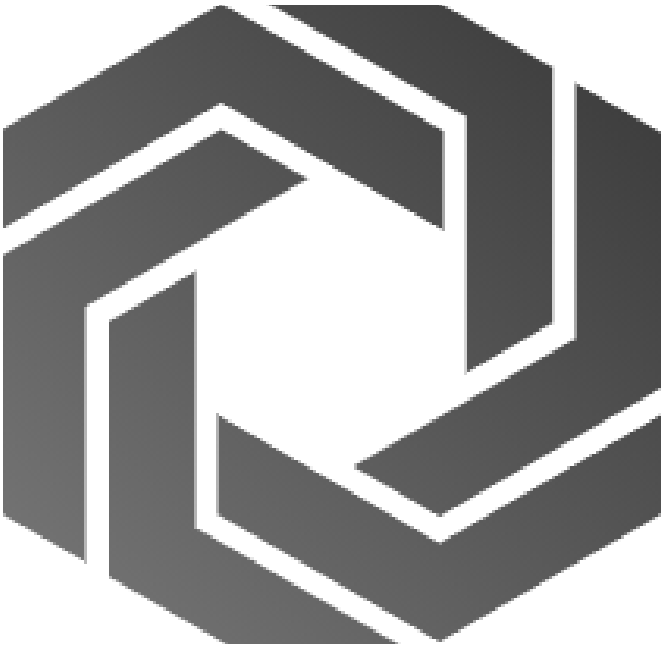
-

Pràctica 2: ClubUB



Oscar de Caralt Roy MC

Pau Segura Baños MC

ENGINYERIA INFORMÀTICA UB

Primer de carrera: Programació 2

09/04/2021

ÍNDEX:

Introducció:

La pràctica que hem fet durant aquestes tres setmanes consisteix a construir un programa que modeli la informació corresponent al ClubUB i permeti organitzar i/o gestionar els socis pertanyents a aquest club.

Per a realitzar això es requereixen uns coneixements bàsics de POO (programació orientada a objectes), herència de classes, de polimorfisme, de creació de classes amb els seus respectius atributs, d'encapsulament, de creació d'interfícies, de creació de fitxers i de control d'excepcions.

A l'apartat d'implementació de classes hi ha una explicació de la classe, però l'explicació de cada mètode està inclosa als respectius javadocs.

1. Explicar les classes implementades:

Classe Soci: aquesta classe conté tota la informació de soci. Conté un constructor amb l'String nom i l'String DNI. També conté els respectius setters i getters. En aquesta classe s'hi troba el mètode toString, que transforma les dades en una única String que mostra per pantalla. En la classe Soci, també s'instancien els mètodes per calcular la quota mensual i el preu de les excursions. Aquests dos mètodes només es declaren, ja que s'acabaran d'implementar en les tres classes filles de Soci, mitjançant el polimorfisme.

Classe LlistaSocis: en aquesta classe es crea una ArrayList que contindrà informació de tipus Soci. Conté tots aquests mètodes: iguals, contains, getSize, addSoci, removeSoci, getAt, getSoci, clear, isFull, isEmpty. La majoria d'aquests mètodes també contenen les excepcions.

Classe Asseguranca: aquesta classe conté el constructor amb els seus dos atributs (tipus i preu) i els seus respectius setters i getters. A més també hi ha el mètode toString. Aquesta classe serveix per declarar l'assegurança de cada soci, la qual comporta una despesa extra.

Classe Federacio: aquesta classe conté el constructor amb els seus dos atributs (nom i preu) i els seus respectius setters i getters. A més també hi ha el mètode toString. Aquesta classe serveix per declarar la federació de cada soci, que comporta descomptes pels socis federats.

Classe SociJunior: aquesta classe hereda de la classe Soci. El constructor és el mateix constructor que a Soci (això ens és possible d'implementar gràcies a la paraula reservada super). També conté el mètode toString. Aquest tipus de soci com a particularitat té que la quota d'excursions li surt gratuïta.

Classe SociFederat: aquesta classe hereda de la classe Soci. El constructor a més dels elements de Soci, afegeix el preu i el nom de la federació i els descomptes a la quota i a les excursions. També conté el mètode toString i mitjançant el polimorfisme s'implementa el mètode calculaPreuExcursió que aplica el descompte a la quota de les excursions.

Classe SociEstandard: aquesta classe hereda de la classe Soci. El constructor a més dels elements de Soci afegeix el preu i el tipus de l'assegurança. Aquesta assegurança farà que el preu de la quota sigui més elevat.

Interfície InSoci: s'instancien tots el mètodes i constructors que posteriorment es desenvoluparan a la classe Soci.

Interfície InSociList: s'instancien tots els mètodes i constructors que posteriorment es desenvoluparan a la classe LlistaSocis.

Classe ClubUB: en aquesta classe és on desenvolupen la majoria de mètodes. És el pas previ per la implementació del programa. En aquesta classe es crea un llista buida, per anar ficant socis. També es declara el preu base de les excursions i de la quota mensual i els seus respectius descomptes. En aquesta classe hi haurà els mètodes per afegir els tres tipus de soci a la llista, un mètode per mostrar tota la llista, un per eliminar un soci de la llista, un per mostrar la factura d'un soci determinat i un altre per modificar l'assegurança dels socis estàndards. A més hi ha dos mètodes per guardar i carregar un arxiu que conté les dades del club.

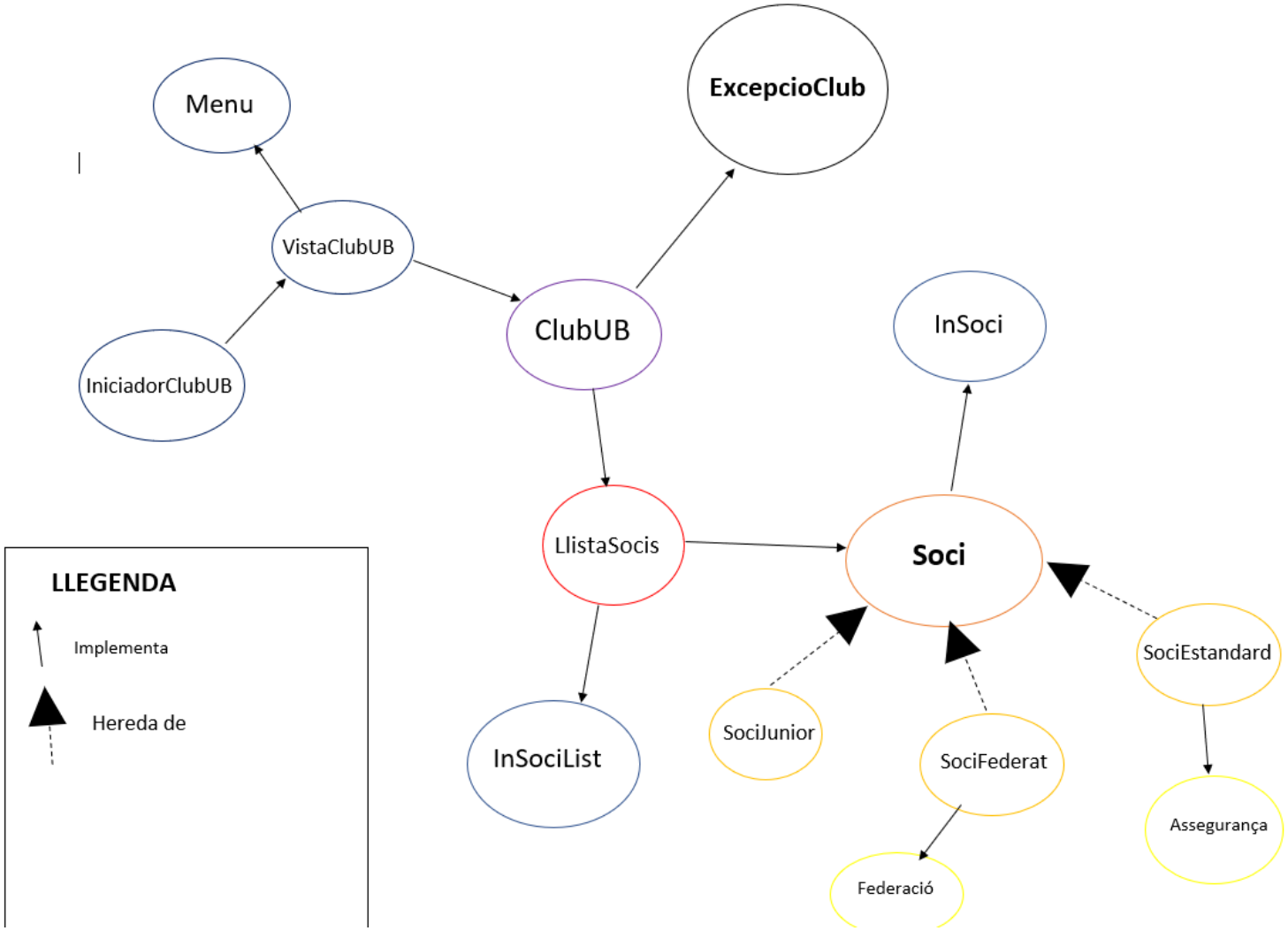
Classe ExcepcioClub: En aquesta classe hem fet un *extends* exception. Hem implementat el constructor sense paràmetres i un mètode que llença un missatge.

2.Explicar com has declarat l'atribut de la classe ClubUB i perquè:

Hem declarat l'atribut ClubUB a la classe VistaClubUB de la següent manera: ClubUB club = new ClubUB();.

L'hem declarat així perquè no requereix cap argument per a la seva construcció i hem preferit optar per la privacitat per defecte que és similar a la pública, per així evitar possibles problemes d'herència entre classes.

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 per què:

Com a atributs de la classe Soci tenim el nom i el DNI. El DNI té el paper d'identificador personal i és un codi únic per cada persona. El DNI s'utilitza en el nostre codi per comprovar si ja hi ha algun usuari introduït amb el mateix DNI. En cas afirmatiu, el programa no ens deixaria registrar un altre soci amb el mateix DNI (perquè significaría que estariem registrant 2 cops el mateix soci). El nom serveix també per identificar a cada usuari del club, però aquest sí que es pot repetir.

5. Per què creieu que la classe Soci ha de ser abstracta?

Les classes SociJunior, SociEstandard i SociFederat hereden de Soci. Un mètode abstracte ens dóna la possibilitat d'introduir la declaració d'un mètode (i no la seva definició) en una classe, i implementar aquest mètode en alguna de les subclasses de la classe en que ens trobem (com podrien ser sociEstandard, sociFederat i sociJunior).

D'aquesta manera, la declaració del mètode estarà disponible a la classe, i el podrem utilitzar per a, per exemple, definir altres mètodes, i a més no ens veiem en l'obligació de definir-lo, ja que el seu comportament pot ser que sigui encara desconegut.

Per tot això val la pena que la classe Soci sigui 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?

No. Es necessiten les tres perquè cada classe té una manera diferent de calcular les seves quotes i excursions. A més, la classe sociEstandard requereix d'un objecte de classe assegurança, en canvi la classe sociFederat necessita de la classe federació.

Per tant, no es podria codificar el nostre programa amb només la classe Soci i SociEstandard ja que ens donaria masses errors i dificultats.

7. Indiqueu si hi ha i on es troben els exemples de mètodes polimòrfics al vostre codi.

Sí que hi ha mètodes polimòrfics. A les classes SociJunior, SociEstandard i SociFederat s'empra el polimorfisme amb el mètode de la classe Soci on es calcula la quota mensual. També a cada mètode toString() de cada classe s'utilitza el polimorfisme.

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?

L'objecte de tipus Asseguranca es crea fora del constructor de la classe SociEstandard perquè s'ha de comprovar el tipus d'assegurança i en casos on es veu involucrada l'herencia, els constructors prenen un significat especial, ja que lo normal és que la subclasse necessiti que s'executi el constructor de la superclasse abans que el seu propi constructor, per a que s'inicialitzin correctament aquelles variables que deriven de la superclasse.

9. Expliqueu com heu utilitzat la classe ExcepcioClub al vostre codi:

En casos susceptibles d'excepció, com en els casos de llista buida, llista plena, soci no trobat, etc... hem implementat els seus respectius try-catch i throw new ExcepcioClub. Aquests throws llençaran un missatge diferent segons el possible tipus d'excepció amb la qual podem arribar a tractar.

10. Si teniu un soci federat que fa 5 excursions en un mes quina serà la quantitat de la seva factura aquell mes?

Cada excursió val 20 euros, i diu que fa cinc en un mes, per tant 5 x 20 = 100 €, però al ser un sociFederat té un descompte del 20% a cada excursió. 5 x (20 - (20 x 0,2)) = 80€.
A la quota mensual del club passa el mateix. El preu base son 25 € però amb el descompte del 30 %, el preu queda en 17,5 €.
80 € d'excursions + 17,5 € de quota + preu federació = 97,5 € + preu federació.

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?
Aquest és un sociEstandard. fa 5 excursions al mes i cada una té un preu de 20 euros. A més, per cada excursió que fa, el soci ha de contractar una assegurança, que en aquests cas és de 5 €. Al càlcul del preu total de les excursions se li ha de sumar la quota base mensual.
5 x (20 + 5) de les excursions + 25 de la quota = 150 €.

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

En primer lloc vam crear una llista de 0 elements per comprovar si les excepcions d'afegir un soci en cas de llista plena anaven bé. Un cop comprovat això, amb una llista de 100 elements vem començar a ficar diferents tipus de soci per veure si teníem algun error. També els vam imprimir per comprovar el funcionament correcte dels toStrings. Després vam passar a la part de modificar. Vam provocar situacions on havien de llançar-s'hi excepcions, i efectivament, així va ocórrer. Un cop modificades correctament les assegurances i els noms dels socis, calculadora en mà vam comprovar que la sortida de la factura fos correcta.

13. Observacions generals.

En totes les classes tenim *implements* Serializable perque l'*Stream* es pugui generar correctament. També en totes les opcions de vistaClubUb li entra com a paràmetre l'Scanner, ja que el vam definir a la classe ClubUb perquè no donés cap error al crear l'Stream.

D'aquest treball, el que més podríem destacar, són els mètodes on tractem amb fitxers, amb els que hem treballat durant dies, fins i tot setmanes, i finalment hem pogut implementar-los. Hem trobat l'error que feia que no poguéssim guardar el club en l'arxiu. Teníem declarat un scanner a la classe llistaSocis i un cop eliminat aquest, el programa funciona correctament. No hagués estat possible sense l'ajuda del nostre professor de teoricopràctica. També podem destacar el tractament d'excepcions, que ens ha comportat bastanta feina ja que han sorgit certes dificultats que al final hem pogut tractar i superar a l'hora de crear-les i programar-les.

A part d'això, aquesta entrega ens ha semblat bastant més difícil que la primera, amb moltes més coses a tenir en compte, com els polimorfismes o les excepcions.