Pt11.3 Interfaces

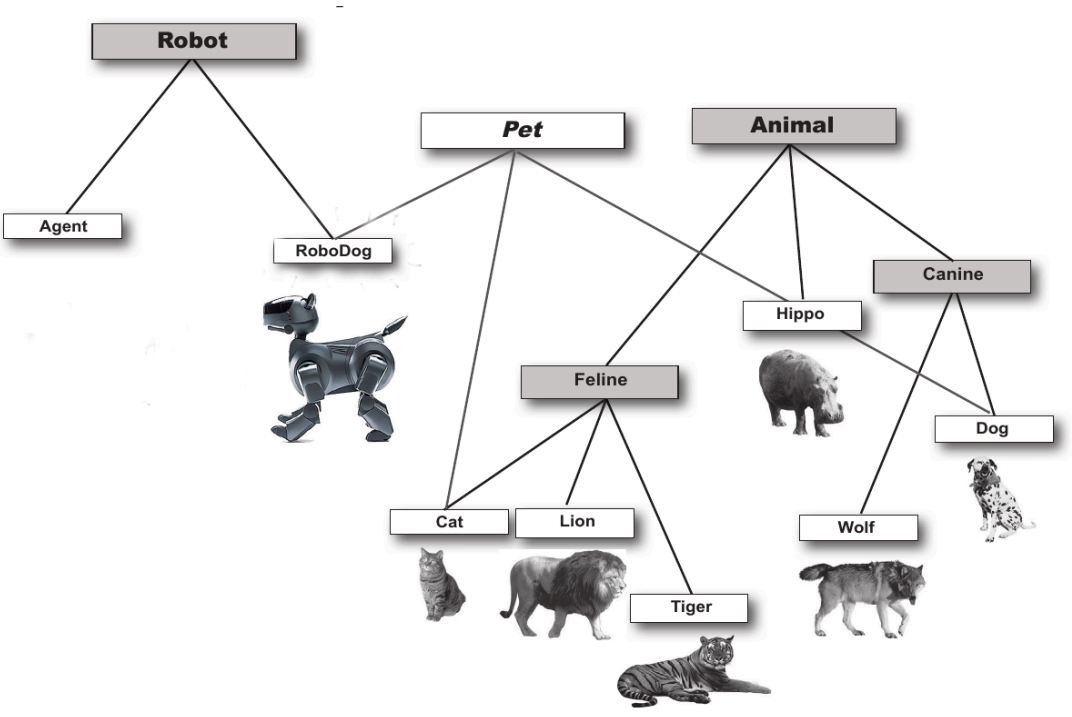
1. Animals i mascotes

En aquest exercici no s'ha d'implementar res, només us demanem que penseu com resoldre el següent problema.

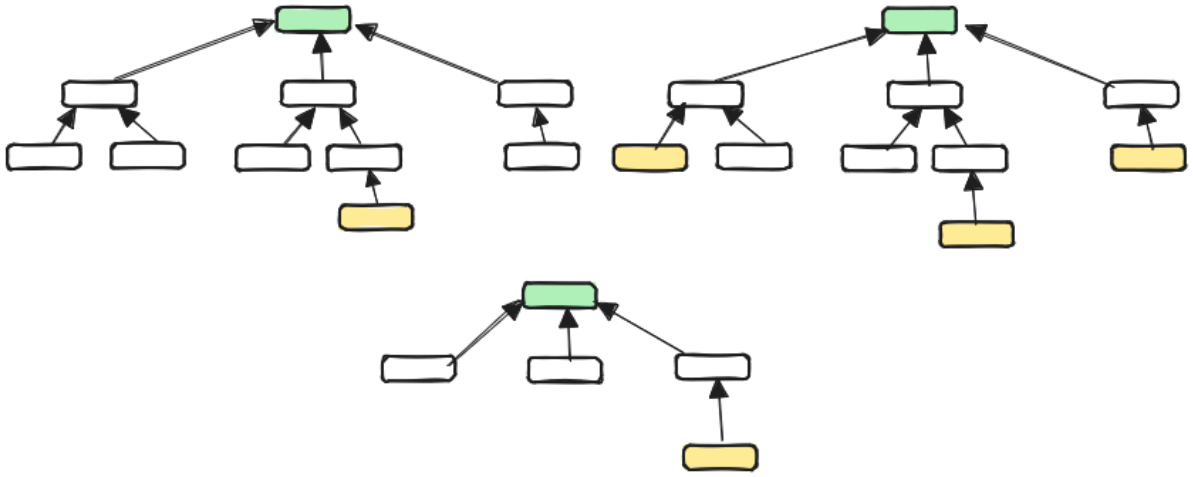
Tenim un videojoc amb dos diferents tipus de personatges, *Robots* i *Animals,* amb la jerarquia que es mostra a la figura, on els rectangles en gris són classes abstractes. *Robots i Animals* comparteixen una sèrie d'atributs i comportaments (mètodes), com *move(), interact(),* etc, i es van desplaçant de manera aleatòria pel mapa del videojoc. Són el que diem non-playable characters (NPC), és a dir, personatges que interactuen en el videojoc però que no són manejats per cap jugador.

Quan s'inicia el joc, internament es creen dues llistes de *Robot i Animals,* amb un nombre aleatori de robots i animals que es mouen pel mapa del videojoc. A cada frame, es recorren aquestes liistes i es criden als mètodes *move() i interact()* (en cas de que es detecti un altre personatge al voltant).

Ara volem afegir un nou tipus de personatge anomenat *Pet* amb un comportament específic de les mascotes, per exemple, volem afegir nous comportaments com *play() i feed(),* propis de les mascotes. A més, el jugador principal *Player()* tindrà una llista de mascotes que s'anirà actualitzant durant el joc.
En definitiva, necessitem que *Player()* tingui com atribut una llista de tipus List<Pet> pets.
Com dissenyaries aquest sistema?



2. Report System



En aquest exercici no s'ha d'implementar res, només us demanem que penseu com resoldre el següent problema.

Tenim un sistema amb diversos dominis tal com mostra la figura. Ara ens demanen que implementem una nova funcionalitat per les classes en grog. També, s'espera que un futur haguem de incloure més classes amb aquesta funcionalitat . Es tracta d'un mètode que ha de ha de retornar un String amb un petit report per pantalla, una mica més personalitzat que toString().
Com ho faries? quines alternatives veus per resoldre aquesta nova funcionalitat?

3. Library System

Escriu un programa per a una biblioteca que contingui llibres i revistes. És necessari la creació d'una classe abstracta que aglutini les característiques comunes.

Les característiques comunes que s'emmagatzemen tant per a les revistes com per als llibres són el codi, el títol, i l'any de publicació. Aquestes tres característiques es passen per paràmetre en el moment de crear els objectes.

Els llibres tenen a més un atribut prestat. Els llibres, quan es creen, no estan prestats.

Les revistes tenen un nombre. En el moment de crear les revistes es passa el nombre per paràmetre.

Tant les revistes com els llibres han de tenir (a part dels constructors) un mètode toString () que retorna el valor de tots els atributs en una cadena de caràcters. També tenen un mètode que retorna l'any de publicació, i un altre el codi.

Per prevenir possibles canvis en el programa s'ha d'implementar una interfície Prestable amb els mètodes:

prestar(), posarà el valor prestat a true.

tornar(), posarà el valor prestat a false.

prestat() que retorna el valor de prestat.

La classe Llibre implementa aquesta interfície.

Dibuixa el diagrama de classes i implementa l'estructura.

4. Pizzeria

En una pizzeria tenim un sistema de comandes de menjar i un menú que consta de diversos articles (com ara pizzes, hamburgueses i begudes). Alguns articles, com ara les pizzes o els batuts, permeten personalitzacions addicionals com ara formatge, picant, anxoves (pizzes) o menta, fruits del bosc, etc (batuts), mentre que d'altres no. Aquests personalitzables permeten afegir ingredients ja pre-establerts.

Com a mínim necessitem, una classe pare amb atributs id, name i price que sigui no instanciable. Classes per presentar productes de menjar, amb subclasses pizzes (mida), burgers (tipus) i una altra classe per begudes, que hereden de la classe pare. Una classe per representar els batuts (tipus de llet), que hereta les propietats de begudes. Els ítems que permeten personalitzacions són les pizzes i els batuts (shake) i necessiten un mètode per poder afegir ingredients, la resta no.

Una classe per gestionar comandes que ha de tenir una llista de ítems i mètodes per poder afegir, esborrar i llistar ítems, i un mètode que ens retorna una lista de ítems personalitzats.

Dissenya e implementa aquest sistema. Codi net amb un bon naming i en anglès.

5. Cos Geomètric

Refés l'activitat del Cos Geomètric de forma que tinguem tres interfícies, una que s'anomeni Constants i guardi el nombre PI, una altra interfícies que s'anomeni Calculs2D i que tindrà dos mètodes definits per a l'àrea i el perímetre. I finalment una tercera interfície que hereti de Calculs2D i s'anomeni Calculs3D que tingui un mètode pel volum.

Per altra banda genera una classe NO instanciable que sigui FiguraGeomètrica i contingui un idFigura. A aquesta classe cal que tingui dues especialitzacions: Figures2D, Figures3D també NO instanciables. Per a les Figures2D defineix dues classes que sí siguin instanciables Cercle i Rectangle. Per a les Figures3D implementa dues classes instanciables Cilindre i una Piràmide. Inclou tots els atributs necessaris a cada classe instanciable.

Les figures circulars necessitaran la interfície Constants, les figures 2D necessitaran també la interfície de Càlculs2D i les figures 3D necessitaran també la interfície càlculs 3D.

Fes el diagrama de classes, implementa'n una solució i prova-la en una classe de proves. Realitza un ArrayList que mostri les figures2D i un altre que mostri les figures3D.

6. Herència múltiple

Busca com solucionaries el problema de l'herència múltiple en Java. Redefineix el següent diagrama de classes fent ús d'interfícies i implenta'n l'estructura de classes.