A6.E2: herència

Instruccions
Creeu el projecte **A6E2LabyrinthProject** amb els *packages* **labyrinth.tools**, **labyrinth.models**, **labyrinth.controller** i **labyrinth.main**.

Per entregar l'exercici només haureu d'entregar la carpeta *src* comprimida en zip.

En tots els exercicis cal tenir en compte els següents aspectes importants:
1. Tota interacció amb l'usuari es farà des del fitxer main (petició i presentació de dades). Dins del main, però, podeu fer funcions que interactuïn amb l'usuari.
2. Els atributs i els mètodes de la classe han de tenir la visibilitat adequada per garantir l'encapsulació i la protecció de les dades.

Exercici 1.
Creeu la jerarquia **Cell** dins del *package* **labyrinth.models**. Aquesta jerarquia representa les diverses cel·les del laberint que ha de traspassar l'usuari i està formada per les classes següents:
- Superclasse **Cell**
- Subclasses
  - **NormalCell**
  - **NormalEnterCell**
  - **NormalExitCell**
  - **BombCell**
  - **MonsterCell**
  - **PowerUpCell**
  - **TeleportationCell**
  - **TrapCell**
  - **WellCell**

SUPERCLASSE Cell
Atributs
De cada cel·la se'n vol guardar les dades següents:
- Posició del laberint on es troba (fila i columna): `row` i `col`
- Tipus de cel·la (aquest atribut ha de ser de tipus CellType): `type`
- Si la cel·la ja ha estat oberta (l'usuari hi ha passat algun cop): `opened`
- Missatge generat quan l'usuari traspassa la cel·la: `traverseMessage`

Mètodes
- **Constructor per defecte**
  Inicialitza una cel·la tancada, a la posició de la cel·la a (-1, -1), amb un missatge de traspàs buit i tipus `null`.
- **Constructor parametritzat**
  Rep per paràmetre la posició de la cel·la i crea una cel·la tancada en aquella posició, amb un missatge de traspàs buit i tipus `null`.
- **Getters dels paràmetres `type` i `traverseMessage`**
- **`void openCell()`**
  Funció que obrirà la cel·la (posarà el paràmetre `opened` a `true`)
- **`boolean isOpen()`**
  Funció que retornarà `true` si la cel·la ja està oberta i `false` en cas contrari.
- **`boolean traverse(Player p)`**
  Funció abstracta (s'haurà d'implementar a les subclasses)
- **`String toString()`**
  Retornarà la representació d'aquesta cel·la en format `String`. De cada cel·la se'n vol conèixer la posició i el tipus (sempre que aquest no sigui `null`)
- **`boolean equals(Object obj)`**
  Retornarà cert si aquesta (`this`) cel·la és igual a la passada per paràmetre (`obj`).

SUBCLASSE NormalCell
Representa una cel·la normal (`CellType.NORMAL`), que no afecta a l'usuari.

Mètodes

- **Constructor per defecte**
  Crida al constructor més adient de la super classe per crear una cel·la buida de tipus `CellType.NORMAL`.
- **Constructor parametritzat**
  Rep per paràmetre la posició de la cel·la.
  Crida al constructor més adient de la super classe per crear una cel·la buida, de tipus `CellType.NORMAL` a la posició indicada per paràmetre.
- **`boolean traverse(Player p)`**
  Aquesta funció s'encarrega d'obrir la cel·la (cridar a la funció de la superclasse corresponent a aquesta acció) i d'inicialitzar el `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "No afecta al jugador":

  Cel·la [0, 1] - Tipus NORMAL
  No afecta al jugador

  Per acabar, aquesta funció sempre retorna `true`.

SUBCLASSE NormalEnterCell
Representa la cel·la d'entrada al laberint (`CellType.NORMAL_ENTER`), que no afecta a l'usuari.

Mètodes

- **Constructor per defecte**
  Crida al constructor més adient de la super classe per crear una cel·la buida de tipus `CellType.NORMAL_ENTER`.
- **Constructor parametritzat**
  Rep per paràmetre la posició de la cel·la.
  Crida al constructor més adient de la super classe per crear una cel·la buida, de tipus `CellType.NORMAL_ENTER` a la posició indicada per paràmetre.
- **`boolean traverse(Player p)`**
  Aquesta funció s'encarrega d'obrir la cel·la (cridar a la funció de la superclasse corresponent a aquesta acció) i d'inicialitzar el `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "Accés al laberint":

  Cel·la [0, 0] - Tipus NORMAL_ENTER
  Accés al laberint

  Per acabar, aquesta funció sempre retorna `true`.

SUBCLASSE NormalExitCell
Representa la cel·la de sortida del laberint (`CellType.NORMAL_EXIT`), que no afecta a l'usuari.

Mètodes

- **Constructor per defecte**
  Crida al constructor més adient de la super classe per crear una cel·la buida de tipus `CellType.NORMAL_EXIT`.
- **Constructor parametritzat**
  Rep per paràmetre la posició de la cel·la.

Crida al constructor més adient de la super classe per crear una cel·la buida, de tipus `CellType. NORMAL_ EXIT` a la posició indicada per paràmetre.

- **`boolean traverse(Player p)`**
  Aquesta funció s'encarrega d'obrir la cel·la (cridar a la funció de la superclasse corresponent a aquesta acció) i d'inicialitzar el `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "Sortida del laberint":

  Cel·la [4, 2] - Tipus NORMAL_ENTER
  Sortida del laberint

  Per acabar, aquesta funció sempre retorna `true`.

SUBCLASSE BombCell
Representa la cel·la de tipus bomba del laberint (`CellType.BOMB`). Pot afectar a l'usuari.

Atributs
De cada cel·la de tipus bomba se'n vol saber si la bomba que conté està activada o desactivada: `enabled`

Mètodes

- **Constructor per defecte**
  Crida al constructor més adient de la super classe per crear una cel·la buida de tipus `CellType.BOMB` i amb la bomba activada.
- **Constructor parametritzat**
  Rep per paràmetre la posició de la cel·la.
  Crida al constructor més adient de la super classe per crear una cel·la buida, de tipus `CellType.BOMB`, amb la bomba activada i a la posició indicada per paràmetre.
- **`void disableBomb()`**
  Aquesta funció és privada i desactiva la bomba que conté la cel·la (posa `enabled` a `false`)
- **`boolean traverse(Player p)`**
  Si la cel·la ja està oberta, inicialitza el `traverseMessage` amb el missatge "Ja s'havia passat per aquesta cel·la" i retorna `true`. Si no:
  - si el jugador té el `PowerUp WATER_BALLON`, desactiva la bomba, obre la cel·la, inicialitza el missatge `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "S'ha desactivat i el jugador s'ha salvat" i retorna `true`;
  - si el jugador no té el `PowerUp WATER_BALLON`, només inicialitza el missatge `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "No s'ha desactivat. BADABUUUUUUMMMM!!!!!!" i retorna `false`.

SUBCLASSE MonsterCell
Representa la cel·la que conté un monstre del laberint (`CellType.MONSTER`). Pot afectar a l'usuari.

Atributs
De cada cel·la de tipus monstre se'n vol saber si el monstre que conté està adormit o no: `monsterAsleep`

Mètodes

- **Constructor per defecte**
  Crida al constructor més adient de la super classe per crear una cel·la buida de tipus `CellType.MONSTER` i amb el monstre despert.
- **Constructor parametritzat**
  Rep per paràmetre la posició de la cel·la.
  Crida al constructor més adient de la super classe per crear una cel·la buida, de tipus `CellType.MONSTER`, amb el monstre despert i a la posició indicada per paràmetre.
- **`void tameTheBeast()`**
  Aquesta funció és privada i adorm al monstre que conté la cel·la (posa `monsterAsleep` a `true`)
- **`boolean traverse(Player p)`**
  Si la cel·la ja està oberta, inicialitza el `traverseMessage` amb el missatge "Ja s'havia passat per aquesta cel·la" i retorna `true`. Si no:
  - si el jugador té el `PowerUp FLUTE` adorm al monstre, obre la cel·la, inicialitza el missatge `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "S'ha amansat la fera i el jugador pot continuar de puntetes" i retorna `true`;
  - si el jugador no té el `PowerUp FLUTE`, només inicialitza el missatge `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "No s'ha amansat la fera i el jugador és el sopar del monstre!!!" i retorna `false`.

SUBCLASSE PowerUpCell
Representa la cel·la que dóna a l'usuari un poder (`CellType.POWERUP`).

Atributs.
De cada cel·la de tipus poder se'n vol saber quin `PowerUp` dóna a l'usuari: `power`.

Mètodes

- **Constructor per defecte**
  Crida al constructor més adient de la super classe per crear una cel·la buida de tipus `CellType.POWERUP` i amb un poder `null`.
- **Constructor parametritzat**
  Rep per paràmetre la posició de la cel·la.
  Crida al constructor més adient de la super classe per crear una cel·la buida, de tipus `CellType.POWERUP`, amb el poder `null` i a la posició indicada per paràmetre.
- **`PowerUp getPowerUp()`**
  Aquesta funció és privada i, en cas que `power` sigui `null`, inicialitza l'atribut de manera aleatòria i en retorna el valor.
- **`boolean traverse(Player p)`**
  Si la cel·la ja està oberta, inicialitza el `traverseMessage` amb el missatge "Ja s'havia passat per aquesta cel·la" i retorna `true`. Si no, obre la cel·la, dóna el poder que conté a l'usuari, inicialitza el missatge `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "Ha concedit al jugador el poder XXX" i retorna `true`.

SUBCLASSE TeleportationCell
Representa la cel·la que teletransporta (`CellType.TELEPORTATION`) a l'usuari a una altra cel·la del laberint (radi màxim de 5 caselles).

Atributs.
De cada cel·la de tipus teletransport se'n vol saber la posició de la cel·la on mou l'usuari: `rowTranslation` i `colTranslation`.

Mètodes

- **Constructor per defecte**
  Crida al constructor més adient de la super classe per crear una cel·la buida de tipus `CellType.TELEPORTATION`, inicialitzant `rowTranslation` i `colTranslation` a (-1, -1).
- **Constructor parametritzat**
  Rep per paràmetre la posició de la cel·la.
  Crida al constructor més adient de la super classe per crear una cel·la buida, de tipus `CellType.TELEPORTATION`, inicialitzant `rowTranslation` i `colTranslation` a (-1, -1) i a la posició indicada per paràmetre.
- **`int[] teleportation()`**
  Aquesta funció és privada i inicialitza `rowTranslation` i `colTranslation` a valors aleatoris dins del rang [0, 4]. Un cop inicialitzats els atributs, retorna un array de dos posicions amb els valors de la posició on s'enviarà a l'usuari.
- **`boolean traverse(Player p)`**
  Aquesta cel·la SEMPRE afecta a l'usuari (estigui oberta o no).
  Genera una nova posició de teletransport, mou l'usuari a la nova posició i inicialitza el missatge `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "El jugador està viatjant...". A més a més, si la cel·la està tancada, s'obre. Finalment, retorna `true`.

SUBCLASSE TrapCell
Representa la cel·la de tipus trampa (`CellType.TRAP`) dins del laberint. Pot afectar a l'usuari.

Atributs.
De cada cel·la de tipus trampa se'n vol saber si s'ha pogut tallar la corda o no: `ropeCutted`.

Mètodes

- **Constructor per defecte**
  Crida al constructor més adient de la super classe per crear una cel·la buida de tipus `CellType.TRAP`, inicialitzant `ropeCutted` a `false`.
- **Constructor parametritzat**
  Rep per paràmetre la posició de la cel·la.
  Crida al constructor més adient de la super classe per crear una cel·la buida, de tipus `CellType.TRAP`, inicialitzant `ropeCutted a false a` i a la posició indicada per paràmetre.
- **`void cutTheRope()`**
  Aquesta funció és privada i talla la corda de la trampa.
- **`boolean traverse(Player p)`**
  Si la cel·la ja està oberta, inicialitza el `traverseMessage` amb el missatge "Ja s'havia passat per aquesta cel·la" i retorna `true`. Si no:
  - si el jugador té el `PowerUp KNIFE`, talla la corda, obre la cel·la, inicialitza el missatge `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "S'ha pogut tallar la corda i el jugador pot escapar" i retorna `true`;

si el jugador no té el `PowerUp KNIFE`, només inicialitza el missatge `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "No s'ha pogut tallar la corda i el jugador queda penjat cap per avall!!!!" i retorna `false`.

SUBCLASSE WellCell
Representa la cel·la de tipus pou (`CellType.WELL`) dins del laberint. Pot afectar a l'usuari.

Atributs.
De cada cel·la de tipus pou se'n vol saber si s'ha pogut saltar el pou o no: `jumped`.

Mètodes

- **Constructor per defecte**
  Crida al constructor més adient de la super classe per crear una cel·la buida de tipus `CellType.WELL`, inicialitzant `jumped` a `false`.
- **Constructor parametritzat**
  Rep per paràmetre la posició de la cel·la.
  Crida al constructor més adient de la super classe per crear una cel·la buida, de tipus `CellType.WELL`, inicialitzant `jumped` a `false` a i a la posició indicada per paràmetre.
- **`void jumpOver()`**
  Aquesta funció és privada i indica que el jugador ha saltat per sobre del pou.
- **`boolean traverse(Player p)`**
  Si la cel·la ja està oberta, inicialitza el `traverseMessage` amb el missatge "Ja s'havia passat per aquesta cel·la" i retorna `true`. Si no:
  - si el jugador té el `PowerUp JUMPER_BOOTS`, salta el pou, obre la cel·la, inicialitza el missatge `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "S'ha pogut saltar per sobre i el jugador pot continuar" i retorna `true`;
    si el jugador no té el `PowerUp JUMPER_BOOTS`, només inicialitza el missatge `traverseMessage` concatenant el valor obtingut per la funció `toString()` de la mare amb el missatge "No s'ha pogut saltar per sobre i el jugador ha caigut i ha quedat atrapat!!!!!" i retorna `false`.

Exercici 2.
Creeu la classe **Player** dins del *package* **labyrinth.models**. Aquesta classe representarà un jugador que entra al laberint.

Atributs
De cada jugador se'n vol guardar les dades següents:

- El nom: `name`
- La posició del laberint on es troba: `row` i `col`
- La mida del laberint (nombre de files i columnes): `labyrinthRows, labyrinthCols`
- Els poders que té acumulats: powers[]
  Aquest array serà de 4 posicions i cada posició representarà el nombre d'unitats de cada poder (PowerUp):
  Posició 0: unitats del poder `WATER_BALLON`
  Posició 1: unitats del poder `KNIFE`
  Posició 2: unitats del poder `JUMPER_BOOTS`
  Posició 3: unitats del poder `FLUTE`

Mètodes

- **Constructor per defecte**
  Inicialitza un jugador amb el nom de "Visitant", a la posició (-1, -1), dins d'un laberint de mida 0x0 i amb 3 poders inicials.
- **Constructor parametritzat**
  Rep per paràmetre el nom del jugador, la posició on es troba (casella d'inici) i la mida del laberint. Inicialitza el jugador amb aquestes dades i, a més a més, amb 3 poders inicials.
- **Getters dels paràmetres `name`, `row` i `col`**
- **`boolean move(char movement)`**
  Si la posició actual del jugador és la (-1, -1) retornarà `false`.
  Si no, mourà el jugador una casella amunt, avall, a dreta o a esquerra segons la lletra rebuda per paràmetre (U, D, R i L respectivament).
  Cal tenir en compte que si el jugador està en algun dels marges del laberint no podrà fer tots els moviments (per exemple, si es troba a la part superior, no podrà pujar més) i la funció retornarà `false`. Així mateix, si la lletra del paràmetre és incorrecta, la funció no farà res i retornarà `false`.
- **`void setStartingCell(int row, int col)`**
  Si la posició actual del jugador és la (-1, -1) el situarà a la cel·la d'inici del laberint (`row`, `col`). Si no, no farà res.
- **`void teleport(int row, int col)`**
  Mourà al jugador a la posició indicada per paràmetre.
  En cas que els paràmetres `row` i `col` indiquin una posició fora del laberint, el jugador es mourà fins al marge més proper.
- **`void setLabyrinthSize(int labyrinthRows, int labyrinthCols)`**
  Si la mida actual del laberint és de 0x0, s'inicialitzarà a `labyrinthRows` x `labyrinthCols`. Si no, no es farà res.
- **`int getPowerUpQuantity(PowerUp power)`**
  Retornarà el nombre d'unitats del `PowerUp power` que té el jugador.
- **`boolean usePower(PowerUp power)`**
  Si el jugador té unitats del `PowerUp power`, n'utilitzarà 1 (en restarà una) i retornarà `true`. Si no, no farà res i retornarà `false`.
- **`void addPower(PowerUp power)`**
  Incrementarà en una unitat la quatitat del `PowerUp power` que tingui el jugador.
- **`void initializeInitPowers()`**
  Aquesta funció és privada i inicialitzarà els 3 poders inicials del jugador de manera aleatòria.

Exercici 3.
Creeu la classe **Labyrinth** dins del *package* **labyrinth.controller**. Aquesta classe representarà el laberint.

Atributs
De cada laberint se'n vol guardar les dades següents:

- Les cel·les, que serà una matriu de mida mínima 5x5 i de mida màxima 10x10: `labyrinth[][]`
- El jugacor: `player`
- El nombre de files i columnes del laberint: `nrows, ncols`
- La cel·la d'inici, que estarà en alguna casella de la columna 0: `iniCellRow` i `iniCellCol`
- Si el joc a acabat: `gameEnded`

- El missatge generat per l'última cel·la que ha visitat l'usuari: `lastCellMessage`

Mètodes

- **Constructor per defecte**
  Inicialitza un laberint amb els paràmetres `labyrinth` i `player` a `null`, de mida 0x0, amb el joc acabat (`gameEnded` a `true`) i amb el missatge de l'última cel·la buit.
- **boolean loadLabyrinth(String filename)**
  Llegirà el fitxer `filename` i carregarà les dades del laberint a memòria. Si tot funciona bé, retornarà `true`, si no, `false`.
- **`boolean createLabyrinth(String filename)`**
  Crearà un laberint aleatori i n'emmagatzemarà les dades al fitxer `filename`. Si tot funciona bé, retornarà true, si no, false.
- **`boolean startGame(String playerName)`**
  Si el laberint és `null`, retornarà false.
  Si no, posarà `gameEnded` a `false`, crearà un nou usuari amb les dades necessàries i retornarà `true`.
- **`CellType getPlayerCellType()`**
  Si `gameEnded` és `false`, retornarà `null`.
  Si no, retornarà el tipus de la cel·la on es troba, actualment, el jugador.
- **`CellType movePlayer(char movement)`**
  Si `gameEnded` és `false`, retornarà `null`.
  Si no, intentarà moure el jugador segons el moviment indicat. Si el jugador s'ha pogut moure a la següent cel·la, retornarà el tipus d'aquesta nova cel·la. En cas contrari, retornarà `null`.
- **`boolean traverseCell()`**
  Aplicarà l'efecte de la cel·la on es troba el jugador sobre aquest.
  En cas que la cel·la sigui de tipus `NORMAL_EXIT` o el jugador no tingui la capacitat per poder-la recórrer, `gameEnded` passarà a `true`.
  A més a més, inicialitzarà l'atribut `lastCellMessage` amb el `messageTraverse` de la cel·la que s'està tractant.
  La funció retornarà `true` si la cel·la s'ha pogut recórrer i `false` en cas contrari.
- **`String getLastCellMessage()`**
  Retorna el valor de l'atribut `lastCellMessage`.
- **`boolean isGameEnded()`**
  Retorna `true` si el joc s'ha acabat i `false` en cas contrari.
- **`String toString()`**
  Representa el laberint en format `String`.
  Detalls que cal tenir en compte:
  - La cel·la d'entrada i les cel·les obertes es representaran amb un espai en blanc
  - La cel·la de sortida es representarà amb el caràcter "~"
  - Les cel·les de teletransport obertes es representaran amb el caràcter "·"
  - Les cel·les tancades es representaran amb el caràcter "#"
  - La cel·la on es troba el jugador es representarà amb el caràcter "O"
  - Sota el dibuix del laberint s'hi afegirà la representació de l'estat del jugador.

Exercici 4.
Creeu el programa **LabyrinthMain** dins del *package* **labyrith.main** per gestionar el joc.
El primer que es mostrarà serà el menú següent:

```
~~~~ BENVINGUT AL JOC DEL LABERINT ~~~~

1. Carregar partida
2. Crear una nova partida
```

S'esculli l'opció que s'esculli, si és correcta, es demanarà al jugador que introdueixi el nom de la partida que vol carregar (o on vol guardar el nou laberint). **Aquests fitxers s'emmagatzemaran dins de la carpeta games/ (al mateix nivell que la carpeta src/ del projecte)**. Cal tenir en compte que caldrà informar dels errors produïts per la introducció d'opcions errònies.

Un cop establer-ta la partida, es demanarà el nom del jugador i s'iniciarà el joc.

Mentre el joc no acabi, s'aniran fent tirades, de tal manera que, per cada tirada s'hauran de processar els següents passos:

1. Mostrar el laberint amb l'estat actual
2. Demanar un nou moviment
3. Moure el jugador (si el jugador no es pot moure en aquella direcció, avisar per pantalla)
4. En cas que el jugador es pugui moure
   a) Intentar recórrer la cel·la i mostrar el missatge que genera. Si no es pot travessar, el joc acabarà amb GameOver.
   b) Si la cel·la es pot recórrer, però és de tipus TELEPORTATION, caldrà repetir els punts 3 i 4 fins que el jugador caigui a una cel·la que no sigui de tipus TELEPORTATION.
   c) Si la cel·la es pot recórrer i no es de tipus TELEPORTATION tornar al pas 1
5. Finalització del joc indicant si hi ha hagut GameOver o si el jugador ha pogut sortir del laberint.

VALORACIONS A TENIR EN COMPTE

1. **Un nom de Classe, variable/atribut o funció/mètode incorrecte (amb majúscules i minúscules mal posades) implicarà una nota de 0 de la pràctica!**
2. Sempre que pugueu, feu ús de constants.
3. Tabuleu bé el codi

DOCUMENTACIÓ ADJUNTA

1. Un fitxer de laberint de mostra
2. Els Enums **CellType** i **PowerUp**, que es guardaran al *package* **labyrinth.tools**.

INSTRUCCIONS D'ENTREGA

Per entregar aquests exercicis només cal que comprimiu en **ZIP** (**si m'ho envieu en RAR no ho podré obrir!**) la carpeta *src* del projecte i la pugeu al Moodle.

Atenció: *si feu l'exercici en parelles cal que adjunteu un fitxer TXT on hi poseu el nom dels dos membres del grup.*