Programazio Modularra eta Objektuei Bideratutako Orientazioa– 4. Laborategia

Aurrebaldintzak

Ikaslea Eclipse ingurunea kontrolatzen du eta Java ikuspegia erabiltzen badaki. Eta, klase sinpleak eta klaseen arteko erlazioak sortzen eta adierazten (UML bidez) badakiela suposatzen da. Badaki proba kasuak (TestCase) eta proba multzoak sortzen ere JUnit frameworkaren bitartez. Interfaze kontzeptua ulertu izana eta interfaze bat inplementatzen duen klase bat sortzeko gai izatea. Finkatu izana Objektuei Bideratutako paradigmaren ezagutza, ulertzea UML diagramak, eta *Junit*ak sortzen jakitea.

Helburuak

Laborategi honetan Datu Mota Abstraktu (DMA-TAD) eta Egoera Makina Abstraktuen (EMA-MAE) kontzeptuak landuko dira. Pantailan mezuak idazten ere ikasiko dugu (sistemako konsolan), eta ArrayList<> klasearen erabilpena ikasiko da, bere metodoak eta iteradorea baliatuz.

Laborategi hau bukatzean, ikasleak hurrengo gaitasun/ezagutza lortuko ditu:

- EMAk (MAE) eta DMAk (TAD) klaseak baliatzen dituzten ariketak burutzeko gaitasuna, EMAk eta DMAk ondo desberdinduz eta identifikatuz.
- *ArrayList*<> eta iteratzaileak erabiltzen jakitea.
- Sistemako kontsolan mezuak erakusten jakitea.

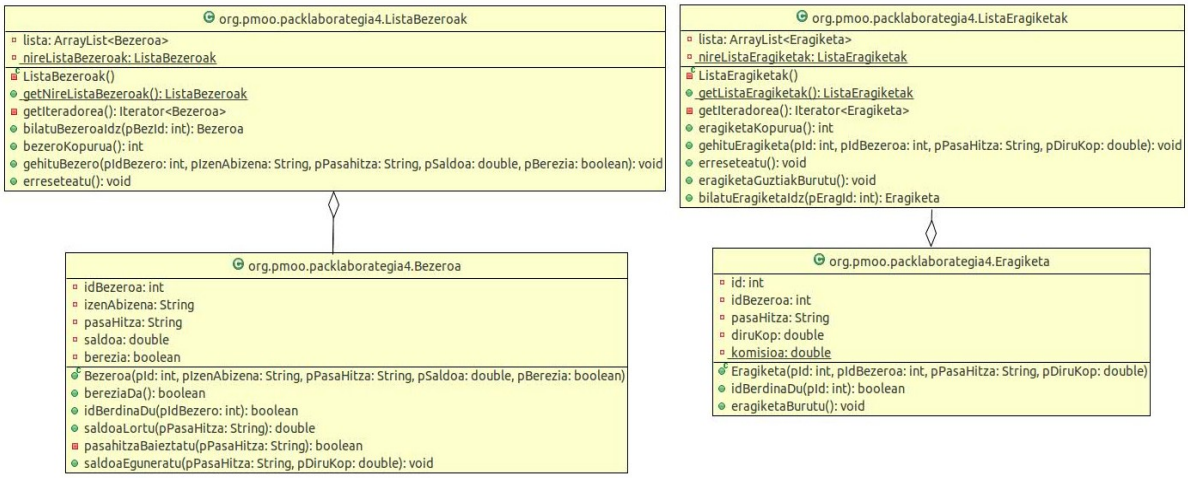
Motibazioa

Laborategi honetan, lehenengo aldiz, hainbat klase (alegia, 4 klase) erlazionatuta erabiliko dira. Printzipioz, klase diagrama eskaintzen da, diseinuan sakontzea ez dagoelako, momentuz, helburuen artean, hurrengo kurtsoan, Software Ingeniaritzan landuko baitira.

Ataza bakarra: PromoBank

Ataza honetan, PromoBank banku-kudeaketa testuinguruan, bezeroen banku-eragiketak kudeatzeko aplikazio bat eraiki nahi dugu.

Hurrengo irudian, ataza hau burutzeko behar diren klaseen diagrama azaltzen da. Ikus daitekeenez, 2 EMA identifikatu dira: ListaBezeroak (bankuak dituen bezeroen zerrenda duelarik) eta ListaEragiketak (egindako eragiketen zerrenda duelarik).



Irudia – PromoBank eragiketak kudeaketako UML klase diagrama.

Bezeroen zerrendak, *Bezero* guztiei dagozkien informazioa du. *Bezero* bakoitzeko, zera gordetzen da: Bezeroaren identifikadorea, bere izena, bere pasahitza, bere saldoa eta atributu boolear bat "bezero berezia" den adierazteko.

Eragiketen zerrendak denbora errealean gertatzen ari diren eragiketen informazioa gordetzen du. *Eragiketa* bati identifikadore bat dagokio, eragiketa egin duen bezeroaren identifikadorea eta bezeroak eragiketa egiteko sartu duen pasahitza.

Ariketa ez konplikatzearren, eragiketa guztiak, diru-ateratze motatakoak direla suposatuko da. Diru-atera eragiketa bat burutzean, bezeroaren saldoari eragiketak esleitua duen diru kopurua deskontatzea dakar (**KONTUZ!!** ea bukaeran nork egin behar duen saldoaEguneratu....). Esan behar da ere, diru-atera eragiketek %0.1eko komisioa aplikatzen dietela bereziak ez diren Bezeroei.

Eragiketa zerrendan agertzen diren eragiketa guztiak gauzatzeko behar den metodoa inplementatu nahi dugu, hala nola, *eragiketaBurutu()* metodoa. Eragiketa bat egin baino lehenago bezeroak sartutako pasahitza egokia dela baieztatu behar da, hau da, bezeroak tekleatutako pasahitza eta bezeroak pasahitza atributuan duten informazioak bat datozela. Baieztapena ez balitz ondo gauzatuko, orduan pantailatik mezu bat agertu beharko litzateke, ohartarazteko mezu bat pantailaratuko litzateke, eta transakzioa baliogabetu.

Segurtasun arrazoiengatik, *Bezero* klaseko objektu batek ez du beste klase bati bere pasahitza erakutsiko. Arrazoi honengatik, *pasahitzaBaieztatu()* metodo pribatua erabiliko da.

Metodo honek, sarrera bezala jasotzen duen gakoa *Bezeroaren* gakoaren berdina bada TRUE itzuliko du, FALSE bestela. Horretxegatik, *saldoaLortu()* eta *saldoaEguneratu()* metodo publikoek, sarrera bezala jasotako gako hau aztertuko dute, eta zuzena bada, beren eginkizuna beteko dute.

Eskatzen da:

1. Osatu *eragiketaBurutu()* metodoari dagokion **Sekuentzia Diagrama**.
2. Ondoren aipatzen diren metodok eta atributuak inplementatu. Baita metodo hauek probatzeko behar diren JUnitak.

- **Bezeroa Klasea**
  - **Atributuak:** identifikadore bat (int), izenAbizena(String), pasahitza (String), saldoa (double) eta berezia den ala ez jakiteko (boolean).
  - **Metodo eraikitzailea:** atributu guztiak hasieratzeko parametroak jasotzen ditu.
  - **Gainontzeko metodoak*:***
    - *idBerdinaDu()* id bat parametrotzat hartuta, objektuaren id-arekin konparatuko du, berdinak balira true bueltatuz eta bestela false.
    - *pasaHitzaBaieztatu(), gorago komentatu denez boolear bat bueltatuko du, jasotzen duen pasahitza eta bezeroarena kointzidentzien dutenean. Pribatua izango da.*
    - *saldoaLortu()* parametro bezala jasoko duen pasahitza egokia dela konprobatuko du, saldoa bueltatuz eta bide batez erakutsiz, beti pasahitza konprobatu ostean. Pasahitza okerra balitz mezu bat aterako du egoera hori adierazteko ("bezeroaren izenAbizena, pasahitza okerra") eta bueltatuko duen saldoa 0.0 izango da. Pasahitza egokia balitz, saldoa bueltatzeaz gain, pantailaratu ere egingo du ("bezeroaren izenAbizena,zure saldoa "+**this**.getSaldoa()+" da").
    - *saldoaEguneratu()* Lehenik, parametro bezala duen diru kopurua positiboa izatea begiratuko du. Bigarrenik, kopuru hori bezeroaren saldoa baino txikiagoa dela begiratuko du. Eta azkenik, bezeroaren pasahitza eta jasotakoa berdinak direla konprobatuko du. Baldintzak betez gero, bezeroaren saldoa eguneratuko da, eta gainerako kasuetan ez. Edozein kasutan, pantailatik mezu bat bueltatuko du burutu den eragiketa ondo ala txarto bukatu den adierazteko. Ondo burutu bada, "*bezeroarenIzenAbizena,* zure saldo berria XXX da" pantailaratuz, eta gaizki burutu bada, "*bezeroarenIzenAbizena,* saldoa ezin izan da aldatu" inprimatuko luke.

- **ListaBezeroak Klasea (EMA)**
  - **Atributuak:** bezeroen lista duen klasea izango da, kasu honetan ArrayList baten bitartez inplementatua. EMA bat da, klase honen instantzia bakarra dagoelako.
  - **Metodo eraikitzailea:** Bezeroen lista sortzen du. Singleton patroia aplikatuz, beraz eraikitzailea pribatua izango da.

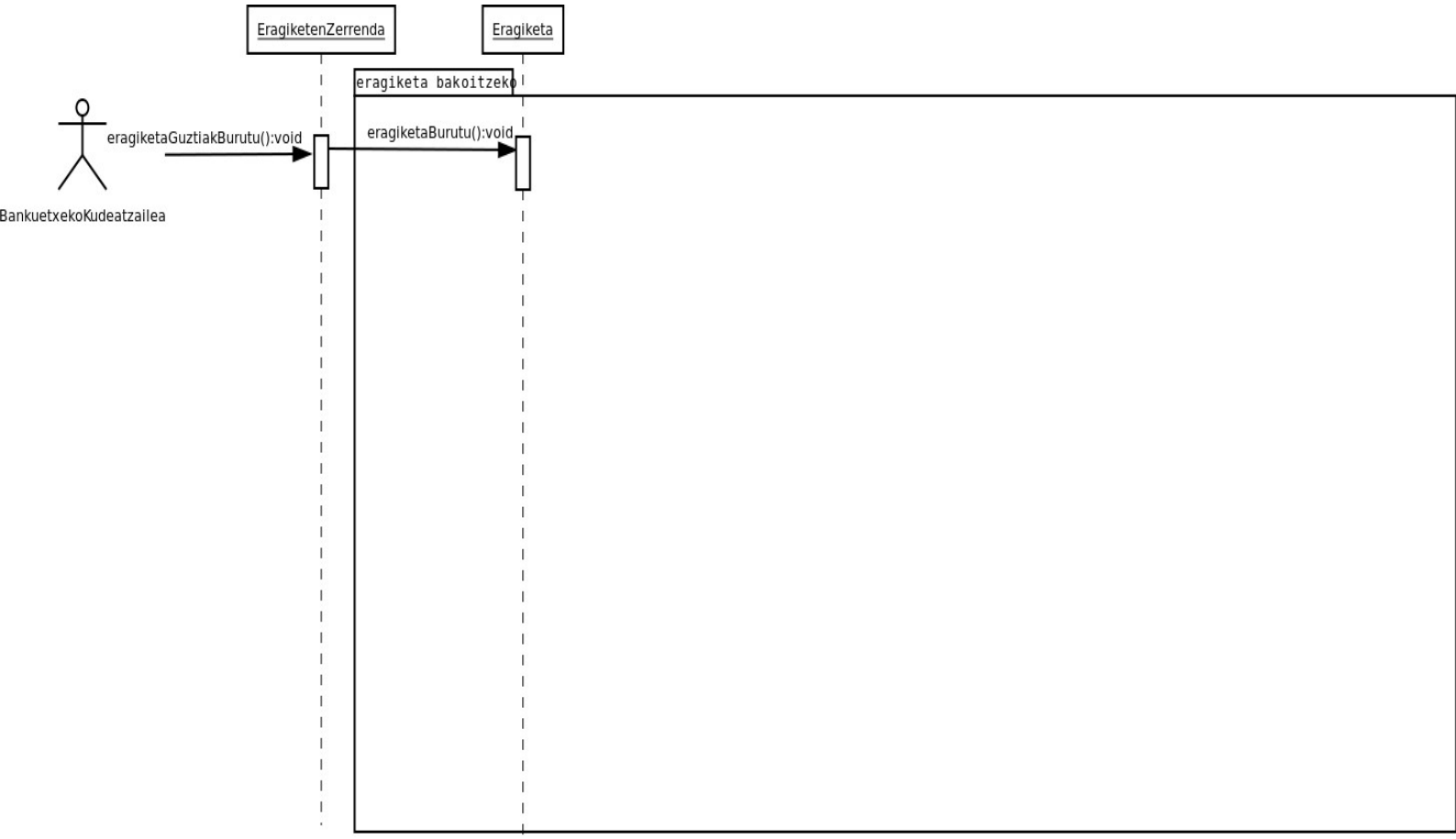
- **Gainontzeko metodoak*:***
    - *bezeroKopurua()* publikoa izango da Junitetan ikusgarri izatea beharko dugulako, bestela ez du zertan publikoa izan behar, printzipioz zerrenda ez den beste inork ez lukeelako zertan jakin behar zenbat elementu dituen. Zerrendaren elementu kopurua bueltatuko du. Metodo hau Junitetan erabiliko da, adibidez jarraian agertzen den *gehituBezero()* konprobatzeko.
    - *bilatuBezeroaIdz()* ***KONTUZ HEMEN!!*** *jasotzen du bezero baten id-a eta horrekin kointzidentzia duen barruko bezeroa bueltatuko du, baldin badago horrelakorik, bestela null. Jakiteko ea topatu duen erabiliko da idBerdinaDu().* ***IKASTEKO OHARRA.*** *Pasatuko bagenio Bezero bat eta ez bezero baten Id-a, eta bezeroa topatzeko equals()* erabiliko bagenu, gertatu zitekeen bezeroa ez egotea, nahiz eta id, izenAbizen eta gainontzeko atributuak dituen bezero bat eduki lista barruan. Gogoratu *equals()* metodoak defektuz objektuen memoria helbideak konparatzeko dituela eta ez objektuen "edukia".
    - *gehituBezero()* bezeroen zerrendan jasotzen duen bezero bat gehitzen du emandako parametroekin. Bezero hori jadanik sartuta balego, mezu bat aterako du abisatzeko ("ezin da gehitu jadanik badagoelako").
    - *erreseteatu()* Bezeroen zerrenda hasierako egoeran uzten du, hau da, hutsik. Metodo hau probak egiteko baino ez dugu erabiliko. Bukaeran kendu beharko genuke hau aplikazio erreal bat balitz. Batez ere publikoa denean, ez dugu nahi inork gure bezeroen zerrenda garbi uztea.

- **Eragiketa Klasea**
    - **Atributuak:** identifikadore bat (int), eragiketa dagokion bezeroaren identifikadorea (int), kutxazainean tekleatutako pasahitza (String) eta atera nahi duen diru kopurua (double).
    - **Metodo eraikitzailea:** atributuak hasieratzeko bezainbeste parametro jasoko ditu.
    - **Gainontzeko metodoak:**
        - *eragiketaBurutu()* eragiketaren idBezero duen bezero bat existitzen dela konprobatzeaz arduratuko da. Eta horrela ez balitz, mezu bat aterako luke abisatzen (operazioarenId operazioan,bezeroa ez da existitzen). Gero behin id hori duen bezeroa lortu ostean, eta kopurua egokia baldin bada honen *saldoaEguneratu()* metodoa inbokatuko luke. Kopurua egokia ez baliz (negatiboak ez dira egokiak) mezu bat aterako luke (operazioarenId, operazioan kopurua ez da egokia). ***(kontuz !! bereziak ez diren bezeroei %0.1ko komisioa kobratzen zaie, hau da, 0.0001)***
        - *idBerdinaDu()* id bat parametrotzat hartuta, objektuaren id-arekin konparatuko du, berdinak balira true bueltatuz eta bestela false; *Gogoratu getIdEragiketa() pribatua dela.*

- **<u>ListaEragiketa Klasea</u> (EMA)**
  - **Atributuak:** eragiketen lista duen klasea izango da, kasu honetan ArrayList baten bitartez inplementatua. EMA bat da, klase honen instantzia bakarra dagoelako.
  - **Metodo eraikitzailea:** Eragiketen lista sortzen du. Singleton patroia aplikatuz, eraikitzailea pribatua izango da.
  - **Gainontzeko metodoak:**
    - *getListaEragiketak()* eragiketen lista duen instantzia bakarra bueltatzen du, eta publikoa izango da edonondik atzitua izan dadin.
    - *getIteradorea()* pribatua izango da.
    - *bilatuEragiketaIdz()* parametro bezala pasatzen zaion id-a duen eragiketa bueltatzen du. Eragiketa hori ez bada existitzen, *null* itzuliko du.
    - *gehituEragiketa()* parametro bezala jasotzen duen eragiketa *ListaEragiketak* klasean gehitzen du. Beti, noski, eragiketa hori jadanik lista barruan ez balego, bestela mezu bat aterako du abisatzeko.
    - *erreseteatu()* bezeroen zerrenda hasierako egoeran uzten du, hau da, hutsik. Metodo hau probak egiteko baino ez dugu erabiliko. Bukaeran kendu beharko genuke hau aplikazio erreal bat balitz. Batez ere publikoa denean, ez dugu nahi inork gure bezeroen zerrenda garbi uztea.
    - *eragiketaKopurua()* listan dauden eragiketa kopurua itzuliko du.
    - *eragiketaGuztiakBurutu()* listan dauden eragiketa guztiak burutuko ditu.

OHAR GARRANTZITSUAK HEMENDIK AURRERA WEB-CAT-en ARAZOAK EKIDITZEKO:

- Kontuz *package* lerroarekin. WebCatek ezingo baitu zuen kodea konprobatu. Hortaz, lehenengo lerroa, package laborategi3; komentatu igo aurretik (eclipsen errore gisa markatuko badizue ere)
  //package packlaborategi4;
- Ez programatu ariketaren emaitza lortzeko behar-beharrezkoak baino ez diren getters&setters.
- Metodo batean return bat jartzen duzuenean ez jarri bueltatzen den aldagaia () artean
- Baldintzetan aldagai boolearrak erabiltzen dituzuenean, ez erabili berdinketarik. If (topatu==false)..., sistemak zigortuko zaituzte. Jarri if(!topatua) .... Oinarrizko Programazioan egiten genuen programatzen ikasten geundelako, baina pausu hori gainditua duzue jada.


EragiketenZerrenda
Eragiketa
eragiketa bakoitzeko
eragiketaGuztiakBurutu():void
eragiketaBurutu():void
BankuetxekoKudeatzailea