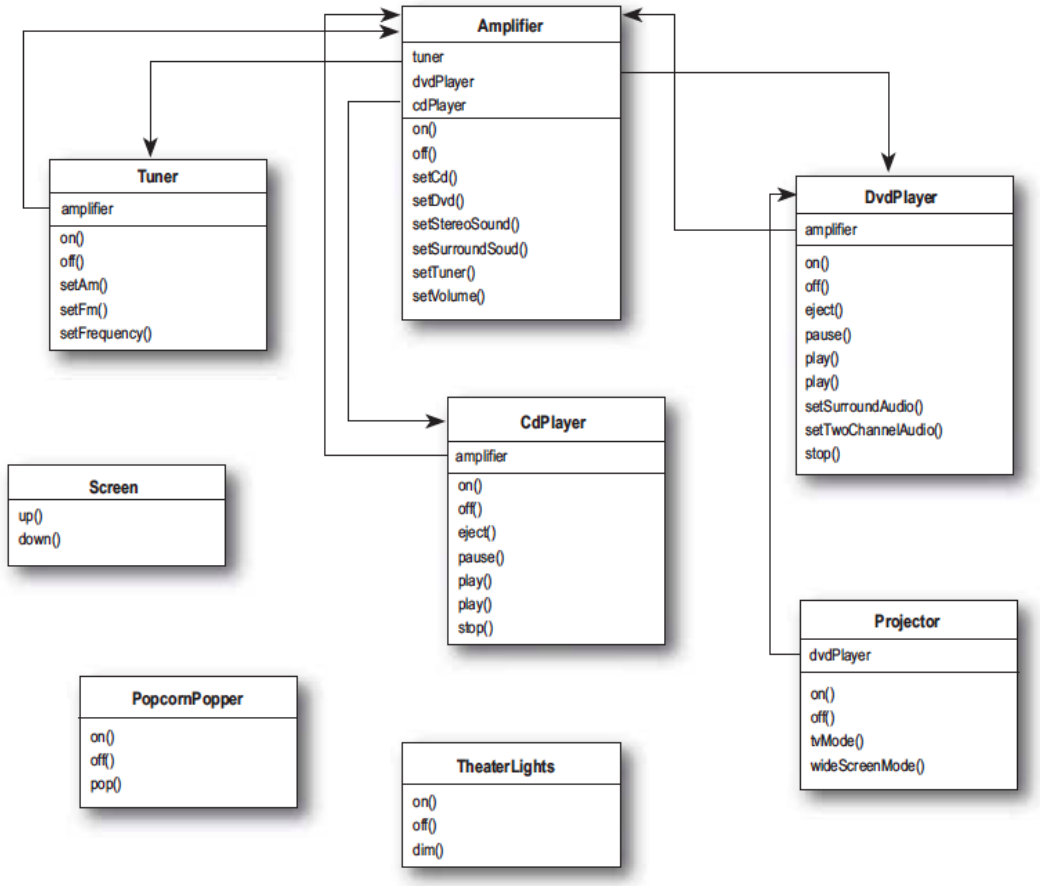
Exercicis de Patrons: Facade, Strategy

Exercici 1: Es vol modelar un HomeCinema al saló de casa. Després de setmanes desenvolupant el model de Domini i fent un primer Diagrama de classes, ha quedat el següent esquema:



Un conjunt de moltes classes interelacionades entre sí i amb moltes funcionalitats diferents. Un cop definides, es voldria fer la funcionalitat de posar una pel·lícula. Per això, cal:

1. Endollar l'aparell de crispetes (Popcorn Popper)
2. Engegar l'aparell de crispetes
3. Suavitzar les llums a un 10% (dim)
4. Baixar la pantalla de cine
5. Engegar el projector
6. Posar el input del projector al DVD
7. Posar el projector en pantalla ampla
8. Activar el volum del projector
9. Posar l'input de l'amplificador al DVD
10. Activar el so de l'amplificador a "Surround"

11. Posar el volum de l'amplificador a nivell mig (5)
12. Engegar el DVD
13. Fer play del DVD

Però encara hi han més requeriments: Quan s'acabi la pel·lícula, es voldrà apagar tot. Si avalues com de complex serà escoltar el CD o la ràdio, serà tan difícil com posar una pel·lícula? Quins principis de disseny S.O.L.I.D. s'estan vulnerant?

Pensa en el patró **Facade** com una alternativa per a fer les funcionalitats de:
1. Veure una pel·lícula
2. Acabar de veure una pel·lícula
3. Escoltar un cd
4. Acabar d'escoltar un cd
5. Escoltar la radio
6. Acabar d'escoltar la radio

Com es modificaria el teu diagrama de classes? Qui crearia la classe que fa de Façana? Qui té la responsabilitat de crear les classes internes del HomeCinema? Prova a fer la solució en el projecte Factory del campus problemes sessió 6.

Exercici 2: Es vol dissenyar una aplicació que simuli diferents tipus d'ànecs. Els ànecs naden (swim), alguns parlen (quack) i alguns volen (fly). Un dissenyador ha fet el disseny que teniu en el projecte 2 de IntellIJ del campus (sessió 6 de problemes). Com veureu hi ha una classe Duck, amb els mètodes quack(), swim(), fly() i display(). D'aquesta classe Duck, hereten les classes MallardDuck, ReadheadDuck, RubberDuck, DecoyDuck amb diferents implementacions dels mètodes fly() i quack(). Fixa't que quan volen, tots volen igual i, quan parlen, tots parlen igual. Identifiqueu quins principis S.O.L.I.D. vulnera. Com ho resoldríeu?

Mirant la solució que proposes, quantes classes caldria modificar si es vol fer algun canvi en la forma de volar? O permetre volar de forma diferent a algun d'ells? Es podria fer que canviessin la forma de volar quan s'executa el programa (de forma dinàmica)?

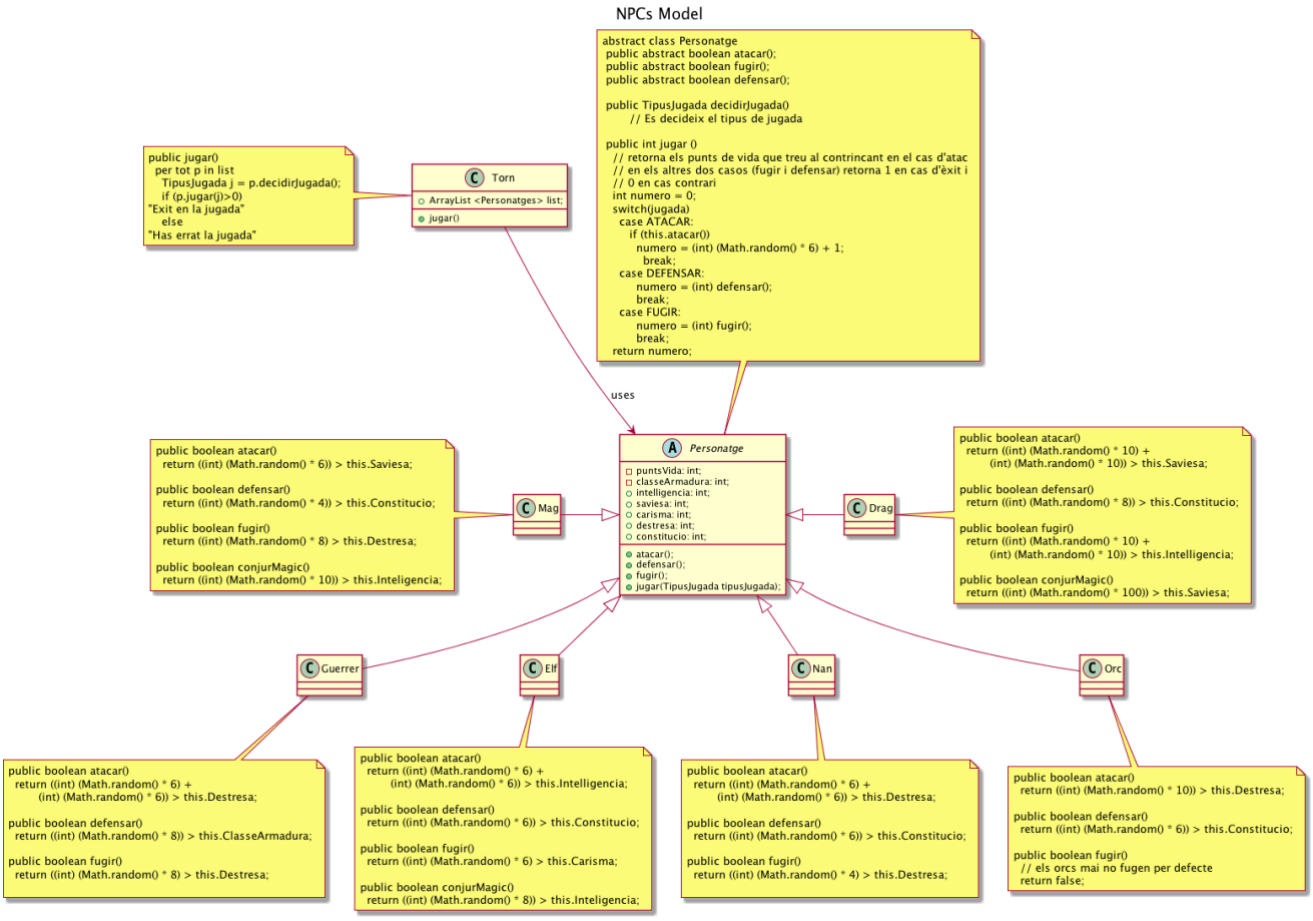
Si apliquéssiu el patró **Strategy**, com es faria el nou disseny de classes? Es podria variar el comportament de forma dinàmica? Modifica el projecte strategy del campus per aportar la teva solució.

Exercici 3: (Problema del segon parcial de DS 2017-18) En un joc de rol es volen dissenyar els diferents personatges que hi surten: guerrers, mags, elfs, nans, orcs i dragons. Tots els personatges tenen Punts de Vida i Classe d'Armadura i es caracteritzen per la seva Intel·ligència, Saviesa, Destresa, Carisma i Constitució. Els diferents personatges només poden atacar, defensar-se i fugir. Per això necessiten tirar un o varis daus - segons el tipus de personatge, el

dau tindrà diferent número de cares - i superar amb la/es tirada/es alguna de les característiques que tenen. Per exemple, si un guerrer vol atacar, s'haurà de llençar un dau de 6 cares dues vegades i la suma final de las tirades haurà de ser més gran que el valor de la seva Destresa. Si en canvi, un drag vol fugir ha de tirar un dau de 10 cares dos cops i la suma de les tirades haurà de ser superior al valor de la seva Intel·ligència. Només quan s'aconsegueix superar el valor, el personatge aconsegueix realitzar l'acció. Si està atacant amb èxit, podrà treure Punts de Vida al seu atacant (tants com un dau de 6). Si no s'aconsegueix superar el valor, el seu atac fallarà.

Al llarg del joc, però, els personatges poden trobar objectes, pujar de nivell o trobar conjurs que fan canviar els tipus d'atac, defensa o fugida, canviant el tipus de dau, tenint bonificacions especials en les característiques del personatge en certes tirades, o inclús canviant la manera de calcular l'acció. Per exemple, un Mag de nivell 2 no pot usar conjurs màgics per atacar però quan sigui de nivell 3, el seu atac es decidirà fent la tirada d'atac normal i la del conjur màgic.

Davant d'aquest problema, un dissenyador de software, preveient aquest funcionament del joc, ha fet el disseny següent:



a) Quins principis S.O.L.I.D. vulnera aquest codi? Raona la resposta.

b) Com redissenyaries aquest disseny? Quin/s patró/ns de disseny faries servir? Per a contestar aquest apartat omple els apartats següents.

b.1. Indica el nom del patró i tipus de patró (de creació, d'estructura o de comportament)

b.2. Aplicació del patró (Dibuixa el diagrama de classes obtingut després d'aplicar el patró i explica els detalls més rellevants del teu disseny)

b.3. Anàlisi del patró aplicat en relació als principis S.O.L.I.D.

b.4. Programa jugar() que mostra l'ús del patró utilitzat

b.5. Observacions addicionals

<A partir de la semana del 25.11> c) En una segona versió del joc es volen fer atac mitjançant tropes de personatges. Una tropa es defineix com el conjunt de diferents flancs, al marge que pot tenir personatges que no estiguin a cap flanc. En un flanc sempre hi ha com a mínim un personatge. Quin patró utilitzaries per a definir les tropes? Raona la teva resposta en 10 línies com a màxim.