UNIVERSITAT DE BARCELONA

PRÀCTICA 2: CLUB UB

14 D'ABRIL DE 2021

NOAH MÁRQUEZ VARA & GUILLEM NAVARRA PANOS
GRUP F

Pràctica 2

Classes implementades

1. **Classe IniciadorClubUB**
   Té com a responsabilitat llançar el bucle principal del nostre programa. La classe de la vista ***IniciadorClubUB*** té un mètode estàtic ***main()*** on es crea un objecte de tipus ***VistaClubUB*** anomenat *vistaClub*. Després crida el mètode ***gestioClubUB()*** de l'objecte *vistaClub*, on es troba el bucle principal de l'aplicació.

2. **Classe VistaClubUB**
   És la responsable de fer la interfície entre l'usuari i el model del nostre programa. Conté un atribut privat de tipus *ClubUB.* Conté també un mètode públic anomenat ***gestioClubUB()*** (es crida des de la classe ***IniciadorClubUB***), on s'implementa el menú de l'aplicació.

3. **Classe Menu**
   Classe encarregada del funcionament i gestió del menú d'opcions del programa. Comprova que l'input de l'usuari sigui correcte (que seleccioni una de les opcions del menú).

4. **Classe ClubUB**
   La classe *ClubUB* conté tota la informació sobre el club. A través d'aquesta classe, la vista (***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***.

   Conté 5 atributs, 4 dels quals són de tipus *private static final* degut a que són constants:

   - **PREU_EXCURSIO = 20** (***private static final float***, especifica el preu de cada excursió, tot i que els socis federats tenen un descompte).
   - **QUOTA_MENSUAL = 25** (***private static final float***, especifica la quota mensual a pagar per cada soci, tot i que els socis federats tenen un descompte).
   - **DESCOMPTE_PREU_EXCURSIO = 20** (***private static final float***, especifica el descompte a aplicar al preu de les excursions que realitzin els socis federats).
   - **DESCOMPTE_QUOTA_MENSUAL = 30** (***private static final float***, especifica el descompte a aplicar al preu de la quota mensual a pagar per part dels socis federats).

- ***LlistaSocis _llistaSocis***, atribut que farem servir per manipular la llista de socis del nostre club.

Consta també d'un constructor sense paràmetres on cridem al constructor de l'atribut ***_llistaSocis*** per tal d'inicialitzar la llista on guardarem els nostres socis.

A continuació la recopilació de mètodes de la classe ***ClubUB***:

- ***getLlistaSocis***, ens serveix per retornar un *String* amb la llista de socis o bé un *String* dient que: 'La llista no conté cap soci.'.
- ***llistaBuida***, mètode que farem servir des de la classe ***VistaClubUB*** per comprovar si la llista de socis està buida abans de deixar a l'usuari modificar el nom d'un soci, sol·licitar la factura d'un soci, eliminar un soci o modificar l'assegurança d'un soci.
- ***llistaPlena***, mètode que farem servir des de la classe ***VistaClubUB*** per comprovar si la llista de socis està plena abans de demanar a l'usuari les dades per afegir un nou tipus de soci.
- ***removeSoci***, fem servir aquest mètode per eliminar un soci a una determinada posició de la llista especificada per l'usuari. Si la posició introduïda és negativa o be excedeix la mida actual de la llista, es llença una excepció.
- ***calcularFactura***, mètode que fem servir per calcular la factura d'un determinat soci. Rep per paràmetres el DNI del soci a mostrar la factura i el nombre d'excursions que ha realitzat durant el mes. Es crida llavors als mètodes ***calculaPreuExcursio*** i ***calculaQuota*** del soci, per tal d'efectuar el càlcul de l'import total. Si el nombre d'excursions introduït és superior a 31 (un mes) es llença una excepció. Si no es troba cap soci a la llista amb el DNI introduït també saltarà una excepció.
- ***afegirSociFederat***, afegeix un soci de tipus federat. Rep per paràmetres el nom, el seu DNI, el nom de la federació i el preu d'aquesta. Es crea un objecte de tipus ***Federacio*** i es crida al seu constructor per paràmetres amb el nom i el preu de la federació.

  Posteriorment es crea un objecte de tipus ***SociFederat*** cridant al seu constructor amb el seu nom, dni, objecte de tipus ***Federacio***, el descompte que se li aplicarà en el càlcul de la quota mensual i el descompte que se li aplicarà a les excursions que realitzi. Si es detecta que el preu de la federació introduït és menor a 100, salta una excepció (procedent del constructor de la classe ***SociFederat***). Si a l'hora de crear el soci es detecta que ja hi ha un soci amb el mateix DNI, salta una excepció i no s'afegeix el

soci a la llista.

- ***afegirSociEstandard***, afegeix un soci de tipus estàndard. Rep per paràmetres el nom, el DNI, el tipus d'assegurança (Bàsica o Completa) i el preu d'aquesta. Es crea un objecte de tipus ***Asseguranca*** i es crida al seu constructor per paràmetres amb el tipus i el preu de l'assegurança.

  Posteriorment es crea un objecte de tipus ***SociEstandard*** cridant al seu constructor amb el seu nom, dni i objecte de tipus ***Asseguranca***. Si es detecta que el tipus d'assegurança no és 'Bàsica' o 'Completa' salta una excepció (procedent del constructor de la classe ***SociEstandard***).

  Si a l'hora de crear el soci es detecta que ja hi ha un soci amb el mateix DNI, salta una excepció i no s'afegeix el soci a la llista.
- ***afegirSociJunior***, afegeix un soci de tipus junior. Rep per paràmetres el nom i el DNI. Es crea un objecte de tipus ***SociJunior*** cridant al seu constructor amb el seu nom i DNI. Si a l'hora de crear el soci es detecta que ja hi ha un soci amb el mateix DNI, salta una excepció i no s'afegeix el soci a la llista.
- ***modificaNom***, mètode per a modificar el nom d'un soci. Rep per paràmetres el DNI del soci del qual es vol modificar el nom, i el nou nom. Si a l'hora de buscar el soci es detecta que no existeix cap soci amb el DNI introduït, salta una excepció.
- ***modificaAsseguranca***, mètode per modificar l'assegurança d'un soci estàndard. Rep per paràmetres el DNI del soci del qual es vol modificar l'assegurança i el nou tipus d'assegurança. Si a l'hora de buscar el soci es detecta que no existeix cap soci amb el DNI introduït, salta una excepció. Si el soci no es de tipus ***SociEstandard*** salta una excepció.
- ***static ClubUB load***, permet carregar les dades del club des d'un fitxer. Rep per paràmetres la ruta del fitxer d'es d'on es vol carregar les dades. Pot llençar vàries excepcions (ex: si la ruta al fitxer és incorrecte, si no existeix cap fitxer en la ruta, si hi ha un problema de lectura, si no es pot tancar el fitxer...).
- ***save***, mètode que permet guardar un arxiu amb les dades del club per el seu posterior ús. Rep per paràmetres la ruta al fitxer on volem guardar les dades. Pot llençar vàries excepcions (ex: problema d'escriptura, no es pot tancar el fitxer...).

5. Classe Soci

La classe *Soci* ha de guardar la informació d'un soci. És una classe abstracta i conté els atributs principals d'un soci (nom i DNI).

Els mètodes de la classe són implementats respecte als mètodes de la interfície

InSoci. Entre aquests es troben els *getters & setters* dels principals atributs (nom i DNI), un mètode abstracte (***calculaPreuExcursio***) que calcula el preu de les excursions del soci (al que se li passa per paràmetre un *float* amb el preu de les excursions del soci) i un altre mètode (calculaQuota) per calcular la quota mensual del soci (al que se li passa per paràmetre un *float*, amb la quota mensual d'aquest soci).

A part també té un mètode anomenat ***equals*** que rep per paràmetre un objecte de tipus ***Soci*** i comprova si el soci actual i el rebut per paràmetre tenen el mateix DNI. També tenim un mètode ***toString*** per tal de mostrar per pantalla la informació d'un soci.

6. Classe SociFederat

Subclasse de la classe ***Soci***. Conté com a atribut un objecte de tipus ***Federacio*** (amb la informació d'aquesta), i dos atributs amb el descompte a aplicar a la quota mensual i al preu de les excursions.

Conté un constructor per paràmetres que rep el nom del soci, el seu DNI, un objecte de tipus ***Federacio***, el descompte a aplicar a la quota mensual i el descompte a aplicar a les excursions. Dins d'aquest constructor es crida a un mètode de suport de la classe anomenat *comprova*, que llança una excepció si el preu de la federació és menor de 100€, amb el missatge: "El preu de la federació no és correcte".

Conté a més 4 mètodes:

- ***comprova***, mètode que retorna *true* si el preu de la federació és més gran o igual que 100, o *false* altrament. En cas de retornar *false*, des del constructor de la classe es llançaria una excepció.
- ***calculaPreuExcursio***, rep per paràmetre el preu d'una excursió (20€) i aplica el descompte (20%) del que gaudeix un soci federat. El preu d'una excursió llavors seria de: 16€.
- ***calculaQuota***, rep per paràmetre la quota base mensual del club (25€) i aplica el descompte (30%) del que gaudeix un soci federat. El preu de la quota mensual és llavors de: 17,5€.
- ***toString***, per tal de mostrar per pantalla la informació específica d'un soci federat a més de la corresponent a un soci.

7. Classe SociEstandard

Subclasse de la classe ***Soci***. Conté com a atribut un objecte de tipus ***Asseguranca*** (amb la informació d'aquesta).

Conté un constructor per paràmetres que rep el nom del soci, el seu DNI i un objecte de tipus ***Asseguranca***. Dins d'aquest constructor es crida a un mètode

de suport de la classe anomenat *comprova*, que llança una excepció si el tipus d'assegurança no és 'Bàsica' o 'Completa', amb el missatge: "El tipus d'assegurança no és correcte".

Conté *get & set* de l'objecte de tipus ***Asseguranca***, que ens serà útil a l'hora de voler canviar el tipus d'assegurança. A part d'això, conté 4 mètodes:

- ***comprova***, mètode que retorna *true* si el tipus d'assegurança és correcte, o *false* altrament. En cas de retornar *false*, des del constructor de la classe es llançaria una excepció.
- ***calculaPreuExcursio***, rep per paràmetre el preu d'una excursió (20€) i li suma el preu que ha de pagar el soci per l'assegurança contractada.
- ***calculaQuota***, rep per paràmetre la quota base mensual del club (25€) i retorna aquesta mateixa quota, ja que el soci estàndard no gaudeix de cap descompte en la seva quota mensual.
- ***toString***, per tal de mostrar per pantalla la informació específica d'un soci estàndard a més de la corresponent a un soci.

8. Classe SociJunior

Subclasse de la classe ***Soci***.

Conté un constructor per paràmetres que rep el nom del soci i el seu DNI. Conté també 3 mètodes:

- ***calculaPreuExcursio***, rep per paràmetre el preu d'una excursió (20€) però retorna 0, ja que els socis juniors no han de pagar les excursions que fan.
- ***calculaQuota***, rep per paràmetre la quota base mensual del club (25€) i retorna aquesta mateixa quota, ja que el soci junior no gaudeix de cap descompte en la seva quota mensual.
- ***toString***, per tal de mostrar per pantalla la informació específica d'un soci (nom i DNI).

9. Classe LlistaSocis

Un cop creades les classes necessàries per guardar la informació d'un soci, definim la classe que representarà una llista de socis. Igual que en el cas de la classe ***Soci***, hem creat un constructor sense paràmetres que inicialitza l'*ArrayList* amb una mida màxima de 100 objectes tipus ***Soci*** i un altre constructor amb un paràmetre per donar l'opció d'introduir la mida màxima com a paràmetre en la construcció de la llista.

Conté tres atributs:

- ***Private static final int MIDA_MAXIMA = 100***, constant que especifica la mida màxima de *l'ArrayList*.
- ***int mida***, atribut per guardar la mida màxima en cas d'oferir l'opció d'introduir la mida màxima com a paràmetre en la construcció de la llista.

- ***Private ArrayList<Soci> llista***, declaració de l'atribut per guardar la llista de socis.

Els mètodes que hem implementat són declarats a la interfície ***InSociList***:

- ***getSize***, retorna la mida de l'*ArrayList*.
- ***addSoci***, rep per paràmetre un objecte de tipus ***Soci*** per afegir-ho a la llista. Si la llista està buida afegeix directament el soci. Si la llista està plena llença una excepció. Si troba un soci amb el mateix DNI al que es vol afegir, llença una excepció.
- ***removeSoci***, rep per paràmetre un objecte de tipus ***Soci*** per eliminar-ho de la llista. Si no troba el soci a la llista, llença una excepció.
- ***getAt***, rep per paràmetre la posició del soci a retornar. Si la posició és més gran que la mida actual de la llista (no fa falta posar que sigui més gran que la MIDA_MAXIMA), llença una excepció. Si la posició és negativa també retorna una excepció.
- ***getSoci***, rep per paràmetre el DNI del soci a retornar. Si la llista està buida, retorna una excepció. Si no troba a cap soci amb el DNI indicat, també llençarà una excepció.
- ***clear***, mètode per eliminar tots els elements de la llista. Si la llista està buida llença una excepció.
- ***isFull,*** comprova si la llista està a la seva capacitat màxima o no. Retorna *true* si la llista està plena (no hi ha lloc per més elements) o *false* altrament.
- ***isEmpty***, comprova si la llista està buida o no. Retorna *true* si la llista està buida (no hi ha cap element) o *false* altrament.
- ***toString***, mètode per mostrar el resum de la llista de socis i alhora fa servir el mètode ***toString*** dels objectes de tipus ***Soci***.

10. Classe Asseguranca

Permet encapsular la informació relativa a l'assegurança. Conté dos atributs:

- ***String tipus***, indicant el tipus d'assegurança ('Bàsica' o 'Completa').
- ***float preu***, indicat el preu de l'assegurança.

Conté també un constructor per paràmetres que rep el tipus d'assegurança i el preu d'aquesta. Posteriorment trobem els corresponents *getters & setters* dels atributs ja esmentats i, per últim, un mètode ***toString*** per definir la forma de mostrar tota la informació per pantalla.

11. Classe Federacio

Permet encapsular la informació relativa a la federació. Conté dos atributs:

- ***String nomFederacio***, indicant el nom de la federació.
- ***float preu***, indicant el preu de la federació.

Conté també un constructor per paràmetres que rep el nom de la federació i el

preu d'aquesta. Posteriorment trobem els corresponents *getters & setters* dels atributs ja esmentats i, per últim, un mètode ***toString*** per definir la forma de mostrar tota la informació per pantalla.

12. Classe ExcepcioClub

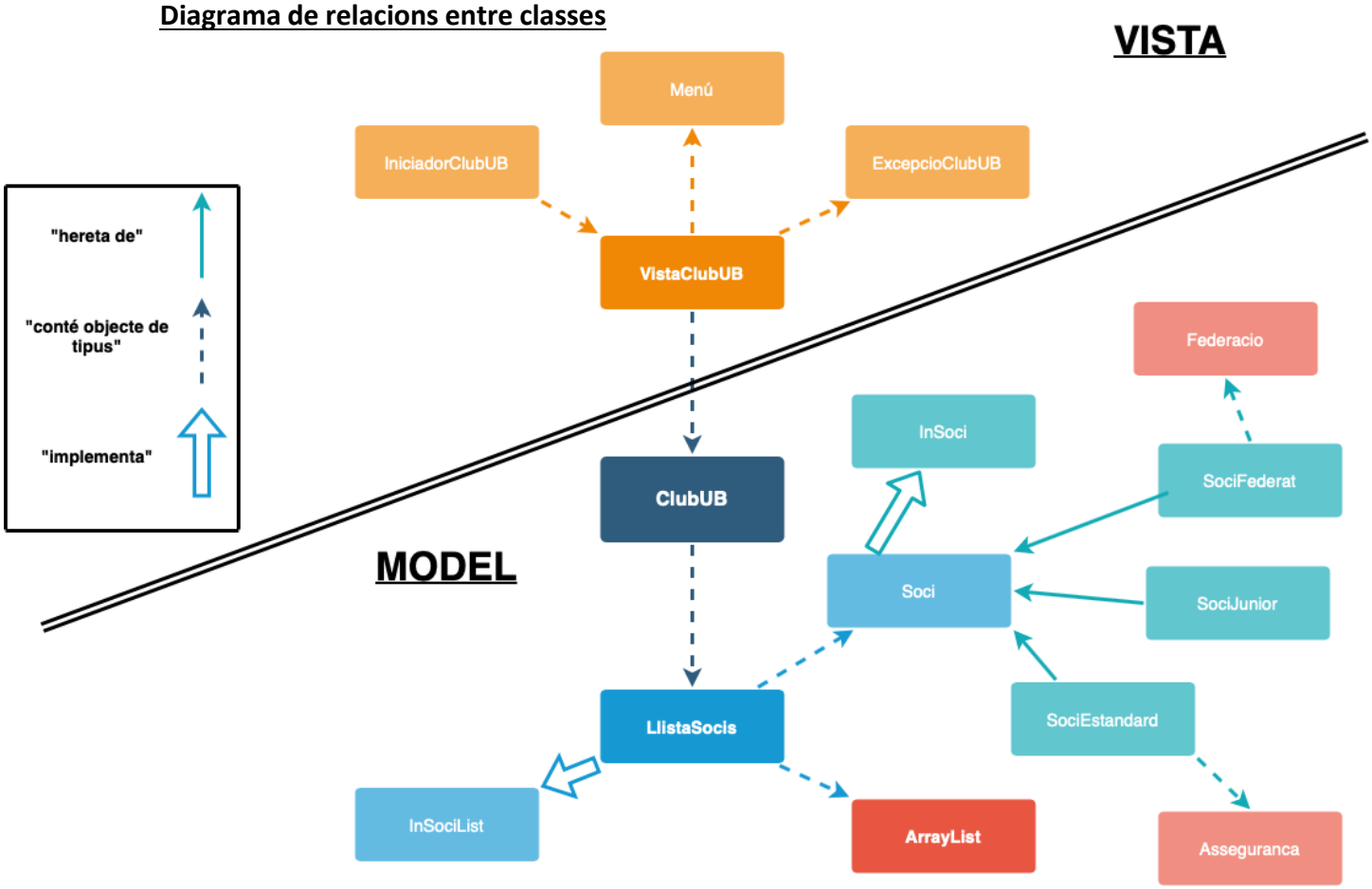
Per tal de fer la gestió d'errors, com per exemple, afegir una assegurança d'un tipus incorrecte o l'error d'intentar eliminar un soci donant un DNI que no correspon a cap soci de la llista. La classe ***ExcepcioClub*** està implementada dins el paquet de la vista per tal de representar tots els errors particulars.

La classe fa un *extends* de *Exception*, té un constructor sense paràmetres que realitza un *super* i un constructor per paràmetres que rep el missatge de l'excepció produïda.

Declaració de l'atribut de la classe ClubUB

L'hem declarat com un mètode privat, ja que no volem que la llista de socis de ***ClubUB*** sigui accedida mitjançant altres mètodes d'altres classes i perquè és necessària per la implementació de mètodes propis de la classe, per poder manipular la llista de socis com faci falta, ja sigui per afegir-hi socis, eliminar, guardar la llista...

Diagrama de relacions entre classes



Atributs de la classe Soci

Els atributs són ***nomSoci*** (String) i ***DNI*** (String). Són aquests ja que són els únics comuns a totes les classes filles. Si haguéssim posat, per exemple, un objecte de tipus ***Asseguranca*** com a atribut de Soci, totes les classes filles haguessin heretat innecessàriament un objecte de tipus ***Asseguranca***, que és únicament necessari a la classe ***SociEstandard***.

Classe Soci abstracte

Perquè d'ella hereten tres Subclasses (***SociJunior***, ***SociFederat*** i ***SociEstandard***) i, principalment, perquè cap Soci pot ser només Soci; ha de ser ***SociFederat***, ***SociEstandard*** o ***SociJunior***, i si la classe no fos abstracta, podríem tenir Socis que només fossin socis, sense especificar de quin tipus.

Treballar amb només dues classes Soci i SociFederat

No tindria gaire sentit treballar només amb dues classes ***Soci*** i ***SociFederat***, ja que seria una implementació menys efectiva i desordenada. És necessari crear les tres classes ja que així podem separar els tres tipus de soci i disposar d'una classe mare (***Soci***) per heretar els atributs bàsics d'un soci (el seu nom i el seu DNI).

En el cas de que només féssim ús de de les dues classes ***Soci*** i ***SociFederat***, a l'hora de voler afegir un soci de tipus estàndard hauríem de tenir un atribut de tipus ***Asseguranca*** a la classe ***Soci***, fent així que també l'heretés ***SociFederat***, quan aquest no necessita d'un objecte de tipus ***Asseguranca***.

Mètodes polimòrfics en el nostre codi

Sí, per exemple al mètode ***calculaFactura*** de ***ClubUB*** es fa

Soci soci = _llistaSocis.getSoci(dni);

Que és un exemple de polimorfisme, ja que s'està declarant com a Soci i després s'apunta a una classe filla de Soci. A la mateixa classe passa el mateix amb els mètodes ***modificaNom***, ***modificaAsseguranca*** i ***removeSoci***.

Creació de l'objecte de tipus Asseguranca fora del constructor de la classe SociEstandard

Això és degut a que ***SociEstandard*** conté un objecte de tipus ***Asseguranca***, i creant l'objecte fora del constructor de la classe ***SociEstandard***, ens permet passar-li per paràmetres directament l'objecte de tipus ***Asseguranca*** ja inicialitzat amb les seves dades (tipus i preu), sense haver de rebre al constructor de ***SociEstandard*** aquests paràmetres.

Es podria instanciar a dins, però en comptes de passar-li l'objecte de tipus ***Asseguranca***, li hauríem de passar els paràmetres de l'assegurança (tipus i preu) i que el constructor de la classe ***SociEstandard*** inicialitzés l'assegurança. No canviaria gaire el sentit de la implementació, però és una millor pràctica crear l'objecte de tipus ***Asseguranca*** fora del constructor de la classe ***SociEstandard***.

Classe ExcepcioClub

L'hem implementat de manera que cada cop que un mètode té la possibilitat de no funcionar per qualsevol raó, llença una excepció, que està definida dins el mètode de sortida, com ara "La llista està plena.", que passa per ***ExcepcioClub*** i, mitjançant un print (en la vista), s'informa a l'usuari de l'error produït.

Soci federat que fa 5 excursions en un mes

Recordem que un soci federat té un descompte del 20% sobre el preu base de les excursions i un descompte del 30% sobre la quota base mensual del club. Llavors, tenim que el preu per cada excursió d'un soci federat serà de 20€ - 20% = 16€/excursió. I la quota mensual serà igual a 25€ - 30% = 17,5€.

Així doncs, tenim que la factura d'aquest soci serà igual a:

17,5€ (quota mensual) + (16€/excursió · 5 excursions) = 97,5€.

L'import total de la factura serà de **97,5€**.

```
Introdueix el nom del Soci Federat:
Peter
Introdueix el DNI del Soci Federat:
123456789A
Introdueix el nom de la Federació:
FCF
Introdueix el preu de la Federació:
150
-----------------------------------------------
MENU CLUBUB
-----------------------------------------------
        1.- Donar d'alta un nou soci
        2.- Mostrar la llista de socis
        3.- Eliminar soci
        4.- Mostrar factura
        5.- Modificar nom soci
        6.- Modificar tipus assegurança soci
        7.- Guardar llista
        8.- Recuperar llista
        9.- Sortir
-----------------------------------------------
Entra una opcio >> 4
Introdueix el DNI del Soci per mostrar la seva factura:
123456789A
Introdueix el nombre d'excursions realitzades pel soci:
5
L'import total de la factura del Soci amb DNI 123456789A és: 97.5€.
```

Imatge de l'execució del programa amb les dades esmentades.

Soci estàndard amb assegurança de tipus Bàsica d'un preu de 5€ i que fa 5 excursions en un mes

Recordem que un soci estàndard no té cap descompte sobre el preu base de la quota base mensual del club i que al preu de les excursions se l'ha de sumar el preu de l'assegurança (que es paga per a cada excursió). Llavors, tenim que el preu per cada excursió d'aquest soci estàndard serà de 20€ + 5€ = 25€/excursió. I la quota mensual serà igual a 25€.

Així doncs, tenim que la factura d'aquest soci serà igual a:

25€ (quota mensual) + (25€/excursió · 5 excursions) = 150€.

L'import total de la factura serà de **150€**.

```
Introdueix el nom del Soci Estàndard:
Peter
Introdueix el DNI del Soci Estàndard:
1234A
Introdueix el tipus d'Assegurança (Bàsica o Completa):
Bàsica
Introdueix el preu de l'Assegurança:
5
-----------------------------------------------
MENU CLUBUB
-----------------------------------------------
        1.- Donar d'alta un nou soci
        2.- Mostrar la llista de socis
        3.- Eliminar soci
        4.- Mostrar factura
        5.- Modificar nom soci
        6.- Modificar tipus assegurança soci
        7.- Guardar llista
        8.- Recuperar llista
        9.- Sortir
-----------------------------------------------
Entra una opcio >> 4
Introdueix el DNI del Soci per mostrar la seva factura:
1234A
Introdueix el nombre d'excursions realitzades pel soci:
5
L'import total de la factura del Soci amb DNI 1234A és: 150.0€.
```

Imatge de l'execució del programa amb les dades esmentades.

<u>Proves realitzades per comprovar el correcte funcionament de la pràctica</u>

Per comprovar el correcte funcionament de la pràctica, a banda de comprovar la funcionalitat de les 8 opcions del menú, s'ha comprovat sobretot que funcionés la persistència de les dades, és a dir, la possibilitat de guardar la informació del club i recuperar-la posteriorment. A continuació una llista de les comprovacions realitzades per veure el correcte funcionament d'aquesta pràctica:

1. Comprovació de la possibilitat d'afegir els tres tipus de soci diferent (estàndard, federat i junior).
2. Intentar afegir un nou soci amb un DNI d'un soci que ja es troba a la llista per comprovar el funcionament de les excepcions.
3. Introduir valors més petits que 100 en el preu de la federació o un tipus d'assegurança diferent a "Bàsica" o "Completa", per comprovar el funcionament de les excepcions.
4. Mostrar la llista de socis de la manera demanada en l'enunciat i veure que els diferents socis s'imprimeixen de manera adequada amb tota la seva informació.
5. Eliminar un soci donant una posició. També s'han donat valors de posició més grans a la mida actual de la llista, i valors de posició negatius per comprovar el funcionament de les excepcions.
6. Mostrar la factura d'un soci donant el seu DNI i el nombre d'excursions realitzades per posteriorment comprovar que l'import total era correcte per tots els tipus de socis. També s'ha donat un DNI d'un soci que no es trobava a la llista

per comprovar el funcionament de les excepcions.

7. Modificar nom soci donant el DNI del soci a modificar el nom i el nou nom. S'ha donat també un DNI d'un soci que no es troba a la llista per comprovar el funcionament de les excepcions.
8. Modificar el tipus d'assegurança d'un soci donat el seu DNI i el nou tipus d'assegurança. S'ha donat també un DNI d'un soci que no es troba a la llista per comprovar el funcionament de les excepcions.
9. Guardar la llista de socis en el disc. S'ha comprovat que funcionés correctament i que es creés el fitxer de manera correcte en el disc. Aquesta opció pot llençar vàries excepcions, ja sigui per un problema d'escriptura o perquè no es pot tancar el fitxer, per exemple.
10. I per últim, recuperar una llista de socis guardada en el disc. S'ha comprovat que fos capaç de llegir fitxers des de disc. Aquesta opció també pot llençar vàries excepcions: si no es troba el fitxer, si hi ha un problema de lectura, si no es pot fer *casting* de l'objecte del fitxer a un de tipus ***ClubUB***, entre d'altres.

Observacions generals

Amb aquesta pràctica hem vist i posat a prova diferents eines del llenguatge de programació Java que ens permeten fer herència entre classes, gestionar excepcions i dur a terme la persistència de les dades, entre d'altres. Tot això ens ha servit per gestionar diferents situacions i controlar diferents tipus de dades finsa arribar a gestionar un centre excursionista. Ens ha servit per a poder familiaritzar-nos amb eines més complexes i versàtils i poder aprendre dels nostres errors i fer un aprenentatge de prova-error, fent així que els temes explicats a classe quedin molt més clars.