EMAITZAK	
		DB EGOERA	SARRERA	DB EGOERA	IRTEERA
1	IF1(T)	“unarux” ∈ DB	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	Ez da aldatzen	UserAlreadyExist

2	IF1(F) AND IF2(T)	“unarux” ∉ DB	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	“unarux” ∈ DB	bez eguneratua
3	IF1(F) AND IF2(F) AND IF3(T)	“unarux” ∉ DB	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “langilea”)	“unarux” ∈ DB	lan eguneratua
4	IF1(F) AND IF2(F) AND IF3(F) AND IF4(T)	“unarux” ∉ DB	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “admin”)	“unarux” ∈ DB	admin eguneratua
5	IF1(F) AND IF2(F) AND IF3(F) AND IF4(T)	“unarux” ∉ DB	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “gaizki”)	Ez da aldatzen	null

OHARRA: DB egoera adierazten den “unarux” string sarrera erabiltzaileren batek erabiltzaile izen hori duen aztertzen du. Kasu hori existitzekotan, jada DB barruan dagoela adieraziko du.

Kutxa txuriko proba kasuen inplementazioa RegisterDAW.java fitxategian dago.

Aurkitutako akatsak: Planteatutako kasu guztiek espero genuen emaitza eman dute.

KUTXA BELTZA:

Orain aztertuko diren proba kasuak honakoak izango dira, metodoaren goiburukoa harturik izan ditzazkeen sarrera baldintzak dira:

SARRERA BALDINTZA	BK EGOKIA	BK EZ EGOKIA
Izena balioa	izena != null	izena == null
abizena1 balioa	abizena1 != null	abizena1 == null
abizena2 balioa	abizena2 != null	abizena2 == null
erabiltzaileIzena balioa	erabiltzaileIzena != null	erabiltzaileIzena == null
pasahitza balioa	pasahitza != null	pasahitza == null
telefonoZbkia balioa	telefonoZbkia != null	telefonoZbkia == null
email balioa	email != null	email == null
jaiotzeData balioa	jaiotzeData != null	jaiotzeData == null
mota balioa	mota != null	mota == null
hemezortzi urte izan	now - jaiotzeData >= 18	now - jaiotzeData < 18
email egokia	email ∈ "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"+"[A-Za-z0-9]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$"	email ∉ "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"+"[A-Za-z0-9]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$"
pasahitza egokia	pasahitza.length() >= 8	pasahitza.length() < 8
erabiltzaileIzena egokia	erabiltzaileIzena.length()>=4	erabiltzaileIzena.length()<4
telefonoZbkia egokia	telefonoZbkia.length()==9	telefonoZbkia.length()!=9
erabiltzaileIzena ez existitu	erabiltzaileIzena ∉ DB	erabiltzaileIzena ∈ DB
mota bezeroa	mota == “bezeroa”	
mota langilea	mota == “langilea”	
mota admin	mota == “admin”	

Ikusita sarrera baldintza kasu guztietarako zein den bere baliokidetasun egokia eta zein ez, proba kasu batzuk erator ditzazkegu:

#	ESTALITAKO BK	PROBA KONTEXTUA		ITXARON EMAITZAK	
		DB EGOERA	SARRERA	DB EGOERA	IRTEERA
1	1-16	user ∉ DB	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”,	bez ∈ DB	Bezeroa bez

			”unocen@gmail.com”, “19/10/2001”, “bezeroa”)		
2	1-15,17	Egoera berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “langilea”)	lan ∈ DB	Langilea lan
3	1-15,18	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “admin”)	adm ∈ DB	Admin adm
4	19	Egoera Berdina	(null, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	Aldaketarik ez	null
5	20	Egoera Berdina	(“Unax”, null, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	Aldaketarik ez	null
6	21	Egoera Berdina	(“Unax”, ”Lazkanotegi”, null, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	Aldaketarik ez	null
7	22	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, null, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”,	Aldaketarik ez	null

			“bezeroa”)		
8	23	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, null, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	Aldaketarik ez	null
9	24	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, null, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	Aldaketarik ez	null
10	25	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, null, “19/10/2001”, “bezeroa”)	Aldaketarik ez	null
11	26	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, null, “bezeroa”)	Aldaketarik ez	null
12	27	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, null)	Aldaketarik ez	null
13	28	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2008”, “bezeroa”)	Aldaketarik ez	null
14	29	Egoera Berdina	(“Unax”, ”Lazkanotegi”,	Aldaketarik ez	null

			”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocenasd”, “19/10/2001”, “bezeroa”)		
15	30	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unax”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	Aldaketarik ez	null
16	31	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”un”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	Aldaketarik ez	null
17	32	Egoera Berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “6888240012”, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	Aldaketarik ez	null
18	33	user ∈ DB	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “19/10/2001”, “bezeroa”)	Aldaketarik ez	UserAlreadyExist

Proba kasu hauen inplementazioa ikusteko, joan zaitez RegisterDAB.java fitxategira, bertan daude guztiak ikusgai.

AURKITUTAKO AKATSAK: Metodo honek zituen proba kasu gehienak, ez dira datu basean kontrolatzen, GUI interfazean baizik eta horregatik proba kasu ugari daude ez dutenak esperotako emaitza itzultzen. Oker funtzionatzen duten guztiek null objetu bat itzuli beharko lukete baina ez dute betetzen esperotakoa. Bakarrik metodo bakar bat dago null itzuli behar duenen artean funtzionatzen duena eta ongi egiten du datu basean filtratzen delako bere baldintza.

MUGA BALIOAK:

#	DB EGOERA	Sarrerako datuak	Emaitza
1	user ¢ DB	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “02/10/2003”, “bezeroa”)	null
2	Egoera berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “03/10/2003”, “bezeroa”)	Bezeroa bez
3	Egoera berdina	(“Unax”, ”Lazkanotegi”, ”Forcen”, ”unarux”, ”unaxPA1234”, “688824012”, ”unocen@gmail.com”, “04/10/2003”, “bezeroa”)	Bezeroa bez

OHARRA: Kontuan izan behar da muga balio hauek 2021/10/03 ko data erreferentzia bezala hartu duela.

INTEGRAZIO PROBA:

Datu basean zuzenean egindako testak kontuan hartu ondoren, berdina egin beharra dago negozio logikako metodoak erabiliz, hau da proba kutxa beltzeko proba kasu berdinak izango ditugu eta berriro guztia ez adierazteagatik, goiko klase estalpen taularen probak egin behar dira berriro.

Kasu honetan, ez ditut guztiak inplementatu, soilik horien artean batzuk: 1, 2, 3, 12 eta 18 estalpen kasuak.

Hauek izango dira datu basean exekutaturiko proba egokiak eta nola ez, kontuan izanik egoera berdinean dagoela negozio logika, hau da, berak ez dituenez konprobatzen argumentuen balioak, null balioa itzuli behar duen kasu gehienetan bere funtzionamendua okerra izango da.

Beraz, aurkitutako akatasak: 0, inplementatutako testak ongi funtzionatzen dute.

Gainera, honetaz gain, programatzerako orduan klaseak proba ditzagun, beste aukera egoki bat daukagu eskura: Mockito

Mockito-k momentuz garatuta ez daukagun klase baten bikoitz bat sortzea eskaintzen digu, honela hainbat simulazio ezberdin egiteko aukera izan dezagun eta programak duen jarrera aztertu dezakegu. Hau egin ahal izateko, guk adieraziko diogu dei bat egiterako orduan, zuzenean zein izango den itzuliko duen balioa eta gero nahikoa izango da konprobatzearekin ea itzulitakoa espero genuenaren balio berdina eman duen.

Mockito inplementatzen duten proba kasuak lehen adierazi diren guztiak dira, hau da, taulako 18 estaldura probak. Honen posible da lortzea proba guztiek nahi dugun emaitza itzultzea, kontuan izanik guk adierazten diogula azkenean zein izan behar den itzuli behar duen balioa eta horregatik lortu dugu inplementazio guztiek emaitza egokia izatea.

Proba kasu hauen inplementazioa ikusi nahi baduzu, nahikoa da RegisterMockInt.java fitxategira bazoaz, bertan daude eta proba kasuen kodeak.

2.2. deleteApustua() - Oier Elola

Hautatutako funtzioa deleteApustua() izan da, eta jarraian ikusi dezakezue zein den bere kodea:

```
        public Bezeroa deleteApustua(Apustua apustua) throws EventFinished{
                try {
1               db.getTransaction().begin();
2               Apustua a=db.find(Apustua.class, apustua.getIdentifikadorea());
3               ArrayList<Pronostikoa> pronostikoak = a.getPronostikoak();
4               Date today = new Date();
5               for(Pronostikoa p : pronostikoak) {
6                       Date eventDate = p.getQuestion().getEvent().getEventDate();
7                       if(!eventDate.after(today)) {
8                               db.getTransaction().commit();
9                               throw new EventFinished();
10                      }
11              }
12              Bezeroa bezeroa = a.getBezeroa();
13              bezeroa.removeApustua(a);
14              if(a.getErrepikatua()!=null) {
15                      Errepikapena errepikapen=bezeroa.getErrepikapena(a.getErrepikatua());
16                      errepikapen.eguneratuHilHonetanGeratzenDena(a.getKopurua());
17              }
18              bezeroa.addMugimendua("Apustua ezabatu
("+a.getIdentifikadorea()+")",a.getKopurua(),"bueltatu");
19              for(Pronostikoa p : pronostikoak) {
20                      p.removeApustua(a);
21              }
22              Vector<Errepikapena> errepikatzaileak= bezeroa.getErrepikatzaileak();
23              for(Errepikapena er : errepikatzaileak) {
24                      Bezeroa bez = er.getNork();
25                      Apustua apusErr = bez.baduApustua(a);
26                      if(apusErr!=null) {
27                              bez.removeApustua(apusErr);
28                              bez.addMugimendua("Apustu errepikatua ezabatu ("+bezeroa+")",
apusErr.getKopurua(), "bueltatu");
29                              for (Pronostikoa p: apusErr.getPronostikoak()) {
30                                      p.removeApustua(apusErr);
31                              }
32                              er.eguneratuHilHonetanGeratzenDena(apusErr.getKopurua());
33                              db.remove(apusErr);
34                      }
35              }
36              db.remove(a);
37              db.getTransaction().commit();
38              return bezeroa;
39              }catch(NullPointerException e) {
40                      db.getTransaction().commit();
41                      throw new NullPointerException();
42              }
```

Esan beharra dago kodeak aldaketa txiki bat duela: hasiera batean try-catch baten barnean ez zen aurkitzen kodea, baina proba kasuetan ongi funtziona zezan aldaketa hori egitera behartuta ikusi nintzen. Eta bestetik, EventFinished salbuespena jaurti baino lehenago ere transakzioa ixteko eskaera egiten dut; izan ere, honekin ere arazoak izan baititut proba kasuetan: transakzio berri bat irekitzen saiatzen nintzen hau itxi gabe.

Bi motatako probak ikusi ahal izango dituzue jarraian: proba bateragarriak eta intergrazio probak. Bakoitzaren barnean, bi proba ezberdin ikusiko dituzue baina guztien egituraketa berdina izango da: proba diseinua eta proben exekuzioen ostean analisia.

PROBA BATERAGARRIAK

Bi motatako probak bereiziko ditugu atal honetan: kutxa zuriko probak eta kutxa beltzeko probak:

- ➢ Kutxa zuriko proba: Proba mota honetan kode guztia exekutatzen dela bermatu nahi da.
- ➢ Kutxa beltzeko proba: Funtzioaren kodea ikusi gabe eta goiburukoa jakinik soilik, bere portaera aztertzeko egiten da. Gehienbat, kutxa zuriko proben osagarriak dira.

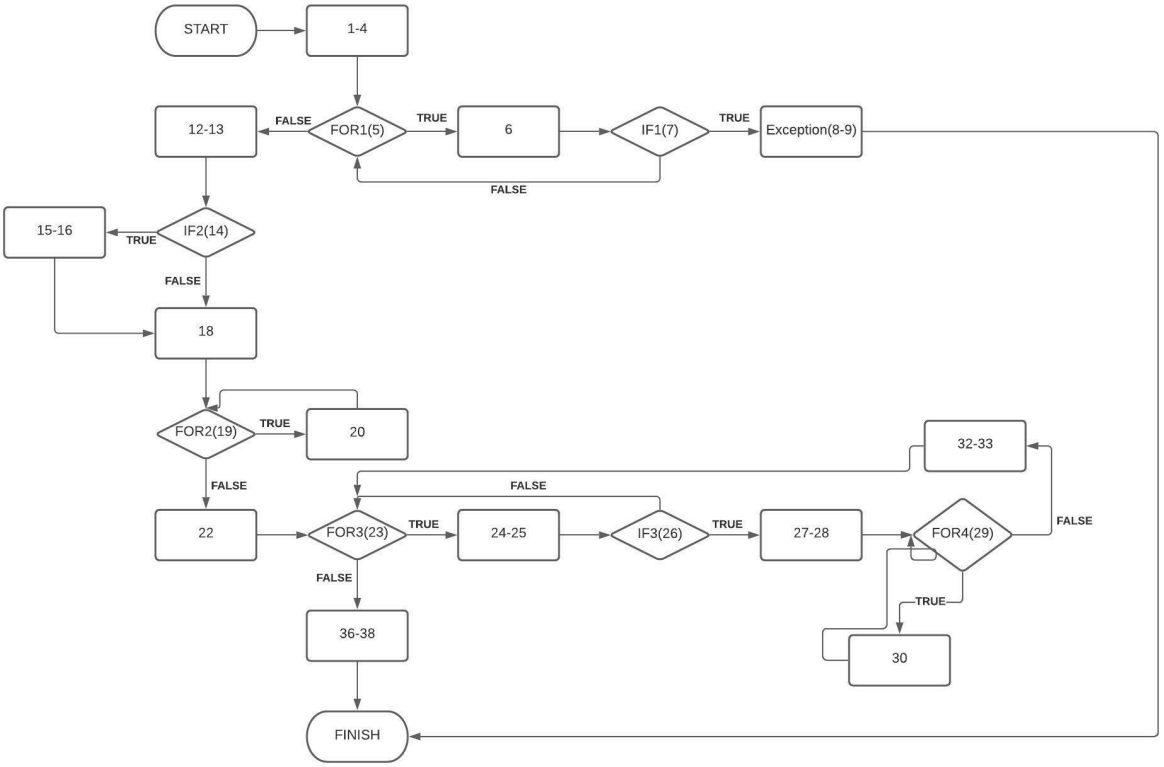
Behin proba bakoitzaren nondik norakoak jakinik, goazen zentratzera gure funtzioan, zehazki deleteApustua() funtzioaren kutxa zuriko eta kutxa beltzeko probetan.

KUTXA ZURIKO PROBAK

Jarraian ikusi dezakezuena fluxu kontrol grafoa da; hau da, funtzioak nola funtzionatzen duen adierazteko erabiltzen da. Elementu bakoitzaren barnean aurkitzen den zenbakitxoak kode zenbakia adierazi nahi du (zenbaki batzuk ez dira agertzen kortxetea soilik dagoelako lerro horretan,eta aldaketa ondorioz sartutako try-catch egitura ere ez da aurkitzen diagraman). Hori ikusita, bere konplexutasun ziklomatikoa zein den kalkula dezakegu:

V(G)= #erabaki-nodo + 1= 3 + 1= 4

Azalpena: Nahiz eta 4 for ikusi ditzakezuen daudela hor, lehen bietatik igarotzea beharrezkoa da: izan ere, Apustua motako objektuaren osaeran definituta dago bere Pronostikoa objektuen ArrayList hori null ezinduela izan,eta zeharkaldi horiek derrigorrezkoak dira eta erabaki nodotik kanpo geratzen dira, eta azken for egiturarekin ere berdina gertatzen da: aurreko if egituran sartuz gero derrigorrez igaro behar du bertatik. Beraz, beste era batera kontatuta konplexutasun ziklomatikoa honela izango litzateke: 3 if egitura + 1= 3 da bere konplexutasun ziklomatikoa. (Agian 5 izan daiteke, baina segurutik jotzea erabaki dut badaezpada).



Beraz, konplexutasun ziklomatikoa zein den jakinik, orain proba kasuak planteatzeko unea da:

#	BALDINTZA	PROBA KONTEXTUA		ITXARON EMAITZAK	
		DB EGOERA	SARRERA	DB EGOERA	IRTEERA
1	IF1(T)	e ∈ DB, q ∈ DB, pronos ∈ DB, u ∈ DB, apu ∈ DB, apu ∈ u.apustuak	apu	Ez da aldatzen	Exception EventFinished
2	IF1(F) AND IF2(T) AND FOR3(F) AND IF3(F)	e ∈ DB, q ∈ DB, pronos ∈ DB, u ∈ DB, apu ∈ DB, apu ∈ u.apustuak apu ∈ pronos.apustuak apu.errepikatua!=null	apu’	apu ‘ ∉ u’ .apustuak apu’ ∉ DB erre eguneratu apu’ ∉ pronos.apustuak	u eguneratua

		u’ ∈ DB, u, u’ ∈ erre, apu’=apu apu’∈ u.apustuak apu’ ∈ pronos.apustuak			
3	IF1(F) AND IF2() AND FOR3(T) AND IF3(F)	e ∈ DB, q ∈ DB, pronos ∈ DB, u ∈ DB, u’ ∈ DB, apu ∈ DB, apu ∈ u.apustuak apu ∈ pronos.apustuak u, u’ ∈ erre, erre ∈ DB, apu ∉ u’.apustuak	apu	apu ∉ u.apustuak apu ∉ DB apu ∉ pronos.apustuak	u eguneratua
4	IF1(F) AND IF2() AND FOR3(T) AND IF4(T)	e ∈ DB, q ∈ DB, pronos ∈ DB, u ∈ DB, u’ ∈ DB, apu ∈ DB, apu ∈ u.apustuak apu ∈ pronos.apustuak u, u’ ∈ erre, erre ∈ DB, apu’=apu apu’ ∈ u’.apustuak, apu’ ∈ pronos.apustuak	apu	apu ∉ u.apustuak apu ∉ DB apu ∉ pronos.apustuak apu’ ∉ u’.apustuak apu’ ∉ DB apu’ ∉ pronos.apustuak erre eguneratu	u eguneratua

OHARRA: berdin digu IF2-ren balioa T edo F izatea azken bietarako azkenean FOR3 eta IF3-ra helduko baita eta azken bi kasuen muina hor dago.

Balioak:
e=("event1", oneDate),
q=("question1", 2)
pronos=(“pronos1”,2),
apu=(4,[pronos],u,null),
apu’=(4,[pronos],u’,u),
u=(“izena”,”abizena1”,”abizena”,”erabizena”,”pasahitza”),
u’=(“izena2”,”abizena21”,”abizena22”,”erabizena2”,”pasahitza2”),
erre[Errepikapen motako objektua]=(u’,u, zenbakibatzuk, zenbakiak);

Bezeroa motako objektuak hobeto definitzen dira proba kasuetan, hemen informazio txiki bat soilik ematen da.

Kutxa zuriko proba kasuen inplementazioa deleteApustuaDAW.java fitxategian aurkitu daiteke.

Aurkitutako Akatsak: Planteatutako proba kasu guztiak espero zuten emaitza eman dute.

KUTXA BELTZEKO PROBA KASUAK

Azpiko taulan ikusi dezakezuena honakoa da: funtzioaren goiburukoa harturik zeintzuk izan daitezkeen sarrerako balioak. Gogoratu funtzioaren goiburukoa honakoa dela:

```
function deleteApustua(Apustua apustu) throws EventFinished
```

SARRERA BALDINTZA	**BK EGOKIA**	**BK EZ EGOKIA**
Apustua balioa	!=Null (1) DBan egotea (2)	==null (6) DBan ez egotea (7)
Apustua.pronostikoak[i].event.date	>Today (3)	<Today (8)

Beraz, ikusirik sarrera baldintza bakoitzerako zein den bere baliokidetasun-klase egoki eta ez egokia, proba kasu batzuk eratorriko dira:

#	**ESTALITAKO BK**	**PROBA KONTEXTUA**		**ITXARON EMAITZAK**	
		DB EGOERA	**SARRERA**	**DB EGOERA**	**IRTEERA**
1	1,2,3,4,5	e ∈ DB, q ∈ DB, pronos ∈ DB, u ∈ DB, apu ∈ DB, apu ∈ u.apustuak	apu	apu ∉ u.apustuak apu ∉ DB apu ∉ pronos.apustuak	u eguneratua
2	6	Egoera berdina	null	Aldaketarik ez	Exception NullPointer
3	7	Egoera Berdina	apu2	Aldaketarik ez	null
4	8	Egoera	apu3	Aldaketarik ez	Exception

		Berdina			EventFinished

Oharra: DB EGOERA atalean “Egoera berdina” egoteak esan nahi du #1 kasuko egoeran aurkitzen dela DB; izan ere, apu2, apu3 eta apu4 kasuan beraien elementuren bat null balioa izango da eta orudan ez dago aldatu beharrik datu basearen egoera (apu2 kasuan objektua ez da datubasean aurkituko, apu3 kasuan bere pronostiko baten gertaera dagoeneko gertatuta egongo da eta apu4ren kasuan bere pronostiko zerrenda hutsa egongo da).

Kutxa beltzeko proba kasuen inplementazioa deleteApustuaDAW.java fitxategian aurkitu daiteke.

Aurkitutako akatsak:

#	PROBA KONTEXTUA		ITXARON EMAITZAK		IRTEERA EMAITZAK	
	DB EGOERA	SARRERA	DB EGOERA	IRTEERA	DB EGOERA	IRTEERA
3	Egoera berdina	apu	Aldaketarik ez	-	Aldaketarik ez	Exception

Muga balioen analisia

JUNIT + MOCKITO PROBAK

Proba mota honetan, programaren funtzionamendua simulatzen da programatzaileak erabakitzen duen moduan, eta egoera horietan lortzen den emaitza ikusi nahi da ea egokia den ala ez.

Kasu honetan, erabiliko den informazioa kutxa beltzeko proban erabilitakoa izango da; eta beraz, erabiliko diren proba kasuak berdinak izango dira. Gogorarazteko, jarraian utziko dizuegu baldintzak eta eratorritako proba kasuak:

SARRERA BALDINTZA	BK EGOKIA	BK EZ EGOKIA
Apustua balioa	!=Null (1) DBan egotea (2)	==null (6) DBan ez egotea (7)
Apustua.pronostikoak[i].event.date	>Today (3)	<Today (8)

Erratorritako proba kasuak:

#	ESTALITAKO BK	PROBA KONTEXTUA		ITXARON EMAITZAK	
		DB EGOERA	SARRERA	DB EGOERA	IRTEERA
1	1,2,3,4,5	e ∈ DB, q ∈ DB, pronos ∈ DB, u ∈ DB, apu ∈ DB, apu ∈ u.apustuak	apu	apu ∉ u.apustuak apu ∉ DB apu ∉ pronos.apustuak	u eguneratua
2	6	Egoera berdina	null	Aldaketarik ez	Exception NullPointer
3	7	Egoera Berdina	apu2	Aldaketarik ez	null
4	8	Egoera Berdina	apu3	Aldaketarik ez	Exception EventFinished

Kasu honetan, inplementazioa ikusi nahi izanez gero, deleteApustuaMockInt.java fitxategian aurkituko duzue.

Akatsak:

1. kasuan ez dut lortu proba kasuak ongi funtzionatzea, beharbada izan daiteke planteamendua ez dela egokia izan.

INTEGRAZIO PROBA

Integrazio probetan, moduluak elkartzen dira eta beraien funtzionamendua egokia dela ziurtatzen da. Gure kasuan, negozio logika (BLFacadeInplementation) eta datu basea (DataAccess) izango dira integrazio probetan parte hartuko duten interfazeak.

deleteApustua() funtzioaren integrazio proba kasuak ezin izan dira inplementatu hainbat arazo izan direla medio:denbora, arazoak datu-basearekin, …

Taldekide bezala, proiektu honetan parte hartu duen orori (bai ikaskideari bai irakasleari) barkamena eskatu nahi diot nire partetik lana ez baitut behar bezala burutu, eta nik egin dudan lana hobeto egin dezakedala esan dezaket.

3. Informazio gehigarria

Jarraian atxikituta doaz GitHub-eko link-a eta SonarCloud-eko linkak, lehenak Bets21 proiektuaren kodea izango du eta bigarrenak SonarLint analisiaren emaitzak:

- GitHub: https://github.com/Elola27/SI2LazkanotegiElola
- SonarCloud: https://sonarcloud.io/dashboard?id=SI2

Kodea FINAL karpetaren barruan aurkituko duzue, eta sonar.pdf fitxategia SonarLint tresna erabiliz egindako proba estatikoen inguruko dokumentazioa da.