Laberint estès

A la sessió anterior ("Laberint bàsic") vau implementar les classes que composen el nucli central de l'aplicació del laberint. Com recordareu, aquestes classes permeten a l'usuari/a controlar l'aventurer del joc, movent-lo a través de les habitacions del laberint, tot indicant-li d'agafar (i deixar) ítems d'un cert valor. Aquesta sessió assumeix que teniu ja implementada una versió funcional del laberint bàsic, el qual estendrem amb noves classes i funcionalitats fent ús dels conceptes d'herència i polimorfisme.

Objectius

Els objectius d'aquesta sessió són:

- Comprendre la utilitat dels conceptes d'herència i polimorfisme en Java i ser capaç d'aplicar-los en una aplicació de complexitat moderada.
- Ser capaç d'afegir noves funcionalitats a una aplicació prèviament existent.

Material que s'adjunta

El material que s'adjunta per a aquesta sessió és el següent:

- Aquest enunciat, amb el diagrama de classes UML d'aquesta sessió.
- El codi ampliat de les classes **UserInterface** i **Controller**.
- El *javadoc* complet d'aquesta sessió.
- Un exemple de funcionament: "ElementsSecrets.txt".

Explicacions prèvies

Tal i com hem esmentat, el punt d'inici d'aquest exercici és exactament el final de la sessió anterior. Ara volem afegir un nou tipus d'elements a les sales del laberint (a més dels ítems que ja teníem) amb els quals l'aventurer pugui interactuar.

En particular, volem introduir dos tipus d'elements diferents: els *contenidors secrets* i les *portes secretes*. Els elements de tipus contenidor amaguen un ítem a dins. Un exemple de contenidor podria ser un "cofre" situat en una de les sales del laberint, i l'ítem que conté podria ser una "corona d'or". Quan activem el contenidor "cofre", el contenidor queda inservible, i la "corona d'or" apareix com un nou ítem, visible a la sala. Per activar el contenidor, l'aventurer necessita un item concret que fa de clau (p.ex., per activar el "cofre" el jugador necessita l'item "estelactita").

En canvi, els elements de tipus porta secreta amaguen una sortida secreta, inicialment invisible, que connecta la sala on estan situats amb una altra de les sales del laberint. Per exemple, un "forat" podria ser una porta secreta, que amaga un accés cap a la sala "túnel". Quan activem la porta secreta, aquesta queda inservible i a la llista de sortides de la sala apareix una nova sortida que té per nom el nom de la porta secreta.

La nova sortida ("forat") permet ara accedir a la sala "túnel". Per activar la porta secreta "forat" cal utilitzar l'ítem "roc", que fa de clau.

Contenidors i portes secretes són dos tipus particulars d'un concepte més general. Ambdós són *elements secrets*.

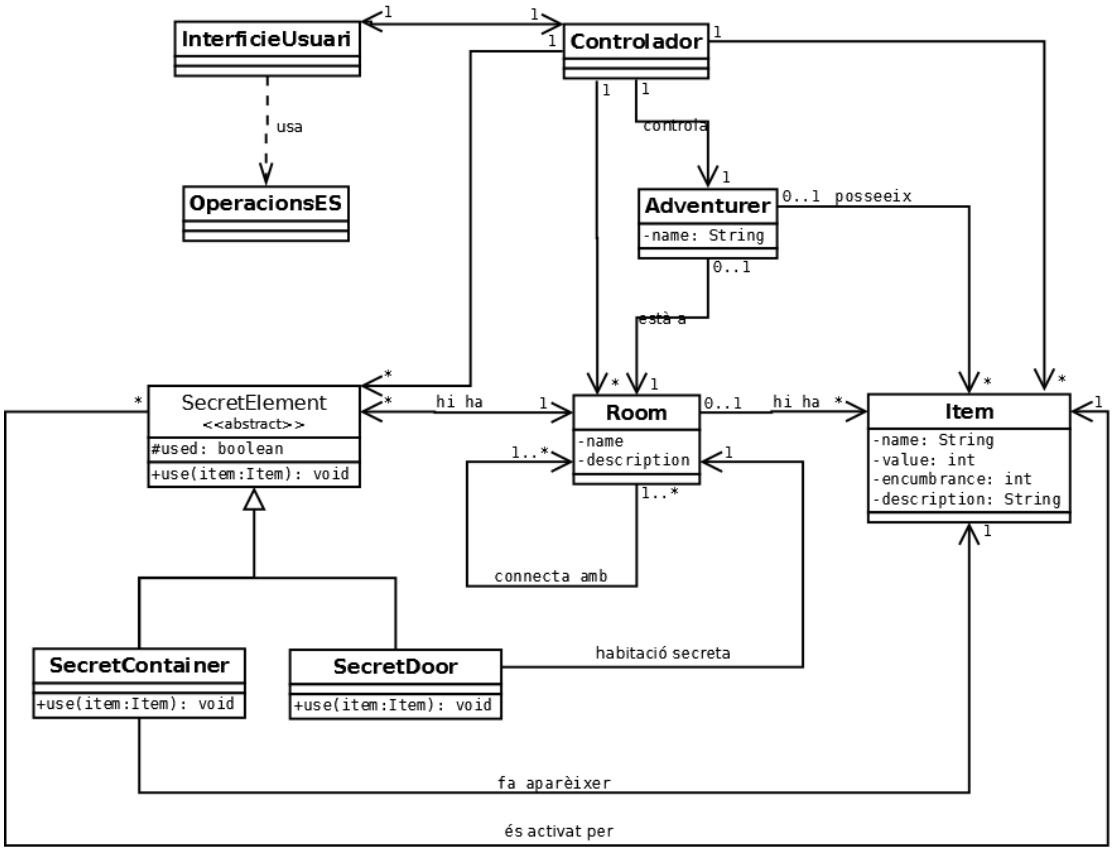
La interacció de l'aventurer amb els elements serà la mateixa, tant per als contenidors com per a les portes secretes: l'aventurer *utilitzarà* un dels ítems del seu inventari per activar l'element (aquest ítem és la clau que activa l'element). Sí l'ítem utilitzat no és l'adequat, és a dir, no és l'ítem clau, l'acció no té cap conseqüència i el programa mostra un missatge d'error a l'usuari/a.

En canvi, en cas que l'ítem utilitzat sigui el que l'element accepta (la clau), els dos tipus d'elements, com hem vist, es comporten de forma diferent: els contenidors fan que l'ítem secret que guardaven aparegui explícitament a la sala on estan situats, i les portes secretes fan que la porta amagada aparegui explícitament a la llista de sortides de la sala.

Un cop s'ha activat correctament un element, sigui del tipus que sigui, futures utilitzacions de l'ítem no tindran cap mena d'efecte (no podem obrir un cofre que ja ha estat obert, o descobrir una porta secreta que ja ha estat descoberta!).

Cal tenir en compte que la interfície textual d'usuari **NO** mostrarà els elements presents a la sala actual, com ho feia en el cas dels ítems, sino que la seva presència s'haurà de deduir a partir de la descripció de l'habitació mostrada. I caldrà també endevinar quin ítem s'ha d'utilitzar com a clau.

A continuació, es mostra el diagrama de classes UML d'aquesta sessió:



Tasques a realitzar

En aquesta sessió s'introduiran de forma progressiva les diferents extensions que haureu d'aplicar al codi del laberint bàsic per tal d'implementar totalment les funcionalitats esmentades anteriorment. Com a documentació, teniu el *javadoc* de l'aplicació i les classes **UserInterface** i **Controller** ja actualitzades.

Apartat 1: Implementació de la jerarquia d'herència Element

1.1) Comenceu a implementar les 3 noves classes que hem introduït al diagrama de classes UML (**SecretElement**, **SecretContainer** i **SecretDoor**). Recordeu que els mètodes constructors de les subclasses han d'inicialitzar tant els atributs propis de la subclasse, com els heretats de la seva superclasse ("super()").

1.2) Implementeu el mètode "equals()" de la classe **Item**.

1.3) Implementeu el mètode polimòrfic

- `int use(Item clau)`

tant a la superclasse **SecretElement** com a cadascuna de les subclasses **SecretContainer** i **SecretDoor**, de tal forma que s'aconsegueixi la funcionalitat descrita anteriorment per a cada tipus d'element. Fixeu-vos que aquest mètode l'invocaria l'aventurer sobre l'element en qüestió i li passaria com a paràmetre

l'ítem que vol usar sobre aquest element (la clau). Llavors l'element actuaria en conseqüència. Creieu que el mètode hauria de ser abstracte a nivell de superclasse **SecretElement**? Per què?

Els codis de retorn seran **OK_ACTION** si s'ha pogut activar l'element; **ELEMENT_USED**, si l'element ja ha estat utilitzat prèviament; i **WRONG_KEY** si l'item utilitzat per activar l'element no és la clau adequada.

1.4) Implementeu l'/els atribut/s necessari/s a la classe **Room** per a que es puguin guardar els elements en una habitació. A més, implementeu "putElement()" i "getElement()".

1.5) Implementeu el mètode "useItemOnElement()" de la classe **Adventurer**.

Heu de realitzar-ho tot de tal forma que s'obtinguin realment els beneficis del polimorfisme en Java.

Apartat 2: Inicialització del laberint amb elements

Les classes **UserInterface** i **Controller** se us donen fetes, i la nova versió del mètode "createDungeon()" inclou algun exemple d'element secret. Si voleu, podeu estendre més encara el laberint modificant aquest mètode.

Apartat 3: Proves

Per facilitar fer proves, al principi de la classe **Room** podeu declarar una constant

private static final boolen SHOW_SECRET_ELEMENTS = false ;

Si la posem a true, el mètode "toString()" de la classe **Room** ha de retornar també els noms dels elements secrets que hi hagi a la sala (per mostrar-los per pantalla quan l'aplicació mostra el que hi ha a la sala actual).