19950904 Espresioak II

`Espresioak` paketea eta `Espresioak_Proba` prozedura emanik, erantzun ondorengo galderei:

(1) Zein dira `Eragiketa_Bitar` motaren eragiketa primitiboak? Zergatik?

(2) Zein dira *dispatching* eragiketak adibide honetan? Zergatik deitzen zaie horrela?

(3) Zer espresio aritmetiko errepresentatzen du `Espresioak_Proba` prozeduran erazagutu eta hasieratutako `Zuhaitza` izeneko objektuak?

(4) Zer inprimatuko du zehatz-mehatz `Espresioak_Proba` prozedurak?

(5) Ba al dago inon *dispatching*-deirik? Baiezkoan, adieraz ezazu zein(tzuk) d(ir)en, enuntziatuan bertan argi eta garbi identifikatuz. Zergatik d(ir)a *dispatching*-deia(k)?

```
package Espresioak is
    type Espresio is tagged null record;
    procedure Put (E : in Espresio);
    procedure Get (E : out Espresio);
    type Literal is new Espresio with
        record
            Balioa : Integer;
        end record;
    procedure Put (L : in Literal);
    procedure Get (L : out Literal);
    type Espresio_Erakusle is access Espresio'Class;
    type Eragiketa_Bitar is new Espresio with
        record
            Ezkerra, Eskuina : Espresio_Erakusle;
        end record;
    procedure Put (O : in Eragiketa_Bitar);
    procedure Get (O : out Eragiketa_Bitar);
    type Batura is new Eragiketa_Bitar with null record;
    procedure Put (S : in Batura);
    procedure Get (S : out Batura);
    type Kendura is new Eragiketa_Bitar with null record;
    procedure Put (S : in Kendura);
    procedure Get (S : out Kendura);
end Espresioak;

with Ada.Text_IO;
package body Espresioak is
    procedure Put (E : in Espresio) is
    begin
        null;
    end Put;
    procedure Put (L : in Literal) is
        package Osoak_SI is new Ada.Text_IO.Integer_IO (Integer);
    begin
        Osoak_SI.Put (L.Balioa, Width => 1);
    end Put;
    procedure Put (O : in Eragiketa_Bitar) is
    begin
        Ada.Text_IO.Put ("(");
        Put (O.Ezkerra.all);
        Ada.Text_IO.Put (") ");
        Ada.Text_IO.Put ("(");
        Put (O.Eskuina.all);
        Ada.Text_IO.Put (")");
    end Put;
    procedure Put (S : in Batura) is
    begin
        Ada.Text_IO.Put ("+ ");
        Put (Eragiketa_Bitar (S));
    end Put;
    procedure Put (S : in Kendura) is
    begin
        Ada.Text_IO.Put ("- ");
        Put (Eragiketa_Bitar (S));
    end Put;
    procedure Get (E : out Espresio) is separate;
    procedure Get (L : out Literal) is separate;
    procedure Get (O : out Eragiketa_Bitar) is separate;
    procedure Get (S : out Batura) is separate;
    procedure Get (S : out Kendura) is separate;
end Espresioak;
--------------------------------
with Espresioak;
procedure Espresioak_Proba is
   Zuhaitza : Espresioak.Espresio_Erakusle :=
     new Espresioak.Batura'
        (Ezkerra => new Espresioak.Batura'
           (Ezkerra => new Espresioak.Literal'(Balioa => 5),
            Eskuina => new Espresioak.Literal'(Balioa => 0)),
         Eskuina => new Espresioak.Kendura'
           (Ezkerra => new Espresioak.Literal'(Balioa => 13),
            Eskuina => new Espresioak.Literal'(Balioa => 7)));

    procedure Inprimatu (E : in Espresioak.Espresio_Erakusle) is
    begin
        Espresioak.Put (E.all);
    end Inprimatu;
begin
    Inprimatu (Zuhaitza);
end Espresioak_Proba;
```

19970603 Analisi gramatikalak

19970602 ariketako `Analisi` mota hedatu nahi dugu orain. Identifikadorea, erroa eta kategoriaz gain, zenbait kasutan analisiaren beste alderdi batzuk erregistratu nahi dira, analizatutako hitzaren kategoriaren arabera:

Izenak eta adjektiboak: numeroa eta kasua.

Aditzak: modua, aldia, pertsona eta numeroa.

Izenordainak: pertsona eta numeroa.

Gainerako kategoriak: bestelako informaziorik ez.

Egin Adaz mota-hierarkia bat mota etiketatuak erabiliz, informazioa modurik hoberenean konpartitzeko aukera emango duena. Erazagutu motak hainbat paketetan, informazio-ezkutatzearen printzipioa egokiro aplikatuz.

`Inprimatu` izeneko eragiketa polimorfikoa (*dispatching*-ekoa) espezifikatu, hierarkiako mota guztietarako.

Inplementatu `Aditz_Analisi` motari (identifikadorea, erroa eta kategoriaz gain modua, aldia, pertsona eta numeroa dituena) **dagokion `Inprimatu`-ren bertsioa**, horretarako hierarkian `Aditz_Analisi` motaren gaineko motei dagozkien `Inprimatu`-ren bertsioen espezifikazioez baliatuz.

20020602 Soldatako kontzeptuak

Enpresa batean, langileen soldatak kudeatzeko aplikazio bat egin nahi dute. Soldata batean hainbat kontzeptu bereizten dira, eta kontzeptu bakoitzak ezaugarri desberdinak ditu. Kontzeptu guztien ezaugarria da zenbatekoa (kopuru gordina). Kontzeptu batzuen gainean ez da zergarik aplikatzen, baina beste batzuen gainean Foru Ogasunak bere zergak edo atxikipenak ezartzen ditu. Langile baten soldata kalkulatzea, soldatari dagozkion kontzeptu guztien zenbateko garbiak (hau da, atxikipenak-eta, dagozkionean, kenduta) batzea baino ez da.

Zergarik gabeko kontzeptuak hainbat lani egokitu lekizkioke, eta, horregatik, beren ezaugarri dira lanaren deskribapena eta ordu kopurua.

Zergadun kontzeptuen kasuan berriz, Ogasunak ehuneko bat hartzen du atxikipen gisa. Kontzeptu hauek honela sailkatzen dira: oinarrizko soldata, osagarriak eta aparteko orduak (estrak):

- Langile bakoitzaren oinarrizko soldata zenbakizko kode batez identifikatzen da.
- Soldataren osagarriek, aipatutako atxikipen-ehunekoaz gain, zerga berezi bat daukate. Honako osagarri mota hauek hartzen dira kontuan: lanpostu-mailako osagarria, kargu-osagarria eta antzinakotasun-osagarria. Lanpostu-mailako osagarri baten ezaugarri hiria eta sukurtsala ditugu; kargu-osagarriarena, kargua (burua, zuzendaritzakoa edo enplegatua), eta, antzinakotasun-osagarriarena, urte kopurua.
- Ordu estren kontzeptuaren ezaugarri dira zenbat ordu sartu diren, eta zergatia (proiektua bukatu beharra, zaintza-lana edo bestelakoak). Aparteko ordu horien berezitasuna dela eta, Ogasunak portzentaia berezia aplikatzen die.

Honako hau eskatzen da:

1) Deskribatutako informazioa errepresentatzeko **klase-hierarkia diseinatu** (marraztu), objektuei zuzendutako programazioaren teknikak (informazio-konpartitzea, etab.) ahal den hobekien aprobetxatuz. Adierazi diagraman, **mota bakoitzari dagozkion atributuak**, **eragiketak** eta **abstraktuak diren klaseak** (lerro etenak erabiliz), halakorik baldin bada.

2) **Inplementa ezazu** langile baten **soldata osoa kalkulatzen duen `Soldata_Kalkulatu` eragiketa**, ondoko erazagupen hauek kontuan hartuta:

```
with Kontzeptuak;
package Langileak is
   type Langile is limited private;
   -- ...
   procedure Soldata_Kalkulatu (L : in Langile; Soldata : out Float);
   -- ...
private
   type Kontzeptu_Erakusle is access Kontzeptuak.Kontzeptu'Class;
   type Kontzeptu_Bektore is array (1 .. 20) of Kontzeptu_Erakusle;
   type Nominako_Kontzeptuak is record
         Zenbat_Kontzeptu : Natural;
         Kontzeptu_Taula : Kontzeptu_Bektore;
      end record;
   type Langile is record
         Izena : String (1 .. 50);
         Kontzeptuak : Nominako_Kontzeptuak;
      end record;
end Langileak;
```

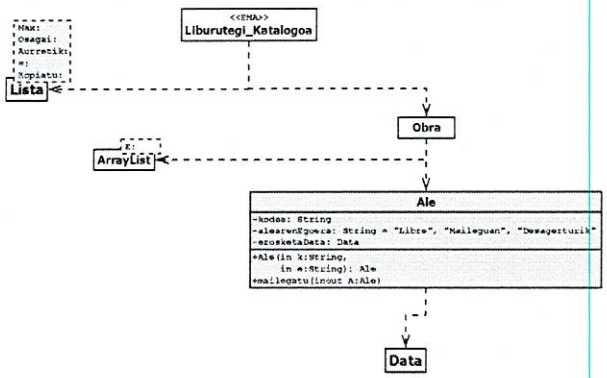
Oharra: erabilitako eragiketa guztiak erazagutu behar dira, nongoak diren zehaztuz.

PROGRAMAZIO MODULARRA ETA OBJEKTU ORIENTAZIOA

Azterketa globala

Izen-deiturak: .....

Liburutegi bat kudeatzeko aplikazioa egin nahi da. Aplikazio horrek hainbat modulu izango ditu, DMA eta EMA-ak, honako eskema honen arabera[1]:



Azterketa honetako ariketak modulu horien gainekoak dira, Adaz lantzekoak batzuk eta Javaz besteak. Ebaluatzerakoan, honako hauek baloratuko dira, besteak beste: objektuen pribatutasun maila eta datu-egitura egokiak aukeratzea, instantziazioen zuzentasuna, eragiketen zehaztapenak, salbuespenen tratamendu egokia eta kodearen zuzentasuna, oro har.

Oharra.- Idatz itzazu beharrezkoak iruditzen zaizkizun eragiketa guztien erazagupenak, argi eta garbi zehaztuz non egin beharko liratekeen. Halaber, adieraz ezazu zure kodearen zein atalek erantzuten dion ariketa hauen ataletako bakoitzari (atalen letrak erabiliz, adibidez).

1. ariketa (3 puntu). Lengoaia: Java.

Liburutegiko obren mota `Obra` DMA-aren bidez inplementatzen da, eta ezaugarri hauek ditu: kodea (stringa), izenburua (stringa), autorea (stringa), maileguan har daitekeen ala ez (boolearra), eta obra horren aleen zerrenda. Ale-zerrenda errepresentatzeko `ArrayList` klase generikoa erabili behar da; hona hemen klase horren zehaztapen partziala:

```
public class ArrayList<E> ... {
   public E get(int index) throws IndexOutOfBoundsException;
   public boolean add(E e);
   public void add(int index, E element)
```

[1] Diagrama honetako identifikadoreen estiloak ez dira homogeneoak: batzuek Ada estiloa jarraitzen dute eta beste batzuek Java estilon, azterketan eskatzen diren ariketetan erabili beharreko lengoaiaren arabera.


```
      throws IndexOutOfBoundsException;
   public int size();
   public E remove(int index)
      throws IndexOutOfBoundsException;
   public void clear();
   public boolean isEmpty();
   ...
}
```

Idatzi `Obra` klasea (*Obra.java*), ondoko ataletan eskatzen dena kontuan hartuz.

A) **Idatzi `Obra` motarako aukeratutako errepresentazioa (atributuak)**, emandako deskribapenaren arabera. Kontuan hartu eskura duzula `Ale` DMA-a (ikus diagrama).

B) **Idatz itzazu honako metodo hauen goiburukoak** (ez da beharrezkoa inplementatzea, aski da `...` idaztea gorputzaren ordez):

- **Eraikitzaile bat**, honako lau parametro hauekin: kodea (stringa), izenburua (stringa), autorea (stringa) eta maileguan har daitekeen ala ez (boolearra).
- **`getKodea`** kontsulta-eragiketa: obraren kodea itzultzen du.
- **`lehenAleaMailegatu`**: `"Libre"` egoeran dagoen lehen alearen egoera aldatzen du, `"Maileguan"` utziz. Metodo honek `AleakAgorturik` salbuespena jaurtiko du libre dagoen alerik ez baldin badago, eta `EzinDaMailegatu` salbuespena obra mailegaezina baldin bada.

C) `Obra` klaseko metodo batzuek jaurtitzen ahal dituzte `AleakAgorturik` eta `EzinDaMailegatu` salbuespenak. **Erazagut ezazu salbuespen horietako bat (edozein)**, argi eta garbi azalduz non egin beharko litzatekeen.

D) **Inplementa ezazu goian aipatutako eraikitzailea.** Eraikitzaile honek, klaseko eremuei balioak esleitzeaz gain, `Ale` klasearen instantzia bat sortu behar du. Ale horren kodea obrarena bera izango da, `"01"` karaktereez hedatua; alearen egoera, `"Libre"` izango da. Sortutako alea obraren ale-zerrendan gehitu beharko da, jakina.

E) **Inplementa ezazu `lehenAleaMailegatu` metodoa**, goiko deskribapenaren arabera. Jaurtiaraz edo trata itzazu salbuespenak, hala behar denean.

2. ariketa (4 puntu). Lengoaia: Ada.

Suposatu eskura dituzula `Obrak` eta `Aleak` Ada paketeak, aurreko ariketako `Obra` eta `Ale` DMA-ak inplementatzen dituztenak.

`Liburutegi_Katalogoa` EMA-ak, berriz, liburutegiko obra guztien bilduma errepresentatzen du, non obrak zerrenda batean ordenatuta gordetzen baitira, obra-kodearen arabera gorantz. Obra-zerrenda errepresentatzeko `Lista_Estatiko_Generikoak` pakete generikoko `Lista` motaz baliatuko gara. Hona hemen pakete horren zehaztapena:

```
generic
   Max : in Positive;
   type Osagai is limited private; -- listako osagaien mota
   with function Aurretik (E1, E2 : in Osagai) return Boolean;
   with function "=" (E1, E2 : in Osagai) return Boolean;
   with procedure Osagaia_Kopiatu (E1 : out Osagai; E2 : in Osagai);
package Lista_Estatiko_Generikoak is

   type Lista is limited private; -- listaren mota
```

```
   procedure Hutsa_Sortu (L : out Lista);
   function Osagai_Kopurua (L : in Lista) return Natural;
   procedure Kokatu (L : in out Lista; E : in Osagai);
   -- Irteera: L eguneratua, non E kokaku baita dagokion lekuan,
   --          Aurretik funtzioaren arabera
   -- Salbuespenak: Lista_Beterik, L-ren osagai kopurua = Max
   --               baldin bada
   procedure Lehena_Lortu (L : in Lista; E : out Osagai);
   -- Salbuespenak: Lista_Hutsik, L-ren osagai kopurua = 0
   --               baldin bada
   procedure Lehena_Kendu (L : in out Lista);
   -- Irteera: L eguneratua, non lehen osagaia kendu baita
   -- Salbuespenak: Lista_Hutsik, L-ren osagai kopurua = 0
   --               baldin bada
   function "=" (L1, L2 : in Lista) return Boolean;
   procedure Kopiatu (L1 : out Lista; L2 : in Lista);
   --...

   Lista_Hutsik, Lista_Beterik : exception;

private
   -- inplementazio estatikoa
   -- ...
end Lista_Estatiko_Generikoak;
```

Idatz itzazu `Liburutegi_Katalogoa` paketearen espezifikazioa eta gorputza, ondoko ataletan eskatzen dena dagokion lekuan ipiniz.

A) **Idatzi, ondoko ataletan eskatzen dena kontuan hartuz, liburutegiko katalogoa errepresentatzeko behar diren egoera-aldagaien (objektu kapsulatuaren) erazagupena.**

B) **Erazagut itzazu honako eragiketa hauek** (ez dira inplementatu behar, aski da goiburukoa idaztea):
   - **`Kargatu`**: Obrak kargatzen ditu zerrendan, informazioa `Obren_Fitx_Izena` konstanteak erreferentziatzen duen fitxategitik[2] irakurriz.
   - **`Obra_Ordeztu`**: Obra bat emanik, kode berekoaren ordez jartzen du zerrendan.
   - **`Obraren_Alea_Mailegatu`**: Obra-kode bat emanik, obra horren lehen alea mailegatzen du (ikus 1. ariketa). Eragiketa honek `Obra_Hori_Ez_Dago` salbuespena altxatuko du, emandako kodea duen obrarik ez baldin badago, eta `Obra_Horren_Alerik_Ez_Da_Geratzen`, obrak ez baldin badu mailegatzeko moduko (hots, libre dagoen) alerik.

C) **Erazagut itzazu** `Obra_Hori_Ez_Dago` eta `Obra_Horren_Alerik_Ez_Da_Geratzen` salbuespenak.

D) **Inplementa ezazu `Obraren_Alea_Mailegatu` eragiketa**, goian emandako deskribapenaren arabera. Altxaraz edo trata itzazu salbuespenak, hala behar denean.

<u>3. ariketa</u> (3 puntu). Lengoaiak: Java eta Ada.

Liburutegia batean hainbat obra mota biltzen dira: liburuak, DVDak, aldizkariak, eta hainbat liburukiz osaturiko obrak, hala nola entziklopediak eta hiztegiak. Obra mota guztien atributu dira kodea, izenburua eta autorea. Atributu komun horietaz gain, liburuen kasuan gaia (artea, historia, zientzia-fikzioa eta abar), argitaletxea eta argitalpen-data gorde behar dira; DVDen kasuan, interpretatzailea eta ekoizlea; entziklopedia eta hiztegien

[2] *obrak.txt.*

kasuan, argitaletxea, argitalpen-data eta liburuki kopurua; eta aldizkarien kasuan, argitaletxea eta argitalpen-data. Atributu horiek guztiak stringak dira, liburuki kopurua izan ezik, zein zenbakizkoa baita.

Hori dena klase-hierarkia baten bitartez errepresentatu nahi dugu. Klase guztietan behar da, ohiko eraikitzaile, eta kontsultarako eta aldaketarako eragiketez gain, inprimatzeko eragiketa bat. Inprimatzeko eragiketa horrek obraren ezaugarriak bistaratuko ditu.

Obra batzuk, gainera, baliabide digitalak ere badira (gaur egun liburuak eta aldizkariak baino ez). Baliabide digitalok formatua (stringa) zein den jakiteko eta obra deskargatzeko eragiketak behar dituzte.

Honako hau eskatzen da:

A) **Marraztu deskribatutako obra motak hobekien errepresentatuko dituen klase-hierarkia**, non klase abstraktuak, konkretuak eta interfazeak ageriko baitira. **Marraztu klaseak eta kokatu atributuak berorietan** (ez da beharrezkoa irudian mota zehaztea), **informazioaren konpartitzea ahalik eta modurik egokienean gauzatu dadin.** Klase abstraktuak eta interfazeak lerro etenez marraztu behar dira. **Koka itzazu eraikitzaileak, eta `inprimatu`, `deskargatu` eta `formatua` metodoak** (ez da beharrezkoa irudian parametroak zehaztea), **berezkoak zein birdefinituak, dagozkien klaseetan SOILIK**.

B) **Idatz ezazu:**

   1. **Javaz: `Liburu` klasean amaitzen den hierarkiaren adarra inplementatzeko klase abstraktu, konkretu eta interfazeak.** Erazagut itzazu klase horietan beharrezko diren eragiketa guztiak: horien artean, eraikitzaile parametrodun bat eta kontsulta-eragiketa edo *getter* bat. Ez da beharrezkoa ezer inplementatzea, aski da `{...}` idaztea gorputzaren ordez.

   2. **Adaz: `DVD` klasean amaitzen den hierarkiaren adarra.** Motak definitu baino ez da egin behar.

C) **Inplementatu Javaz `inprimatu` metodoa, `Hiztegi` klasean amaitzen den adarreko klaseetan** (behar denean soilik, jakina).

D) **Inplementa ezazu Javaz `LiburutegiaProbatu` klaseko `main` metodoa** (programa nagusia). Gehi itzazu liburu bat eta DVD bat `ArrayList` batean, eta korritu gero `ArrayList` hori begizta batez, bertako obren ezaugarriak inprimatuz, eta, bertsio digitala duten obren kasuan, baita formatua ere[3]. **Identifikatu eta azal itzazu programan erabiltzen diren *dispatching* dinamikoko deiak.**

[3] Instantzia (objektu) bat klase edo interfaze batena den jakiteko, erabili `instanceOf` eragilea. Honela, adibidez:

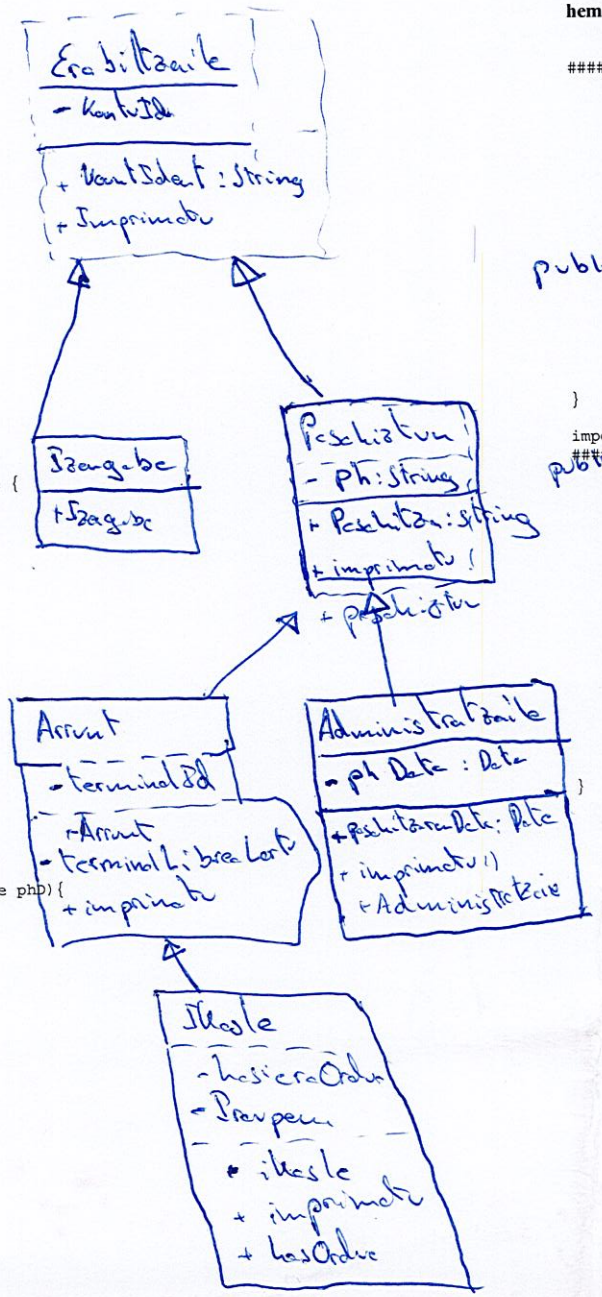
```
if (e instanceof IBidalgarri) {...}
```

Objektuei Orientatutako Programazioa (Java)

20100604 Erabiltzaile motak.

Honako Java klase hauek emanda:

```
public abstract class Erabiltzaile {
   private String kontuId;

   public Erabiltzaile(String id){
      kontuId = id;
   }

   public String kontuIdent(){
      return kontuId;
   }

   public void inprimatu(){
      System.out.println(kontuId);
   }
}

public class Izengabe extends Erabiltzaile {

   public Izengabe(){
      super("Izengabea");
   }
}

public abstract class Pasahiztun extends Erabiltzaile {
   private String ph;

   public Pasahiztun(String id, String pHitza){
      super(id);
      ph = pHitza;
   }

   public final String pasahitza(){
      return ph;
   }

   public void inprimatu(){
      super.inprimatu();
      System.out.println(ph);
   }
}

import java.util.Date;
public class Administratzaile extends Pasahiztun {
   private Date phData;

   public Administratzaile(String id, String ph, Date phD){
      super(id, ph);
      phData = phD;
   }

   public final Date pasahitzarenData (){
      return phData;
   }

   public void inprimatu (){
      super.inprimatu();
      System.out.println(phData);
   }
}
```



1) **Marraztu** erabiltzaile mota hauek errepresentatzen dituen **klase-hierarkia**, klase bakoitzari dagozkion **atributuak eta eragiketak** adieraziz.

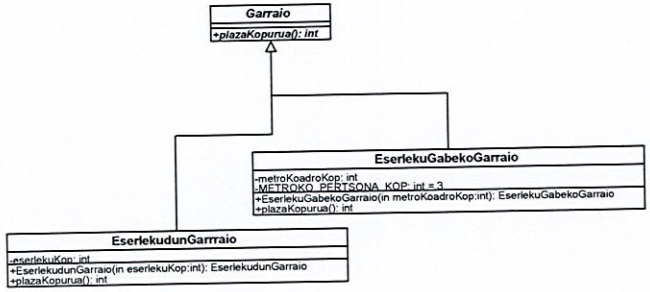
2) Behar desberdinei aurre egiteko, ordea, diseinua aldatu beharrean gaude. Honela, `Arrunt` klase abstraktuak `Pasahiztun` klasea hedatuko du. `Ikasle` eta `Ikertzaile` klase konkretuek, era berean, `Arrunt` klasea hedatuko dute. **Osatu Java klaseotan batetik hemezortzira zenbakiturik dauden hutsuneak**, zenbakiok erantzunean aipatuz.

```
##### 1 ##### class Arrunt ##### 2 ##### {
    ##### 3 ##### String terminalId;
    //klasearen eragiketa lokala (barne-eragiketa); libre
    //dagoen lehen terminalaren identifikadorea itzuliko du.
    ##### 4 ###### String terminalLibreaLortu() {...}

    //eragiketa eraikitzailea (libre dagoen
    //lehen terminalaren esleipena ere egiten du,
    //terminalId aldagaian
    ##### 5 ##### Arrunt(String ident, String ph) {
       ###### 6 ######
    }
    ##### 7 ##### void inprimatu() {
       ##### 8 #####;
    }
}

import java.util.Date;
##### 9 ##### class Ikasle ##### 10 ##### {
    ##### 11 ##### Date hasieraOrdua;
    ##### 12 ##### int IRAUPENA = 3; //konstantea

    //eragiketa eraikitzailea
    ##### 13 ##### Ikasle(String i, String ph, Date hData){
       ###### 14 ######
    }
    ##### 15 ##### void inprimatu () {
       ##### 16 #####;
    }
    ##### 17 ##### Date hasOrdua () {
       ##### 18 #####;
    }

    //...
```

20080604 Garraiobideak

Eusko Jaurlaritzak ebakuazio-plan bat egin nahi du, hondamendi naturalak gertatzen direnetarako. Horretarako, dauzkan garraioetan zenbat pertsona eraman litezkeen kalkulatu nahi du.

Garraioak bi motatakoak dira: eserlekuak dituztenak eta ez dituztenak. Eserlekuak dituztenen plaza kopurua eserleku kopuruaren adinakoa da, noski. Eserlekurik ez dutenetan sartzen den pertsona kopurua, berriz, garraioaren metro koadro erabilgarriak kontuan hartuz kalkulatzen da (3 pertsona sartzen dira metro koadroko). Hona hemen klase-diagrama (letra etzanak klase eta eragiketa abstraktuak direla adierazten du):



Honako hau eskatzen da, Java programazio-lengoaia erabiliz:

1. **Inplementatu** `Eserlekugabeko_Garraio` klasean amaitzen den adarreko klase guztiak.

2. **Inplementatu** `Plaza_Kopuru_Totala` **eragiketa,** garraioen zerrenda bat emanik, plaza kopuru totala kalkulatzen duena. `Vector` klasea erabil dezakezu horretarako

2)

public EserlekuGabekoGarraio EserlekuGabekoGarraio(int metroKoadroKop){}