FINAL GENER

Assignatura: **Disseny de Software**

Data: 16 de Gener de 2018 Curs: **2017/2018**

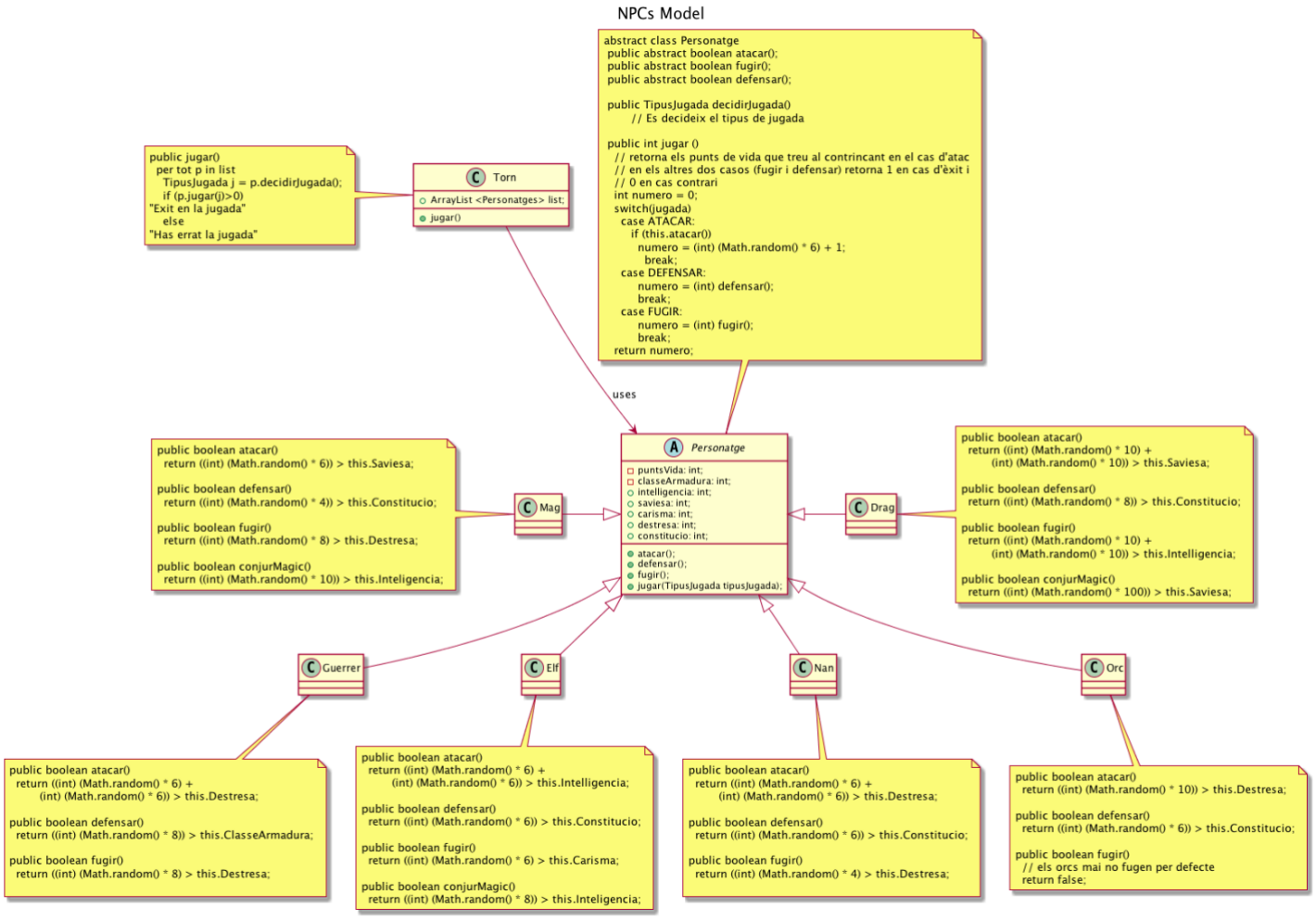
UNIVERSITAT DE BARCELONA

Problema (60 punts): Temps estimat: 2 hores

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 drac 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.

S: No el vulnera, tot i que hi hagin diferents mètodes, la classe té una única responsabilitat. Es pot raonar que el vulnera si es diu que la classe està donant la responsabilitat de decidirJugada i la responsabilitat de Jugar.

O: Es vulnera clarament. Quan es vulgui fer un altre tipus d'acció, s'ha de modificar la classe Personatge, tot i que es diu en l'enunciat que NOMES pot atacar, defensar i fugir i el que realment canviarà serà la manera de fer l'acció o bé que se'n podran fer de combinades. Si es considera que no pot haver més que atacar, defensar i fugir, aquest principi no el vulnera.

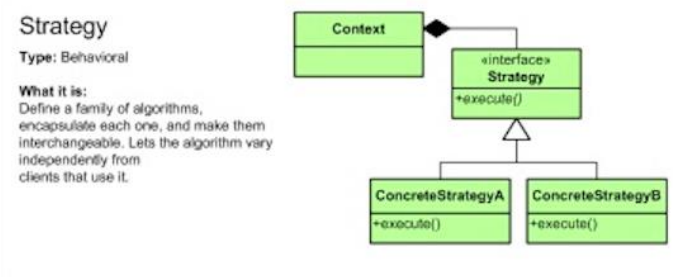
L: No el vulnera. Totes les subclasses fan un comportament esperat per una classe Personatge, tot i que es pot argumentar que els Orcs no fugen mai, però es el seu comportament. No es que no ho sàpiguen fer.

I: No aplica

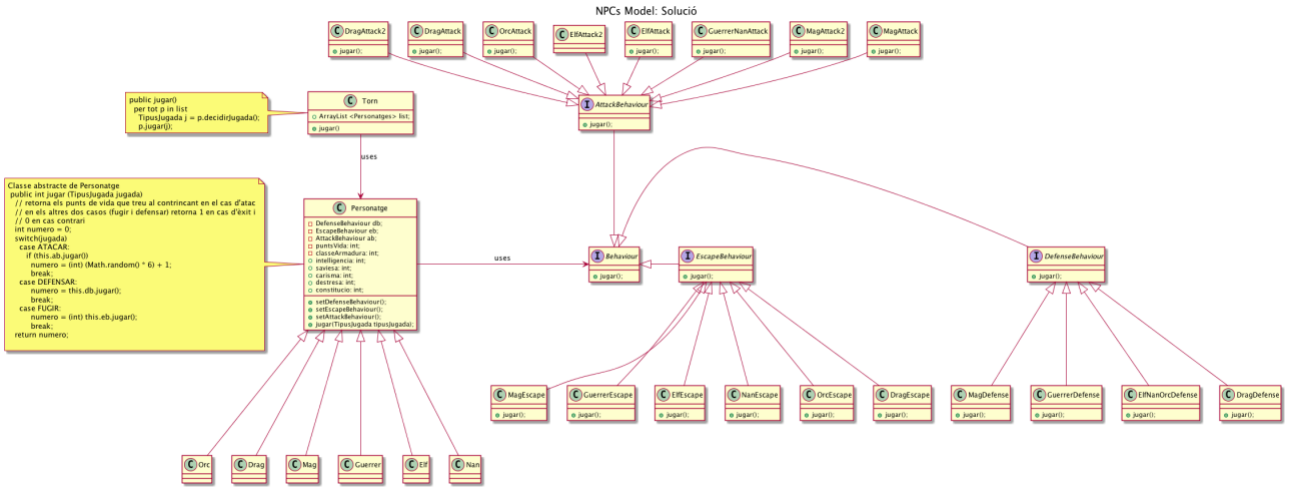
D: Es pot veure dependència de les accions de cada personatge en implementacions no concretes i estàtiques. Bàsicament és aquesta vulneració la que porta a aplicar el patró Strategy

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

b.1. Nom del patró i tipus de patró: **Strategy. Patró de comportament.**



b.2. Aplicació del patró:



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

Igualment es vulnera el Open-Closed Principle que no es pot evitar. Es podria acabar treient de Personatge el switch fent un array de 3 Behaviours en el Personatge i que decidirJugada ens doni l'índex del tipus de Behaviour que es vulgui fer en aquell moment. Tot i que es treu la vulneració de la classe Personatge, igualment tindrem el problema d'Open-Closed en el decidirJugada.

En el mètode jugar, de la classe Personatge, es separa la decisió de l'èxit de l'atac i el dany que fa en atacar. Aquí es clarament una vulneració del S ja que s'ataca i es calcula el dany. El mètode jugar s'hauria de partir en dues parts: una per saber si hi ha èxit i l'altra per fer mal.

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

```
public int jugar (TipusJugada jugada)
   // retorna els punts de vida que treu al contrincant en el cas d'atac
   // en els altres dos casos (fugir i defensar) retorna 1 en cas d'èxit i
   // 0 en cas contrari
   int numero = 0;
   switch(jugada)
     case ATACAR:
        if (this.ab.jugar())
         numero = (int) (Math.random() * 6) + 1;
         break;
     case DEFENSAR:
          numero = this.db.jugar();
         break;
     case FUGIR:
          numero = (int) this.eb.jugar();
         break;
   return numero;
```

b.5. Observacions addicionals:

El tema del Open-Closed no està solucionat. A part a la part de crear els diferents Personatges es pot aplicar una AbstractFactory que instanciï la forma d'atacar, de defensar i de fugir de cada personatge.

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.

Aquí aplicaríem el patró Composite per a poder fer aquest composició de conjunts formats per més conjunts o elements aïllats, on els Components són la tropa/flanc que estaries formats per flancs i les Leafs serien els personatges.