**TP3 : semaines 3 et 4**

**TRAVAIL A FAIRE : Jeu du labyrinthe en mode console sur NetBeans**

On souhaite réaliser un jeu de type labyrinthe enregistré dans un fichier texte : voir fichier **labyrinthe.txt** sur campus à télécharger mais que vous pouvez modifier en respect de son format ci-dessous comme exemple :

**5 5 0 0 4 4** (taille du labyrinthe en X et Y, point de départ en X et Y, point d’arrivée en X et Y)

**\_X\_\_\_** (dessin du labyrinthe avec **X** un mur et **\_** une case à trou)

**\_XX\_\_**

**\_\_\_\_\_**

**\_ XX \_\_**

**\_\_X\_\_**

L’objectif est de se déplacer du point de départ au point d’arrivée en passant par les cases à trou **\_** et en évitant les murs **X**.

Ce jeu dispose des informations suivantes :

* Une classe *Labyrinthe* représente un labyrinthe composé d’une collection de cases. Ses dimensions en X (largeur) et en Y (hauteur) sont définies par les attributs *tailleX* et *tailleY*. Le point de départ est donné par les attributs *departX* et *departY*, l’arrivée par les attributs *arriveeX* et *arriveeY*, la position courante par les attributs *posX* et *posY*. Tous les attributs sont privés.
* Une interface (équivalent d’une bibliothèque .h en C avec seulement les prototypes des méthodes et sans attribut) *Case* représente une case du labyrinthe avec les prototypes des méthodes suivantes :

public interface Case {

public int getPositionX() ; // retourne la position en X de la case

public int getPositionY() ; // retourne la position en Y de la case

public boolean canMoveToCase() ; // indique s’il est possible ou non d’aller dans la case

}

* Une classe *CaseImplementee* implémente l’interface Case et toutes ses méthodes. Ses attributs protégés sont *posX* et *posY* pour la position en X et Y de la case, un booléen *vasy* indique s’il est possible ou non d’aller dans la case.
* Les classes *CaseMur* et *CaseTrou* héritent de la classe *CaseImplementee*.

**Exercice 1 : conception du diagramme de classes**

Pour la conception du diagramme de classes, appuyez-vous sur le « Support conception orientée objet » sur la page campus du cours : <http://campus.ece.fr/course/view.php?id=124>.

Donnez le diagramme de classes correspondant à la description du jeu ci-dessus sans oublier les niveaux de visibilité pour les attributs et les méthodes (**public** + **private** – ou **protected** #) et les multiplicités des relations inter-classes.

**Exercice 2 : implémentation des classes**

Des extraits utiles de la documentation Javadoc sont fournis en **annexe** : pour plus de détails, consulter la [documentation des API Java *(javadoc*)](http://docs.oracle.com/javase/6/docs/api/) sur la page campus du cours.

Commentez votre code : devant les classes, attributs, constructeurs et méthodes vos commentaires doivent respecter le format Javadoc /\*\* commentaires \*/ pour que ceux-ci apparaissent dans la javadoc générée

* 1. Créer le projet et 3 packages
  2. Dans le premier package, écrire l’interface *Case* avec les prototypes de ses méthodes (voir plus haut)

1. Dans le premier package, écrire la classe *CaseImplementee* qui définit ses attributs **protected**, implémente un constructeur avec en paramètres la position en X et en Y pour initialiser les attributs *posX* et *posY*, ainsi que toutes les méthodes de l’interface *Case*
   1. Dans le premier package, écrire les 2 classes *CaseMur* et *CaseTrou* qui implémentent un constructeur héritant de celui de la classe *CaseImplementee* et spécifie si on peut aller ou non dans la case avec le booléen *vasy*. Pour les objets de la classe *CaseMur*, impossible d’aller dans leur case : voir la méthode *canMoveToCase()* de la classe *Case* et donc une redéfinition de cette méthode est à envisager.
   2. Dans le second package, construire la classe *Labyrinthe* chargée de répertorier ces cases. L’une des 3 approches suivantes à choisir librement peut définir une collection de cases comme attribut :

* Une matrice de cases (*Case* [][]).
* Un *ArrayList* de cases.
* Un *ArrayList* d’*ArrayList* de cases.

La classe *Labyrinthe* devra implémenter les méthodes suivantes en respect de leurs prototypes et des commentaires, sans oublier de définir les attributs privés qui sont précisés plus haut, ni les getters pour y accéder :

/\*\* Lit un labyrinthe avec un fichier en paramètre (voir classe **File** en annexe) au format décrit plus haut

Initialise tous les attributs avec les valeurs lues dans le fichier puis instancie la collection de cases et chaque case

Déclenche l’exception *FileFormatException* si le fichier ne peut être lu ou si son format est incorrect \*/

public void initFromFile(File lab) throws FileFormatException ;

/\*\* Tente de bouger le curseur dans la case (x, y) en paramètres. Déclenche l’exception *ImpossibleMoveException* si la

case déborde du labyrinthe ou si on ne peut pas aller dans la case :

voir la méthode *canMoveToCase()* de la classe *Case* \*/

public void move (int x, int y) throws ImpossibleMoveException;

/\*\* Se déplace aléatoirement d’une seule case (direction en x et y aléatoire) de la position courante (*posX*, *posY*) sauf si

si ce déplacement sort du labyrinthe ou va dans un mur \*/

public void autoMove() ;

public int getCurrentPositionX() ; // Donne la position courante en *posX*

public int getCurrentPositionX() ; // Donne la position courante en *posY*

Les classes d’exception *FileFormatException* et *ImpossibleMoveException* n’existant pas, vous devez les implémenter.

* 1. Dans le troisième package, écrire une classe avec le *main* avec sa boucle de jeu :
* Introduire un menu avec les options suivantes : déplacement manuel (directions), déplacement automatique intelligent, déplacement automatique aléatoire et sortie du jeu.
* Saisir un nom de fichier, appeler les méthodes nécessaires pour lire le fichier et afficher le labyrinthe.
* Se déplacer du point de départ au point d’arrivée si c’est possible selon l’option du déplacement choisi dans le menu, en affichant au fur et à mesure la position courante.
* En cas d’exceptions *FileFormatException* et *ImpossibleMoveException* afficher des messages d’erreur.

Si nécessaire, implémenter d’autres méthodes dans cette classe comme le déplacement automatique intelligent :

* Par exemple, avec l’algorithme de ces sacrés bons vieux Dijkstra ou BFS (*Breadth First Search* avec une file d’attente) pour avoir le chemin le plus court (nostalgie des ex-ING2 en Théorie des graphes ☺)
* Avec un algorithme récursif si vous voulez traiter tous les chemins possibles (là je sens que vous adorez ☺ ).

**ANNEXE : extraits de la documentation Javadoc**

**java.io**

**Class File**

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**File**](http://docs.oracle.com/javase/6/docs/api/java/io/File.html#File(java.lang.String))([String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) pathname)  Creates a new File instance by converting the given pathname string into an abstract pathname. |  |

**Class IOException**

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**IOException**](http://docs.oracle.com/javase/6/docs/api/java/io/IOException.html#IOException())()  Constructs an IOException with null as its error detail message. |  |
| [**IOException**](http://docs.oracle.com/javase/6/docs/api/java/io/IOException.html#IOException(java.lang.String))([String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) message)  Constructs an IOException with the specified detail message. |  |

## java.util

## Class Scanner

|  |  |  |
| --- | --- | --- |
| **Constructor Summary** | | |
| [**Scanner**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#Scanner(java.io.File))([File](http://docs.oracle.com/javase/6/docs/api/java/io/File.html) source)  Constructs a new Scanner that produces values scanned from the specified file. | |  |
| [**Scanner**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#Scanner(java.lang.String))([String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) source)  Constructs a new Scanner that produces values scanned from the specified string. | |  |
| **Method Summary** | | |
| void | [**close**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#close())()  Closes this scanner. | |
| boolean | [**hasNext**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#hasNext())()  Returns true if this scanner has another token in its input. | |
| boolean | [**hasNextByte**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#hasNextByte())()  Returns true if the next token in this scanner's input can be interpreted as a byte value in the default radix using the [nextByte()](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#nextByte()) method. | |
| boolean | [**hasNextInt**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#hasNextInt())()  Returns true if the next token in this scanner's input can be interpreted as an int value in the default radix using the [nextInt()](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#nextInt()) method. | |
| boolean | [**hasNextLine**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#hasNextLine())()  Returns true if there is another line in the input of this scanner. | |
| [IOException](http://docs.oracle.com/javase/6/docs/api/java/io/IOException.html) | [**ioException**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#ioException())()  Returns the IOException last thrown by this Scanner's underlying Readable. | |
| [String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) | [**next**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#next())()  Finds and returns the next complete token from this scanner. | |
| byte | [**nextByte**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#nextByte())()  Scans the next token of the input as a byte. | |
| int | [**nextInt**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#nextInt())()  Scans the next token of the input as an int. | |
| [String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) | [**nextLine**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#nextLine())()  Advances this scanner past the current line and returns the input that was skipped. | |

## java.util

**Class ArrayList<E>**

|  |  |  |
| --- | --- | --- |
| **Constructor Summary** | | |
| [**ArrayList**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#ArrayList())()  Constructs an empty list with an initial capacity of ten. | |  |
| [**ArrayList**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#ArrayList(java.util.Collection))([Collection](http://docs.oracle.com/javase/6/docs/api/java/util/Collection.html)<? extends [E](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html)> c)  Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator. | |  |
| **Method Summary** | | |
| boolean | [**add**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#add(E))([E](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html) e)  Appends the specified element to the end of this list. | |
| void | [**add**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#add(int, E))(int index, [E](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html) element)  Inserts the specified element at the specified position in this list. | |
| boolean | [**addAll**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#addAll(java.util.Collection))([Collection](http://docs.oracle.com/javase/6/docs/api/java/util/Collection.html)<? extends [E](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html)> c)  Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator. | |
| boolean | [**addAll**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#addAll(int, java.util.Collection))(int index, [Collection](http://docs.oracle.com/javase/6/docs/api/java/util/Collection.html)<? extends [E](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html)> c)  Inserts all of the elements in the specified collection into this list, starting at the specified position. | |
| void | [**clear**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#clear())()  Removes all of the elements from this list. | |
| boolean | [**contains**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#contains(java.lang.Object))([Object](http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) o)  Returns true if this list contains the specified element. | |
| [E](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html) | [**get**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#get(int))(int index)  Returns the element at the specified position in this list. | |
| int | [**indexOf**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#indexOf(java.lang.Object))([Object](http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) o)  Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. | |
| boolean | [**isEmpty**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#isEmpty())()  Returns true if this list contains no elements. | |
| int | [**lastIndexOf**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#lastIndexOf(java.lang.Object))([Object](http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) o)  Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element. | |
| [E](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html) | [**remove**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#remove(int))(int index)  Removes the element at the specified position in this list. | |
| boolean | [**remove**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#remove(java.lang.Object))([Object](http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) o)  Removes the first occurrence of the specified element from this list, if it is present. | |
| [E](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html) | [**set**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#set(int, E))(int index, [E](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html) element)  Replaces the element at the specified position in this list with the specified element. | |
| int | [**size**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#size())()  Returns the number of elements in this list. | |
| [Object](http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html)[] | [**toArray**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#toArray())()  Returns an array containing all of the elements in this list in proper sequence (from first to last element). | |
| |  | | --- | | <T> T[] | | [**toArray**](http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html#toArray(T[]))(T[] a)  Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array. | |

## java.util

**Class Random**

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Random**](http://docs.oracle.com/javase/6/docs/api/java/util/Random.html#Random())()  Creates a new random number generator. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| protected int | [**next**](http://docs.oracle.com/javase/6/docs/api/java/util/Random.html#next(int))(int bits)  Generates the next pseudorandom number. |
| int | [**nextInt**](http://docs.oracle.com/javase/6/docs/api/java/util/Random.html#nextInt())()  Returns the next pseudorandom, uniformly distributed int value from this random number generator's sequence. |
| int | [**nextInt**](http://docs.oracle.com/javase/6/docs/api/java/util/Random.html#nextInt(int))(int n)  Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence. |