# PMOBO- 3. laborategia

Aurrebaldintzak

Ikasleak Eclipse ingurunea kontrolatzen du eta Java ikuspegia erabiltzen badaki. Eta, klase sinpleak eta klaseen arteko erlazioak sortzen eta adierazten (UML bidez) badakiela suposatzen da. Eta bukatzeko, badaki proba kasuak (TestCase) eta proba multzoak sortzen ere badakiela JUnit frameworka erabiliz .

Helburuak

Laborategi honetan aurretik lortutako ezagupena bermatzen lagunduko duela, baita objektuei bideratutako paradigman sakontzea ere. Kontzeptu berri bezala interfaze kontzeptua ikusiko da.

Laborategi hau bukatzean, hurrengo gaitasunak lortzea espero da:

- 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.

Motibazioa

Datu Mota Abstraktuak (TAD) eta Egoeren Makina Abstraktua (MAE) edo herentzia erabiltzen hasi baino lehenago, ikusitako guztia finkatzea da laborategi honen helburua. Hasteko, programazio prozedural tradizionala eta objektuei bideratutako programazioaren arteko desberdintasunak azaleratzeko ariketa batzuk planteatzen dira .

## 1 Ataza: IData interfazea

Lehenengo ataza honetan, IData interfazea inplementatzen duen *Data* izeneko klasea eta bertan eginiko metodoen funtzionamendua egokia den aztertzeko DataTest klasea garatu behar da.

```
package packlaborategia3;
public interface IData
{
	public abstract void dataGehitu();
	public abstract void dataKendu();
}
```

1. irudia – *IData* interfazea.

Enuntziatu honekin batera, *Data.java* fitxategian *Data* klasea eskaintzen zaizue, guztiz garatu gabe noski. Fitxategian ikus daitekeen bezala, metodo eraikitzaileak hiru parametro behar ditu, urtea, hilabetea eta eguna (**zehazki ordena honetan**). Metodoak, lehenik data hori egokia den aztertuko du, ez bada, eguneko data sortuko du.

Interfazean agertzen diren, *dataGehitu()* eta *dataKendu()*, dataren hurrengo eguna eta dataren aurreko eguna kalkulatuko dute hurrenez hurren. Printzipioz ariketa hau ezaguna izango da Oinarrizko Programazioan garatutako algoritmo bat delako.

Ataza honetan, data bat egokia den aztertzeko metodo bat garatu beharko duzue baita ere. Data bat egokia dela esaten da baldin eta urtea zenbaki positibo bat bada (0<urtea, alegia), hilabetea 1 eta 12 zenbakien tartekoa bada (hauek barne), eta eguna, 1 eta …

- 31, baldin hilabetea 1,3,5,7,8,10 edo 12 bada.
- 30, baldin hilabetea 4,6,9 edo 11 bada.
- 29, baldin hilabetea 2 eta urtea bisurtea bada.
- 28, baldin hilabetea 2 eta urtea bisurtea ez bada.

Urtea bisurtea da, baldin eta 4ren multiploa bada baina ez 100ena. Honez gain, nahiz eta 100en multiploa izan, 400en multiploak diren urteak ere bisurteak dira.

*Data.java* fitxategian, *toString()* metodoa ere eskaintzen zaizue. Honek, data bat testu modura bihurtzen du. Ez duzue berau aldatu behar. Metodo honen erabilpena, proba kasuak errazteko da, adibide:

***assertEquals(data1.toString(),"5000/01/01");***

## 2 Ataza: IZatikia klasea

Ataza honetan, zatikiekin lan egitea eskatzen da. Bi atributu izango ditu, zenbakitzailea eta izendatzailea. Metodoei dagozkienez, *gehitu*, *kendu*, *biderkatu*, *zatitu* eta zatiki baten *sinplifikazioa*) eta zatikien arteko konparazioak egiteko (*berdinaDa*, *handiagoaDa*, *txikiagoaDa*), izango ditu klase honek.

- Proba kasuak inplementatu.
- *Zatikia* klaseak jarraian agertzen den interfazea inplementatu behar du. IZatiki interfazea eskaintzen zaizue.

```
package packlaborategia3;
public interface IZatikia
{
	public abstract void sinplifikatu();
	public abstract Zatikia gehitu(Zatikia pZatikia);
	public abstract Zatikia kendu(Zatikia pZatikia);
	public abstract Zatikia biderkatu(Zatikia pZatikia);
	public abstract Zatikia zatitu(Zatikia pZatikia);
	public abstract boolean berdinaDa(Zatikia pZatikia);
	public abstract boolean handiagoaDa(Zatikia pZatikia);
	public abstract boolean txikiagoaDa(Zatikia pZatikia);
	 public abstract boolean izendatzaileBerdinaDu(Zatikia pZatikia);
}
```

2. irudia – *IZatikia* interfazea.

*Zatikia* klasearen metodo eraikitzaileak bi parametro beharko ditu, zenbakitzailea eta izendatzailea, itxura hau izanik:

***public** Zatikia(**int** pZenb, **int** pIzen)*

Metodo eraikitzaileak, izendatzaileak 0 balioa ez izatea ziurtatuko du, 0 balioa denean, pantailan *"Ezin daiteke izendatzailearen balioa 0 izan dezakeen zatikirik sortu ..."* mezua igorriko du.

Ataza honetan, *sinplifikatu* metodoak, zatikiaren ikurra zenbakitzaileari esleituko dio. Adibidez, zatikia 4/-6 bada, *sinplifikatu* metodoa aplikatu ostean, -2/3 izan beharko litzateke emaitza. *Gehitu, kendu, biderkatu, zatitu* eta gainontzeko metodoak erabiltzerakoan, *sinplifikatu* metodoa erabili beharko da. Adibidez, 2/-3 eta 2/6 zatikien batuketaren emaitza -1/3 izango litzateke.

Oharra hurrengo metodoa inplementatzea komenigarria da:

private int zkh() {

/** *zatitzaile komunetako handiena topatzeko metodoa***/*

…}

### *Zatikiak sinplifikatu*

*Zatiki baliokide guztiek balio bera adierazten dute. Horretarako, komeni da zatikirik sinpleena erabiltzea; hau da, zenbakitzaile eta izendatzaile txikienak lortu behar ditugu. Zatiki horri* ***zatiki laburtezina*** *deitzen zaio, ezin baita gehiago sinplifikatu.*

*Zatiketaren oinarrizko propietatea erabiliko dugu: hau da, zenbakitzailea eta izendatzailea zenbaki berarekin biderkatu edo zatitu, eta beste zatiki baliokide bat lortuko dugu.*

*Zatiki bat sinplifikatzeko, zenbakitzailearen eta izendatzailearen zatitzaile komuna aurkitu behar dugu, eta horrekin zatitu.*

*Komeni zaigu zatitzaile komunetako handienarekin zatitzea; horrela, pauso bakarrean* ***zatiki laburtezinera*** *iritsiko gara.*

*E****rreferentzia: http://recursostic.educacion.es/descartes***

## ***WebCat-era igotzeko***

- Sei fitxategiak igo behar dituzue:
  - Data, IData, TestData
  - Zatikia, IZatikia, TestZatikia

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 packlaborategi3;`
- Ez programatu ariketaren emaitza lortzeko behar-beharrezkoak baino ez diren getters&setters.
- Metodo batetan 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.