ClubUB

Programació II: Pràctica 2

Toni Garcia i Javier Pedragosa

# Classes

El projecte consta d’un total de 15 classes, dividides en 2 grups: model i vista.

Les classes del paquet vista són aquelles que permeten que la aplicació sigui útil, és a dir, són les que creen el menú i l’imprimeixen, agafen l’entrada per teclat de l’usuari...

La classe principal, o en altres paraules, aquella que s’executa quan executem el programa, és IniciadorClubUB, que l’única cosa que fa és crear una instancia de la classe VistaClubUB i executar gestioClubUB(), que és el bucle principal de la aplicació.

gestioClubUB() imprimeix el menú, contingut i administrat per la classe Menu, i s’encarrega d’executar la opció que entri l’usuari per teclat.

Per una altra banda, les classes del paquet model són aquelles que s’encarreguen del funcionament bàsic. D’aquesta manera, es podria dir d’una manera no gaire acurada que model és el “back end” i vista és el “front end”.

El paquet model està dividit en 3 grups: abstractes, atributs i socis, a més de dues classes independents a aquests 3 grups.

La més important és ClubUB, que és la que inclou tota la informació sobre el club, com ara el nom, les quotes i la llista de socis.

La llista de socis és una classe en si mateixa, ja que no només conté els socis, sinó que també té diversos mètodes que ens permeten una funcionalitat molt més completa.

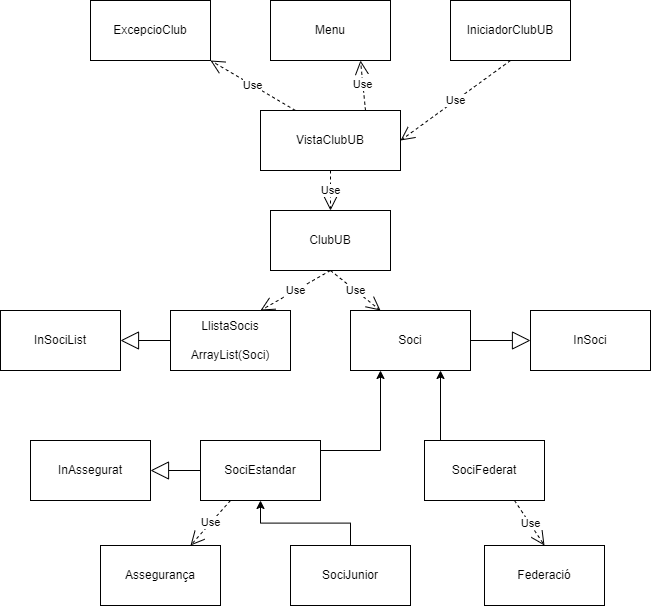
Els tres tipus de socis, continguts al grup de classes de socis, són estàndard, federat i júnior, i hereten de la classe Soci, que conté només els atributs més bàsics i els mètodes en la seva versió més fonamental. Soci, juntament amb les interfícies, forma part del grup de classes abstractes.

# Classe ClubUB

Els atributs de la classe ClubUB són els principals pilars del negoci que és el club d’excursionistes.

ClubUB conté el nom del club, el màxim de membres que admet el grup, la llista de socis, i tots els preus, quotes i descomptes definits.

# Diagrama de relacions



# Classe soci

Els atributs de la classe Soci són el nom i el DNI del soci, ja que son els únics atributs comuns a tots els socis. La resta d’atributs només els tenim en funció del tipus de soci.

# Perquè la classe Soci és abstracta

Per a que una classe sigui considerada com “abstracta” només cal que tingui un mètode declarat com abstracte. Que un mètode sigui abstracte vol dir que no el tenim implementat, només sap que existeix però no sap el que fa.

Si una classe és abstracta, necessàriament haurà de tenir alguna classe que hereti d’aquesta i que implementi aquest mètode abstracte. També podem tenir una classe filla que no implementi el mètode, però també serà abstracta.

# Necessitat de les 3 classes Soci

Si haguéssim treballat només amb la classe Soci i SociFederat no podríem haver donat als socis no federats un atribut del tipus assegurança sense que els federats l’haguessin de tenir.

De igual manera, necessitem crear una classe SociJunior per a poder atribuir-li una data de naixement a aquests i calcular de la forma apropiada el preu de les excursions.

# Mètodes polimòrfics

Un mètode polimòrfic és aquell que esta implementat diverses vegades gracies a la herència de classes. Un exemple d’això ho podem veure entre les classes SociEstandar i SociJunior. El mètode toString de la classe SociJunior crida al mateix mètode de la classe mare (SociEstandar) i lo afegeix al resultat el que correspon als atributs propis d’aquesta classe.

Un altre exemple es el mètode calculaPreuExcursio d’aquesta mateixa classe. Podem veure que en aquest cas no té res a veure amb el mètode de la classe mare sinó que simplement sobreescriu el mètode per el que cal.

# Objecte assegurança

El objecte de tipus assegurança el podríem crear perfectament dins del constructor de la classe SociEstandar, però fent-ho d’aquesta manera necessitaríem passar per paràmetres els paràmetres necessaris per a crear aquest objecte (tipus i preu).

Si creem l’objecte fora, i el passem per paràmetre, només passem 1 paràmetre en relació a aquest atribut i a més tenim l’objecte crear allà des d’on haguem cridat el constructor per si el necessitem usar.

# Classe ExcepcioClub

Al nostre codi la classe ExcepcioClub és fa servir per a llançar excepcions amb missatges adaptats al nostre programa. Això ho podem controlar des de l’origen als mètodes que nosaltres hem declarat e implementat, però hi ha mètodes i funcions pròpies de java que fem servir i que poden llençar excepcions de tipus general. Aquestes les capturem i les transformem en excepcions adaptades.

# Factura SociFederat

La factura d’un soci federat es calcula de la següent manera:

Calculem la quota mensual del soci, la qual son 30€ menys el descompte (30%) i un preu equivalent a la quantitat de excursions que ha fet el soci multiplicat per el preu de cadascuna (25€) i aplicant el descompte (20%)

Un soci federat que fa 5 excursions en un més ha de pagar 97.5€

# Factura SociEstandar

La factura d’un soci estàndar es calcula de la següent manera:

Calculem la quota mensual del soci, la qual son 30€ i un preu equivalent a la quantitat de excursions que ha fet el soci multiplicat per el preu de cadascuna (25€) i el preu de la assegurança.

Un soci estàndar que fa 5 excursions en un més amb una assegurança de 5€ ha de pagar 180€

# Proves realitzades

Les proves per assegurar el correcte funcionament del programa han estat les següents:

* Ens hem assegurat que totes les possibles entrades de dades errònies estiguin blindades.
* Hem provar a guardar i carregar les dades a fitxer.