Lliurament 1

1. Objectius

Organitzar imatges dins una biblioteca. Cada imatge quedarà representada pel seu fitxer, guardant la següent informació:

- Camí on es troba el fitxer (ex. /home/prog2/fitxers/)
- Nom del fitxer (sense extensió en cas que la tingui)
- Extensió (ex. jpg, bmp, png, ...)
- Data última modificació
- Nom de la imatge (pot ser diferent del nom del fitxer)

Per tant, la biblioteca d'imatges en realitat serà una llista de fitxers. Per tal d'interactuar amb la llista, caldrà definir un menú amb les opcions següents:

1. Afegir Imatge: Demanarà les dades d'un fitxer i l'afegirà a la llista de fitxers.
2. Eliminar imatge: Elimina de la llista el fitxer corresponent a una posició donada.
3. Mostrar llista: Mostra el contingut de la llista de fitxers, mostrant davant de cada fitxer, el nombre de la seva posició a la llista començant per 1.
4. Guardar llista: Guarda el contingut de la llista en un fitxer.
5. Recuperar llista: Carrega una llista prèviament guardada d'un fitxer.
6. Sortir: Surt de l'aplicació.

2. Material pel lliurament

Per aquest lliurament us proporcionem una llibreria **UtilsProg2.jar** que conté les classes:

- **ImageFile (abstract class)**
- **InImageList (interface)**
- **ImageList (abstract class)**
- **Menu**

Haureu d'utilitzar aquestes classes en el desenvolupament del lliurament. Podeu trobar aquesta llibreria al Campus Virtual i afegir-la al vostre projecte.

3. Descripció del lliurament

A continuació us anirem plantejant els diferents passos per resoldre la pràctica proposada. Us recomanem que seguiu aquests passos.

3.1 Creació del projecte

El primer pas serà crear un projecte, al qual li heu de posar com a nom ***Cognom1Cognom2Nom***, tenint en compte les següents consideracions:

- La primera lletra de cada part en majúscula i la resta en minúscula.
- Eviteu utilitzar accents i caràcters estranys (ñ o ç).
- La classe principal s'ha de dir ***GestioVisorUB***, i el paquet per defecte ***edu.ub.prog2.Congom1Cognom2Nom.vista***. En el nom del paquet utilitzeu els mateixos criteris anteriors. En el Netbeans s'ha d'indicar la classe principal com ***edu.ub.prog2.Congom1Cognom2Nom.vista.GestioVisorUB***.

Per exemple, una estudiant amb nom Dolça Martínez Castaña, hauria de crear un projecte amb el nom *MartinezCastanaDolca*, i la seva classe principal hauria de ser ***edu.ub.prog2.MartinezCastanaDolca.vista.VisorUB1***.

3.2 Classe *GestioVisorUB*

La classe de la vista ***GestioVisorUB*** tindrà una mètode estàtic main on s'ha de crear un objecte de tipus ***VisorUB1*** anomenat "vista"***.***

3.3 Classe *VisorUB1*

3.3.1 Mètodes principals:

Crearem un mètode d'objecte ***gestioVisorUB*** on s'implementarà el menú de l'aplicació. Aquest mètode ha de tenir la següent signatura:

```
public void gestioVisorUB();
```

Un cop creat, aquest mètode és cridarà des de el mètode main de la classe ***GestioVisorUB***:

```
vista.gestioVisorUB();
```

3.3.2 Implementació del *menú* d'opcions i la lògica del programa

Un cop tenim el projecte i la classe principal, definirem la lògica del programa, o sigui, com es comportarà el programa dins del mètode **gestioVisorUB**. Donat que encara no hem definit res, de moment només es mostrarà un missatge per cada opció del menú, indicant quina opció s'ha triat, excepte en el cas de l'opció de sortir, que finalitzarà l'aplicació.

Per fer aquest punt, disposeu de la classe ***Menu*** dins la llibreria. Aquesta classe gestiona un menú genèric, permetent mostrar la llista d'opcions i controlar la selecció d'una opció per part de l'usuari. Teniu un exemple de com utilitzar aquesta

classe al campus virtual. És un exemple simple d'utilització de la classe**, no us val utilitzar-lo tal qual**, cal adaptar-lo a les necessitats tenint en compte els criteris de modularitat de la pràctica.

3.4 Creació de les classes per emmagatzemar les dades dels fitxers d'imatge

Per fer aquest lliurament, utilitzarem el mecanisme d'herència, que veureu a fons a les classes de teoria. De moment no us heu de preocupar pel què implica aquest fet, us donarem els detalls per a que el pugueu utilitzar de forma pràctica.

A la llibreria que us hem proporcionat trobareu les classes ***ImageList*** i ***ImageFile***, que contenen la definició dels mètodes bàsics que tindran una llista d'imatges i un fitxer d'imatge respectivament. Les classes que utilitzeu per guardar les dades han d'anar dins el [paquet Model] (***edu.ub.prog2.Congom1Cognom2Nom.model***).

2.3.1 Classe Imatge

El primer pas serà implementar una classe ***Imatge*** on guardarem la informació d'un fitxer (amb la informació indicada a l'enunciat). Aquesta classe heretarà de la classe ***ImageFile***, i haureu de definir-ne els mètodes d'accés a la informació. Per fer l'herència, utilitzareu el següent codi:

```
public class Imatge extends ImageFile {

}
```

El mateix Netbeans us informarà de que hi ha mètodes abstractes sense definir, i us proposarà de crear-los ell mateix. Els podeu crear així o definint-los vosaltres mateixos, però en tot cas els mètodes de la classe que cal definir són **getLastModification**, ***getFullPath*** i ***getExtension***, que retornen la data de l'última modificació, el camí al fitxer i la seva extensió, respectivament. La signatura d'aquests mètodes ha de ser:

```
public Date getLastModification()

public String getFullPath();

public String getExtension() ;
```

Tingueu en compte que la classe ***ImageFile*** hereta de la classe File de Java. Per tant, podeu fer servir els mètodes de la classe File per implementar aquests mètodes. Penseu bé, a més, quins seran els atributs de la classe **Imatge** en aquest cas.

2.3.2 Classe TaulaImatges

Un cop tenim la classe necessària per guardar la informació d'una imatge, ara és el torn de definir la classe que representarà una llista d'imatges. De la mateixa

manera que en el cas anterior, us donem una classe base, de la qual haureu de partir per implementar la vostra classe. A continuació es mostra com heu de declarar la classe:

```
public class TaulaImatges implements InImageList {

}
```

Igual que en el cas de la classe Imatge, aquí caldrà que implementeu una sèrie de mètodes, que ja venen definits a la interfície base InImageList:

```
public int getSize();

public void addImage(ImageFile imageFile) ;

public void removeImage(ImageFile imageFile) ;

public  ImageFile getAt(int position) ;

públic void clear();

public boolean isFull();
```

Aquests mètodes serviran per:

- **getSize:** Retorna el nombre d'elements que hi ha a la llista.
- **addImage:** Afegeix una nova imatge a la llista.
- **removeImage:** Elimina una imatge de la llista si coincideix amb la imatge passada. Caldrà definir el mètode ***equals*** per a la vostra classe ***Imatge.***
- **getAt:** Retorna la imatge a la posició indicada de la llista.
- **clear:** Elimina tots els elements de la llista.
- **isFull:** Indica si la llista està plena o per el contrari podem afegir més elements.

Per implementar la llista de fitxers heu d'utilitzar les classes vistes a **Programació 1** (Taules, tuples, ...).

Aquesta llista ha de tenir una capacitat màxima de 100 imatges, o se li pot donar la mida màxima com a paràmetre en la construcció de la taula.

Per últim, no oblideu implementar els mètodes toString per mostrar el resum de la llista de fitxers en forma d'String. Aquest resum s'haurà de mostrar de la següent manera:

```
Llista Fitxers
==============

[1]     Imatge{nom= Foto de Carmen, data=Thu Jan 08 12:34:02 CET 2015, nom fitxer=carmen, ext=jpg, cami complet=F:\carmen.jpg}

[2]     Imatge{nom=Deafult name, data=Thu Jan 08 12:35:02 CET 2015, nom fitxer=blue, ext=jpg, cami complet=F:\blue.jpg}
```

3.5 Gestió de les dades d'entrada

Un cop definides les classes que permetran guardar les imatges i les llistes d'imatges, ja podeu començar a donar funcionalitat a les opcions 1, 2 i 3 del menú. En tot moment tingueu en compte que es valorarà la modularitat del vostre codi, la reutilització i el seguiment de les bones pràctiques de programació. A més a més, sempre que pugueu utilitzeu els mètodes estàndards dels objectes (toString, equals, ...).

3.6 Persistència de dades

Arribats a aquest punt ja hauríeu de poder demanar les dades d'una imatge a l'usuari i guardar-les a una llista. També poder mostrar i eliminar elements de la llista.

Ara el què volem és poder guardar les dades a un fitxer i poder-les carregar posteriorment. Implementeu les opcions 4 i 5 del menú.

Per fer aquesta opció, necessiteu seguir els següents passos:

1. Obtenir la ruta al fitxer on voleu guardar les dades o des del qual voleu carregar-les. Necessitareu guardar aquesta informació en un objecte de tipus File. Per exemple, si volem utilitzar el fitxer "dades.dat", farem:

   ```
   File fitxer=new File("dades.dat");
   ```

2. L'accés de lectura i escriptura a un fitxer es fa mitjançant Streams. Per llegir d'un fitxer utilitzarem un objecte de tipus FileInputStream, i per escriure a un fitxer utilitzarem un objecte FileOutputStream:

   ```
   FileInputStream fin=new FileInputStream(fitxer);

   FileOutputStream fout= new FileOutputStream(fitxer);
   ```

3. Finalment, existeixen objectes per gestionar la lectura i escriptura d'un objecte a un Stream. Per escriure un objecte utilitzarem un objecte de tipus

ObjectOutputStream, mentre que per llegir-lo utilitzarem un objecte de tipus ObjectInputStream:

```
ObjectOutputStream oos = new ObjectOutputStream(fout);

ObjectInputStream ois = new ObjectInputStream(fin);
```

Teniu més informació sobre Streams en el document d'ajuda al lliurament 1. No oblideu tancar correctament els objectes d'accés als fitxers. Tenint en compte que en aquest lliurament només tenim una llista de fitxers, podeu guardar directament aquesta llista d'imatges.

3.7 Implementar una llista d'imatges utilitzant una implementació estàndard

En els passos anteriors heu implementat tota la funcionalitat d'aquest lliurament, utilitzant, bàsicament, els coneixements adquirits a Programació 1. En aquest punt, es demana utilitzar una implementació estàndard de Java de taules per a la llista d'imatges. Implementa la classe ***LlistaImatges*** que hereti de la classe ***ImageList***, utilitzant com a implementació de la llista la classe ***ArrayList*** de Java.

Per fer-ho a continuació es mostra com heu de declarar la classe:

```
public class LlistaImatges extends ImageList {

}
```

Igual que en el cas de la classe TaulaImatges, aquí caldrà que implementeu els mètodes, que ja venen definits a la classe base ImageList:

```
public int getSize();

public void addImage(ImageFile imageFile) ;

public void removeImage(ImageFile imageFile) ;

public  ImageFile getAt(int position) ;

públic void clear();

public boolean isFull();
```

Aquesta llista també ha de tenir una capacitat màxima, que pot estar fixada a 100 o per un paràmetre en la construcció de la llista.

3.8 Donar l'opció de seleccionar una de les implementacions

Tingueu en compte que caldrà que es pugui utilitzar qualsevol de les dues implementacions de la llista d'imatges. A la memòria haureu d'explicar quins canvis heu hagut de fer al vostre programa per a que funcioni amb la nova implementació i què cal fer per triar una implementació o l'altra.

4. Ajuda pel lliurament

Utilitzeu la classe **File** de Java per implementar la classe **Imatge.**

Utilització de la llibreria **UtilsProg2.jar**:

- Podeu instanciar objectes de la classe **Menu** a la vostra classe vista per implementar la gestió del menú del visor. Teniu un exemple de com utilitzar la classe **Menu** al Campus Virtual.
- Les classes **ImageFile i ImageList** són classes abstractes que heu d'utilitzar per implementar les vostres classes **Imatge**, **LlistaImatges** i **TaulaImatges**.

5. Format del lliurament

El lliurament consistirà en tot el codi generat en els diferents punts de l'enunciat, juntament amb la documentació especificada en aquest apartat.

En concret, cal generar un fitxer amb el nom:

Cognom1Cognom2Nom_L1.tar.gz

Que conté:

- el projecte sencer de Netbeans
- la memòria del lliurament

Tot el codi generat ha d'estar correctament comentat per a poder executar el JavaDoc, generant automàticament la documentació en línia del codi.

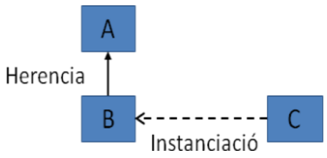
La memòria ha de contenir els punts descrits en la normativa de pràctiques i els punts següents:

1. Introduir el problema tractat en el lliurament (No s'acceptarà una còpia directa de l'enunciat).
2. Explicar les classes implementades.
3. Explicar què s'ha pogut reutilitzar de la classe TaulaImatges (punt 2.3.2) per a la implementació de la classe LlistaImatges (punt 2.7).
4. Explicar com has declarat l'atribut de la classe VisorUB i perquè.
5. Explicar què cal fer per canviar el tipus d'implementació de la llista d'imatges en el VisorUB.

6. Dibuixar el diagrama de relacions entre les classes que has utilitzat a la teva pràctica. Incloure tant les classes implementades per tu com les que pertanyen a la llibreria UtilsProg2. No cal incloure la llista d'atributs i mètodes.
   Utilitzar la notació de la següent figura, per indicar que B és una classe que hereta de A i C conté una instància de la classe B, com al següent codi:

```
public class B extends A{
    //...
}

public class C{
     B b =null;
}
```



7. Explicar quins són els atributs de la classe Imatge i perquè.

8. Explicar com has implementat i on has utilitzat el mètode **isFull** heretat de la classe ImageList.

9. Detallar les proves realitzades per comprovar el correcte funcionament de la pràctica, resultats obtinguts i accions derivades.
10. Segons la implementació de la classe LlistaImatges, si tenim dues imatges corresponents al mateix fitxer, quan cridem al mètode per eliminar un d'aquests fitxers eliminarà l'altre també o no?
11. Observacions generals.

6. Data límit del lliurament

El lliurament es realitzarà en dues parts:

1. El dia 8 de març es lliurarà la primera versió amb les opcions 1, 2 i 3 i 6 del menú. I seguint els punts 3.1. al 3.5.

2. El dia 15 de març es lliurarà la segona versió amb totes les opcions del menú. Es completen els punts 3.6. al 3.8.