Pràctica 2

1. Objectius

Se'ns ha encomanat dissenyar una eina adreçada a la gestió d'un centre excursionista que permeti la gestió dels seus socis. En concret, cal desenvolupar una aplicació amb un menú de text que permeti, entre altres funcionalitats, entrar les dades de cada soci, gestionar la llista de socis i calcular el total a pagar en la factura mensual d'un soci determinat.

La quota base mensual del club és de 25€ i les excursions sempre són d'un sol dia i el seu preu és de 20€. Els socis poden ser federats, estàndards o juniors. De cada soci caldrà guardar el seu nom i el seu DNI.

Pels socis estàndards caldrà guardar també l'assegurança contractada per les excursions, que té un tipus (Bàsica o Completa) i un preu.

Els socis federats tenen la seva pròpia assegurança de la Federació i per tant no caldrà que la paguin a part per a cada excursió. A més, els socis federats tindran un descompte en la quota mensual obligatòria del centre excursionista i en el preu de les excursions. Per tant, pels socis federats caldrà guardar, a més dels atributs de qualsevol soci, també la federació (que té un nom i un preu), el descompte que s'aplicarà a la quota mensual (en %) i el descompte que s'aplicarà al preu de les excursions (en %).

Per últim, els socis junior no tindran assegurança, ja que estan coberts per l'assegurança del club i a més no han de pagar el preu de les excursions.

El total de la factura mensual d'un soci es calcularà sumant la quota mensual del club més el número d'excursions multiplicat pel preu de les excursions per cada tipus de soci.

Pel càlcul de la quota mensual, en el cas dels socis estàndards i els socis juniors serà la quota base mensual integra, mentre que en el cas dels socis federats se li ha d'aplicar el descompte assignat.

Pel càlcul del preu de les excursions, en el cas dels socis estàndards serà el preu fix de les excursions més el preu de l'assegurança contractada, en el cas dels socis federats al preu de les excursions no se li ha de sumar res i s'ha de fer el descompte i en el cas dels socis juniors el preu passa a ser 0.

Els socis s'organitzaran dins una llista de socis. Aquesta llista no podrà contenir dos socis iguals, és a dir amb el mateix DNI.

Per tal d'interactuar amb les dades del club, caldrà definir un menú amb les opcions següents:

1. Donar d'alta un nou soci: Mostra un menú per tal d'afegir un soci a la llista de socis.
    i. Afegir soci federat: Demana les dades necessàries per un nou soci federat.
    ii. Afegir soci estàndard: Demana les dades necessàries per un nou soci estàndard.
    iii. Afegir soci junior: Demana les dades necessàries per un nou soci junior.
    iv. Menú anterior: Torna al menú anterior.
2. Mostrar llista de socis: Mostra el contingut de la llista de socis del ClubUB, mostrant davant de cada soci, el nombre de la seva posició a la llista començant per 1.
3. Eliminar soci: Elimina un soci de la llista indicant la seva posició a la llista.
4. Mostrar factura: Mostra el total que ha de pagar un soci determinat en la factura d'un mes, indicant el seu DNI i el número d'excursions que ha fet aquell mateix mes.
5. Modificar nom soci: Permet canviar el nom d'un soci per un altre.
6. Modificar tipus assegurança soci: Permet canviar el tipus d'assegurança d'un soci indicant el seu DNI i el nou tipus d'assegurança ("Bàsica" o "Completa").
7. Guardar llista: Guarda el contingut de la llista en un fitxer.
8. Recuperar llista: Carrega una llista prèviament guardada d'un fitxer.
9. Sortir: Surt de l'aplicació.

Com a la pràctica 1, organitzareu el codi en dos paquets [paquet Vista] (***prog2.vista***) o [paquet Model](**prog2.model**). Les classes de la vista s'encarregaran de la interacció amb l'usuari i les classes del model contindran totes les dades que s'han de manipular i faran totes les accions que afectin a les dades. Des de la classe principal de la vista es demanaran els valors dels atributs necessaris per passar-los a les classes del model**.** Tingueu en compte que **tots els prints s'hauran de fer a la vista, mitjançant el tractament d'excepcions**, i no hi haurà cap print a cap classe del model.

2. Material per la pràctica

Per aquesta pràctica us proporcionem les següents clases i interfícies:

- ***Menu***
- ***InSoci*** **(interface)**
- ***InSociList*** **(interface)**

Podeu trobar aquest codi al Campus Virtual (junt amb aquest enunciat "PRÀCTIQUES -> Enunciat i codi de la Pr2") i afegir-lo al vostre projecte.
Igual que a la pràctica 1 podeu instanciar objectes de la classe ***Menu*** a la vostra classe vista per implementar la gestió del menú de l'aplicació del Club. A més, haureu d'utilitzar les interfícies en el desenvolupament de la pràctica tal com s'indica en la secció 3.

3. Descripció de la pràctica

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 NetBeans, al qual li heu de posar com a nom ***Cognom1Nom1Cognom2Nom2***, on 1 i 2 fa referència als dos membres de la parella, i tenint en compte les següents consideracions:

- La primera lletra de cada part en majúscula i la resta en minúscula.
- Eviteu utilitzar accents i caràcters del tipus ñ o ç.

  Per exemple, una estudiant amb nom Dolça Martínez Castaña, hauria de crear un projecte amb el nom *MartinezCastanaDolca*.
- La classe principal s'ha de dir ***IniciadorClubUB***, i el paquet per defecte ***prog2.vista***.
- En el NetBeans s'ha d'indicar la classe principal com ***prog2.vista.IniciadorClubUB***

3.2. Classe IniciadorClubUB

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

3.3. Classe VistaClubUB

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

```
public void gestioClubUB();
```

Un cop creat, aquest mètode es cridarà des del mètode ***main*** de la classe ***IniciadorClubUB***:

```
club.gestioClubUB ();
```

Us deixem que penseu per vosaltres mateixos quin serà l'atribut d'aquesta classe.

3.3.1. Implementació del menú d'opcions i la lògica del programa

Un cop teniu el projecte i la classe principal, definiu la lògica del programa, o sigui, com es comportarà el programa dins del mètode ***gestioClubUB***. 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, seguiu l'exemple de la pràctica 1.

3.4. Creació de les classes per emmagatzemar la informació del club i els socis

Les classes que utilitzeu per guardar la informació del club i els socis han d'anar dins el [paquet Model] (***prog2.model***).

3.4.1. Classe ClubUB

La classe ***ClubUB*** conté tota la informació sobre el club. L'haureu de desenvolupar de forma autònoma, seguint el teu criteri de programador(a). A través d'aquesta classe, la vista (és a dir, la classe ***VistaClubUB***) demana tota la informació sobre el club que necessita. Per tant, a la classe ***VistaClubUB*** no s'haurà de fer import de cap classe del model a part de la classe ***ClubUB***.

En aquesta classe estaran definides les constants corresponents al preu per excursió (20€), a la quota mensual (25€) i als descomptes pels socis federats (20% descompte del preu de l'excursió i 30% descompte de la quota base mensual).

A ***ClubUB*** hi definireu dos mètodes per tal de poder **guardar-se** i **carregar-se** **com objecte** mitjançant Streams (veure secció 3.6) al disc de l'ordinador. D'aquesta manera no haureu de tornar a introduir tota la informació del club per cada nova execució del programa. El mètode per carregar-se s'ha de definir com un mètode de classe. Algunes classes hauran d'implementar la interfície ***Serializable***, per tal de permetre als seus objectes ser guardats en un fitxer (veure document d'ajuda a la pràctica 2).

3.4.2. Classe Soci

La classe ***Soci*** ha de guardar la informació d'un soci. Aquesta classe serà abstracta i contindrà els atributs principals d'un soci (nom i DNI). Caldrà que implementeu el/s constructor/s i els mètodes que estan declarats a la interfície ***InSoci***. A més, s'ha d'implementar el mètode ***toString*** per tal de poder mostrar per pantalla la informació d'un soci.

3.4.3. Classes SociFederat, SociEstandard i SociJunior

Les clases **SociFederat**, **SociEstandard** i **SociJunior** són subclasses de la classe **Soci**.

La clase **SociEstandard** tindrà com atribut un objecte de tipus ***Asseguranca*** i la classe **SociFederat** tindrà com atribut un objecte de tipus ***Federacio***. Seguiu les indicacions de l'enunciat a l'apartat 1 per implementar els mètodes necessaris.

A més, els constructors de les clases **SociEstandard** i **SociFederat** hauran de cridar a un mètode de suport de la clase anomenat *comprova* i que llançarà una excepció del tipus ***ExcepcioClub*** en els següents casos:

- Pel **SociEstandard**, si el tipus d'assegurança NO és "Bàsica" o "Completa", es llança una excepció amb el missatge "El tipus d'assegurança no és correcte".
- Pel **SociFederat,** si el preu de la federació és menor de 100€, es llança una excepció amb el missatge "El preu de la federació no és correcte".

Aquí s'ha d'implementar el mètode ***toString*** per tal de poder mostrar per pantalla la informació específica d'un soci estàndard i federat a més de la corresponent a un soci.

3.4.4. Classe *LlistaSocis*

Un cop tenim les classes necessària per guardar la informació d'un soci, ara definiu la classe que representarà una llista de socis. Igual que en el cas del soci, caldrà que implementeu el/s constructor/s i una sèrie de mètodes que us permetran manipular aquesta classe i que estan declarats a la interfície ***InSociList***.

Per implementar la llista de socis heu d'utilitzar la classe **ArrayList** de Java (java.util.ArrayList). Aquesta llista ha de tenir una capacitat màxima per defecte de 100 socis, però s'ha de donar l'opció d'introduir la mida màxima com a paràmetre en la construcció de la llista.

Recordeu que aquesta llista no podrà contenir dos socis iguals. Per tant, abans d'afegir un nou soci a la llista s'haurà de comprovar que aquest no estigui ja inclòs. Per fer-ho podeu fer servir un mètode ***contains*** que comprova si un soci està dins de la llista fent servir el mètode ***equals*** que ha d'estar implementat a la classe **Soci**. Considerarem que dos socis són iguals si tenen el mateix DNI.

Per últim, no oblideu implementar els mètodes ***toString*** per mostrar el resum de la llista de socis i que alhora faci servir el mètode ***toString*** dels objectes socis. Aquest resum s'haurà de mostrar de la següent manera:

```
Llista de Socis:
==============

[1] Nom=Nuria Roca, DNI=11111. Federació: Nom=FCM, Preu=150.0.

[2] Autor=Pere Gil, DNI=22222. Assegurança: Tipus=Bàsica, Preu=15.0.
```

3.4.5. Classe *Asseguranca* i *Federacio*

Aquestes classes permetran encapsular la informació relativa a l'assegurança i federació. Aquestes classes, a més del mètode constructor i els setters i getters, han de tenir un mètode ***toString*** per definir la forma de mostrar tota la informació per pantalla.

3.5. Gestió de les dades d'entrada

Un cop definides les classes que permetran guardar els socis i les llistes de socis, ja podeu començar a donar funcionalitat a les **opcions 1, 2, 3, 4, 5 i 6 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, sempre que pugueu utilitzeu els mètodes estàndards dels objectes:

```
public String toString();

public boolean equals(Object obj);
```

3.6. Persistència de dades

El què volem fer ara és poder guardar les dades (***ClubUB***) a un fitxer en disc i poder-les carregar posteriorment. Implementeu les **opcions 7 i 8 del menú**.

Per implementar aquestes opcions, necessiteu seguir els següents passos, que s'explicaran amb detall a les classes:

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 "clubUB.dat" (Nota: no cal que existeixi a disc, es crea en el moment de fer new), farem:

```
File fitxer = new File("clubUB.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 de tipus ***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 a la pràctica 2 al Campus Virtual. No oblideu tancar correctament els objectes d'accés als fitxers.

3.7. Excepcions

En aquest lliurament, heu d'utilitzar excepcions per fer la gestió dels errors, com per exemple, l'error d'afegir una assegurança d'un tipus incorrecte o l'error d'intentar eliminar un soci donant un DNI que no correspon a cap soci de la llista.

Creeu la classe ***ExcepcioClub*** dins el [paquet Vista] (***prog2.vista***) per representar aquests errors particulars. A la part del codi on es produeix un problema a controlar, heu de crear aquesta excepció i llançar-la, fent servir ***throw*** dins del mètode corresponent. La paraula ***throws*** al final de la capçalera del mètode ens permetrà delegar la captura d'aquesta excepció fins al mètode on es vols gestionar. Utilitzeu el mateix constructor de l'excepció per introduir-hi el missatge d'error adequat i específic de cada error. Finalment, feu servir *try-catch* per capturar la excepció i llavors informar a l'usuari de l'error produït, així com del motiu de l'error i controlar l'execució del reproductor segons l'error produït.

Teniu més informació sobre *excepcions* en el document d'ajuda a la pràctica 2 al Campus Virtual.

4. 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 comprimit (ZIP) amb el nom dels dos membres de la parella: **Cognom1Nom1Cognom2Nom2_P2**, que contingui:

1. El projecte sencer de NetBeans

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

2. La memòria del lliurament.

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

1. Explicar les classes implementades.
2. Explicar com has declarat l'atribut de la classe **ClubUB** i perquè.
3. Dibuixar el diagrama de relacions entre les classes que heu utilitzat a la vostra pràctica. No cal incloure la llista d'atributs i mètodes.
4. Explicar quins són els atributs de la classe **Soci** i perquè.
5. Per què creieu que la classe **Soci** ha de ser abstracta?
6. Creieu que podríem haver treballat només amb dues classes **Soci** i **SociFederat**? Per què creus que és necessari crear les tres classes **Soci**, **SociEstandard**, **SociFederat i SociJunior**?
7. Indiqueu si hi ha i on es troben els exemples de mètodes polimòrfics al vostre codi.
8. Analitzeu perquè l'objecte de tipus ***Asseguranca*** es crea fora del constructor de la classe ***SociEstandard***. Es podria instanciar a dins? Com canviaria el sentit de la implementació?
9. Expliqueu com heu utilitzat la classe ***ExcepcioClub*** al vostre codi.
10. Si teniu un soci federat que fa 5 excursions en un mes quina serà la quantitat de la seva factura aquell mes?
11. Si teniu un soci estàndard amb una assegurança de tipus Bàsica d'un preu de 5€ i que fa 5 excursions en un mes quina serà la quantitat de la seva factura aquell mes?

12. Detallar les proves realitzades per comprovar el correcte funcionament de la pràctica, resultats obtinguts i accions derivades.

13. Observacions generals.

5. Data límit del lliurament

Consultar el calendari de lliuraments al Campus Virtual.