

Guillaume BELDILMI  
avec Ryan AIT CHEIKH  
L1-IE2-31

# Projet Info2A

Rapport

## Table des matières

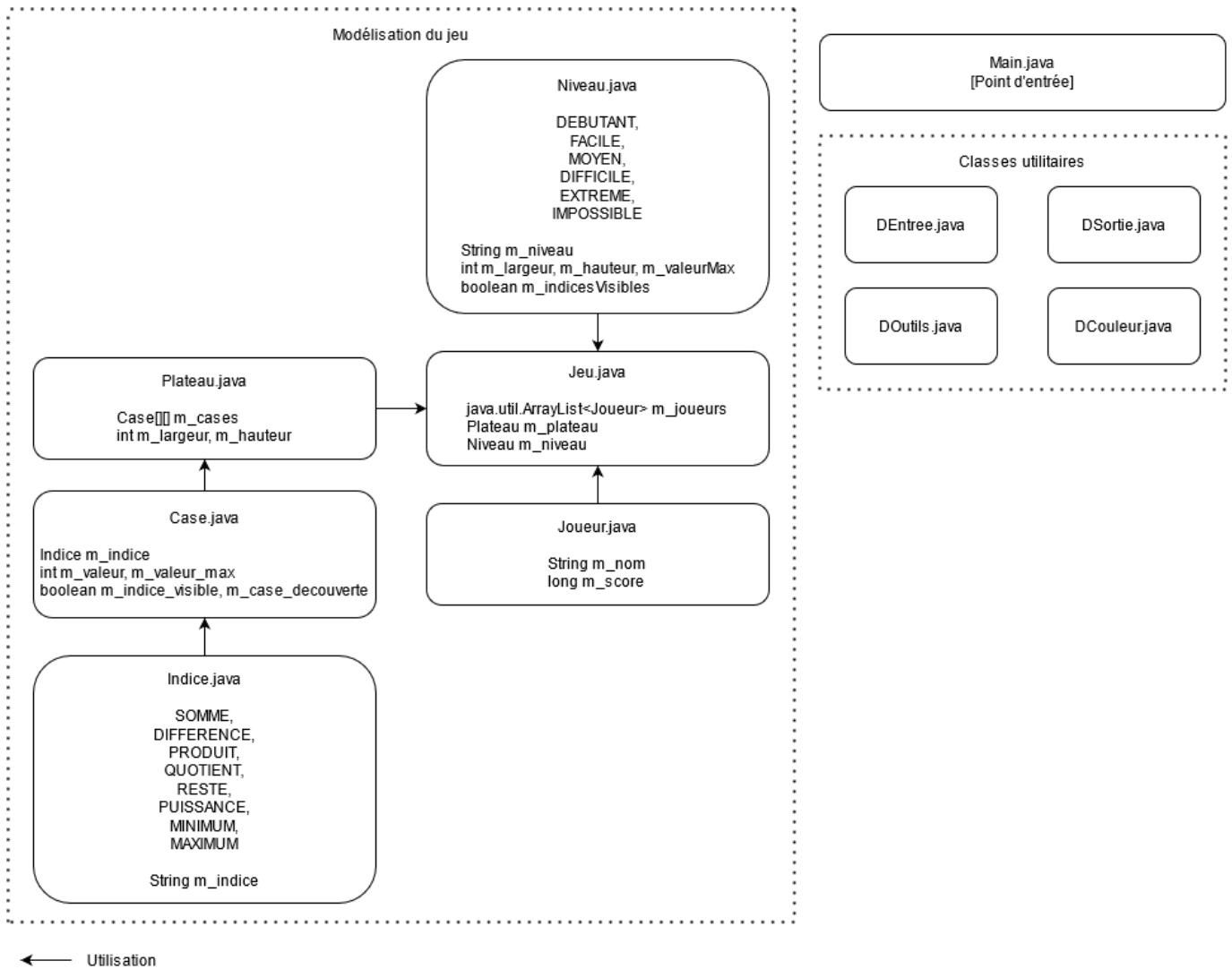
Table des matières .....	2
Diagramme .....	6
Descriptions.....	7
Case.java.....	7
Indice m_indice .....	7
int m_valeur .....	7
int m_valeur_max.....	7
boolean m_indice_visible.....	7
boolean m_case_decouverte .....	7
Case(int max, boolean indiceVisible).....	7
void genCase() .....	7
Indice getIndice().....	7
int getValeur().....	7
int getValeurMax() .....	7
boolean isCaseDecouverte().....	7
boolean isIndiceVisible().....	7
boolean isValeur(int v) .....	7
void setCaseDecouverte(boolean d) .....	7
void setIndice(Indice i) .....	8
void setIndiceVisible(boolean iv).....	8
void setValeur(int v) .....	8
void setValeurMax(int vm) .....	8
String toString() .....	8
DCouleur.java .....	9
String m_couleur .....	9
DCouleur(String couleur).....	9
String getCouleur() .....	9
void setCouleur(String c) .....	9
String toString() .....	9
DEntree.java .....	10
boolean getConfirmation().....	10
int getInt().....	10
int getInt(int min) .....	10
int getInt(int min, int max) .....	10
String getString().....	10

String lire() .....	10
void pause() .....	10
DUtils.java .....	11
int getNbChiffre(long i) .....	11
DSortie.java .....	12
void affiche(Object x) .....	12
void afficheErreur(Object e) .....	12
void afficheLigne(Object x) .....	12
String blanc(Object x) .....	12
String bleu(Object x) .....	12
String cyan(Object x) .....	12
String jaune(Object x) .....	12
String magenta(Object x) .....	12
String noir(Object x) .....	12
String rouge(Object x) .....	12
String vert(Object x) .....	12
Indice.java .....	13
String m_indice .....	13
Indice(String indice) .....	13
String getIndice() .....	13
double getResultat(Case c1, Case c2) .....	13
void setIndice(String i) .....	13
String toString() .....	13
Jeu.java .....	14
java.util.ArrayList<Joueur> m_joueurs .....	14
Plateau m_plateau .....	14
Niveau m_niveau .....	14
Jeu(java.util.ArrayList<Joueur> joueurs, Niveau niveau) .....	14
void afficheCaracteristiquesJoueurs() .....	14
void afficheCaracteristiquesPartie() .....	14
Joueur getJoueur(int i) .....	14
java.util.ArrayList<Joueur> getJoueurs() .....	14
int getNbJoueurs() .....	14
Niveau getNiveau() .....	14
Plateau getPlateau() .....	14
void joue() .....	14

void setJoueurs(java.util.ArrayList<Joueur> j) .....	14
void setNiveau(Niveau n) .....	14
void setPlateau(Plateau p) .....	14
String toString() .....	15
Joueur.java .....	16
String m_nom .....	16
long m_score .....	16
Joueur(String nom) .....	16
String getNom() .....	16
long getScore() .....	16
void joue(Plateau p, Niveau n) .....	16
void reussi(long s) .....	16
void setNom(String n) .....	16
void setScore(long s) .....	16
String toString() .....	16
Main.java .....	17
void main(String[] args) .....	17
Niveau.java .....	18
String m_niveau .....	18
int m_largeur .....	18
int m_hauteur .....	18
int m_valeurMax .....	18
boolean m_indicesVisibles .....	18
Niveau(String niveau, int largeur, int hauteur, int valeurMax, boolean indicesVisibles) .....	18
int getHauteur() .....	18
int getLargeur() .....	18
String getNiveau() .....	18
int getValeurMax() .....	18
boolean isIndicesVisibles() .....	18
void setHauteur(int h) .....	18
void setIndicesVisibles(boolean iv) .....	18
void setLargeur(int l) .....	18
void setNiveau(String n) .....	19
void setValeurMax(int vm) .....	19
String toString() .....	19
Plateau.java .....	20

Case[][] m_cases.....	20
int m_largeur .....	20
int m_hauteur.....	20
Plateau(int largeur, int hauteur, int valeurMax, boolean indicesVisibles).....	20
void affiche().....	20
Case getCas(int x, int y) .....	20
Case[][] getCases().....	20
int getHauteur().....	20
int getLargeur().....	20
void setCase(int x, int y, Case c) .....	20
void setCases(Case[][] c).....	20
void setHauteur(int h) .....	20
void setLargeur(int l) .....	20
String toString() .....	20
boolean toutesLesCasesSontDecouvertes() .....	20

## Diagramme



## Descriptions

### Case.java

Cette classe sert à représenter une case du plateau.

#### Indice m\_indice

Cet attribut sert à représenter l'indice que possède la case.

#### int m\_valeur

Cet attribut sert à représenter la valeur de la case.

#### int m\_valeur\_max

Cet attribut sert à représenter la valeur maximale que peut prendre la case.

#### boolean m\_indice\_visible

Cet attribut sert à savoir si l'indice de la case doit être affiché ou non.

#### boolean m\_case\_decouverte

Cet attribut sert à savoir si la case a été découverte ou non.

#### Case(int max, boolean indiceVisible)

Il s'agit du constructeur de la classe. Pour créer un objet Case, il suffit de lui passer en paramètre la valeur maximale que peut prendre la case ainsi qu'un booléen ayant la valeur 'true' si l'indice de la case devra être affiché tant que la case n'est pas découverte.

#### void genCase()

Cette méthode est appelée par le constructeur et sert à attribuer un indice et une valeur aléatoires à la case.

#### Indice getIndex()

Cet accesseur en lecture sert à renvoyer l'indice de la case, stocké dans l'attribut 'm\_indice'.

#### int getValeur()

Cet accesseur en lecture sert à renvoyer la valeur de la case, stockée dans l'attribut 'm\_valeur'.

#### int getValeurMax()

Cet accesseur en lecture sert à renvoyer la valeur maximale de la case, stockée dans l'attribut 'm\_valeur\_max'.

#### boolean isCaseDecouverte()

Cet accesseur en lecture sert à renvoyer l'état du booléen stocké dans l'attribut 'm\_case\_decouverte'.

#### boolean isIndiceVisible()

Cet accesseur en lecture sert à renvoyer l'état du booléen stocké dans l'attribut 'm\_indice\_visible'.

#### boolean isValeur(int v)

Cette méthode sert à vérifier si la valeur passée en paramètre est égale à la valeur de la case, et renvoie 'true' si c'est bien le cas.

#### void setCaseDecouverte(boolean d)

Cet accesseur en écriture sert à spécifier l'état que doit prendre l'attribut 'm\_case\_decouverte'.

`void setIndice(Indice i)`

Cet accesseur en écriture sert à attribuer un indice à la case, il sera stocké dans l'attribut 'm\_indice'.

`void setIndiceVisible(boolean iv)`

Cet accesseur en écriture sert à spécifier l'état que doit prendre l'attribut 'm\_indice\_visible'.

`void setValeur(int v)`

Cet accesseur en écriture sert à attribuer une valeur à la case, elle sera stockée dans l'attribut 'm\_valeur'.

`void setValeurMax(int vm)`

Cet accesseur en écriture sert à attribuer une valeur maximale à la case, elle sera stockée dans l'attribut 'm\_valeur\_max'.

`String toString()`

Cette méthode retourne une chaîne de caractères pour représenter la case sous la forme '<valeur><indice>'. Si certains attributs de la case ne doivent pas être attribués, ils seront remplacés par des caractères '#'.



## DCouleur.java

Cette énumération fait partie des classes utilitaires du projet, elle ne sert qu'à contenir les séquences d'échappement ANSI permettant de changer la couleur d'écriture du texte dans le terminal.

'NOIR', 'ROUGE', 'VERT', 'JAUNE', 'BLEU', 'MAGENTA', 'CYAN' et 'BLANC' permettent respectivement de changer la couleur du texte en noir, rouge, vert, jaune, bleu, magenta, cyan et blanc. Tandis que 'REINITIALISATION' permet de réinitialiser la couleur du texte dans le terminal.

## String m\_couleur

Il s'agit de l'attribut qui permet de stocker la séquence d'échappement de chaque élément de l'énumération.

## DCouleur(String couleur)

Le constructeur de l'énumération prenant en paramètre la séquence d'échappement de la couleur choisie.

## String getCouleur()

Cet accesseur en lecture sert à renvoyer le code d'échappement, stocké dans l'attribut 'm\_couleur'.

## void setCouleur(String c)

Cet accesseur en écriture sert à attribuer un code d'échappement, stocké dans l'attribut 'm\_couleur'.

## String toString()

Cette méthode permet de renvoyer la une chaine de caractères contenant la séquence d'échappement.

## DEntree.java

Cette classe fait partie des classes utilitaires du projet, elle sert à recueillir ce que l'utilisateur saisit dans la console.

### boolean getConfirmation()

Cette méthode sert à obtenir une réponse négative ou positive de l'utilisateur en entrant 'o' pour une réponse positive et 'n' pour une réponse négative en étant insensible à la casse, une entrée vide ou incorrecte considèrera la réponse comme positive par défaut.

### int getInt()

Cette méthode permet de demander à l'utilisateur d'entrer quelque chose et de renvoyer sa saisie sous la forme d'un entier, et redemande une entrée à chaque fois que l'utilisateur rentre une saisie incorrecte.

### int getInt(int min)

Cette méthode permet de demander à l'utilisateur d'entrer quelque chose et de renvoyer sa saisie sous la forme d'un entier, et redemande une entrée à chaque fois que l'utilisateur rentre une saisie incorrecte ou inférieur à la valeur 'min' passée en paramètre.

### int getInt(int min, int max)

Cette méthode permet de demander à l'utilisateur d'entrer quelque chose et de renvoyer sa saisie sous la forme d'un entier, et redemande une entrée à chaque fois que l'utilisateur rentre une saisie incorrecte, inférieur à la valeur 'min' ou supérieure à 'max' passées en paramètre.

### String getString()

Cette méthode permet de demander à l'utilisateur d'entrer quelque chose et de renvoyer sa saisie sous forme d'une chaîne de caractères.

### String lire()

Cette méthode sert à récupérer ce que saisit l'utilisateur et le renvoie sous forme de chaîne de caractères. Elle n'est accessible que par les autres méthodes de la classe 'DEntree'.

### void pause()

Cette méthode sert à faire une pause dans le programme, en affichant un message à l'utilisateur lui demandant d'appuyer sur la touche entrée pour continuer.

## DUtils.java

Cette classe fait partie des classes utilitaires du projet, elle sert à recueillir diverses méthodes n'ayant pas directement un rapport avec le projet mais qui peuvent être utilisés de diverses façons.

`int getNbChiffre(long i)`

Cette méthode retourne le nombre de chiffre contenus dans un entier.

## DSortie.java

Cette classe fait partie des classes utilitaires du projet, elle sert à gérer l’affichage dans la console.

### void affiche(Object x)

Cette méthode est juste un enrobeur pour la méthode ‘System.out.print()’ mais est plus court à écrire.

### void afficheErreur(Object e)

Cette méthode permet d’afficher une erreur dans la console avec une mise en forme particulière.

### void afficheLigne(Object x)

Comme la méthode précédente, cette méthode est juste un enrobeur pour la méthode ‘System.out.println()’ mais est plus court à écrire.

### String blanc(Object x)

Cette méthode ajoute les séquences d’échappement nécessaire pour que l’argument passé en paramètre apparaisse en blanc une fois affiché dans la console.

### String bleu(Object x)

Cette méthode ajoute les séquences d’échappement nécessaire pour que l’argument passé en paramètre apparaisse en bleu une fois affiché dans la console.

### String cyan(Object x)

Cette méthode ajoute les séquences d’échappement nécessaire pour que l’argument passé en paramètre apparaisse en cyan une fois affiché dans la console.

### String jaune(Object x)

Cette méthode ajoute les séquences d’échappement nécessaire pour que l’argument passé en paramètre apparaisse en jaune une fois affiché dans la console.

### String magenta(Object x)

Cette méthode ajoute les séquences d’échappement nécessaire pour que l’argument passé en paramètre apparaisse en magenta une fois affiché dans la console.

### String noir(Object x)

Cette méthode ajoute les séquences d’échappement nécessaire pour que l’argument passé en paramètre apparaisse en noir une fois affiché dans la console.

### String rouge(Object x)

Cette méthode ajoute les séquences d’échappement nécessaire pour que l’argument passé en paramètre apparaisse en rouge une fois affiché dans la console.

### String vert(Object x)

Cette méthode ajoute les séquences d’échappement nécessaire pour que l’argument passé en paramètre apparaisse en vert une fois affiché dans la console.

## Indice.java

Cette énumération sert à représenter les différents types d'indices que peut prendre une case.

Il n'existe que huit types d'indices pour l'instant : 'SOMME', 'DIFFERENCE', 'PRODUIT', 'QUOTIENT', 'RESTE', 'PUISSANCE', 'MINIMUM' et 'MAXIMUM'.

### String m\_indice

Cet attribut sert à stocker le signe de l'indice de la case sous la forme d'une chaîne de caractères.

### Indice(String indice)

Il s'agit du constructeur de l'énumération, il prend la représentation du type de l'indice sous la forme d'une chaîne de caractères en paramètre.

### String getIndice()

Cet accesseur en lecture renvoie la chaîne de caractères stockée dans l'attribut 'm\_indice'.

### double getResultat(Case c1, Case c2)

Cette méthode permet d'obtenir la valeur de l'indice entre deux cases selon le type de l'indice de la première case.

### void setIndice(String i)

Cet accesseur en écriture sert à attribuer un signe à l'indice, et le stock dans l'attribut 'm\_indice'.

### String toString()

Cette méthode renvoie l'indice sous la forme d'une chaîne de caractères.

## Jeu.java

Cette classe représente une partie et gère son bon déroulement.

`java.util.ArrayList<Joueur> m_joueurs`

Cet attribut sert à gérer l'ensemble des joueurs peu importe le nombre de ces derniers.

`Plateau m_plateau`

Cet attribut permet de stocker le plateau du jeu actuel et son avancement.

`Niveau m_niveau`

Cet attribut représente le niveau de difficulté choisis pour la partie en cours.

`Jeu(java.util.ArrayList<Joueur> joueurs, Niveau niveau)`

Ce constructeur permet d'initialiser une nouvelle partie en spécifiant l'ensemble de joueurs participants et le niveau de difficulté désiré.

`void afficheCaracteristiquesJoueurs()`

Cette méthode sert à afficher les caractéristiques des différents joueurs.

`void afficheCaracteristiquesPartie()`

Cette méthode sert à afficher les caractéristiques de la partie en cours.

`Joueur getJoueur(int i)`

Cette méthode permet d'obtenir la référence d'un joueur en spécifiant son rang dans la partie en cours.

`java.util.ArrayList<Joueur> getJoueurs()`

Cet accesseur en lecture permet d'obtenir l'ensemble des joueurs de la partie en cours stocké dans l'attribut 'm\_joueurs'.

`int getNbJoueurs()`

Cette méthode permet d'obtenir le nombre de joueurs de la partie en cours.

`Niveau getNiveau()`

Cet accesseur en lecture permet d'obtenir le niveau de la partie en cours.

`Plateau getPlateau()`

Cet accesseur en lecture permet d'obtenir la référence du plateau de la partie en cours.

`void joue()`

Cette méthode sert à simuler le déroulement de la partie.

`void setJoueurs(java.util.ArrayList<Joueur> j)`

Cet accesseur en écriture permet d'attribuer un ensemble de joueurs à la partie en cours en le stockant dans l'attribut 'm\_joueurs'.

`void setNiveau(Niveau n)`

Cet accesseur en écriture permet d'attribuer un niveau à la partie en cours en le stockant dans l'attribut 'm\_niveau'.

`void setPlateau(Plateau p)`

Cet accesseur en écriture permet d'attribuer un plateau à la partie en cours en le stockant dans l'attribut 'm\_plateau'.

String toString()

Cette méthode renvoie les caractéristiques de la partie sous la forme d'une chaîne de caractères.

## Joueur.java

Cette classe sert à représenter un joueur.

String m\_nom

Cet attribut sert à contenir le nom du joueur.

long m\_score

Cet attribut sert à contenir le score actuel du joueur.

Joueur(String nom)

Ce constructeur permet de créer un nouveau joueur en spécifiant son nom.

String getNom()

Cet accesseur en lecture permet d'obtenir le nom du joueur en retournant la référence à la chaîne de caractères contenue dans 'm\_nom'.

long getScore()

Cet accesseur en lecture permet d'obtenir le score du joueur en renvoyant la valeur stockée dans l'attribut 'm\_score'.

void joue(Plateau p, Niveau n)

Cette méthode sert à gérer le déroulement du tour d'un joueur et permet d'alléger la classe Jeu.

void reussi(long s)

Cette méthode sert à ajouter le nombre passé en paramètre au score du joueur lorsque ce dernier aura correctement deviné le contenu de deux cases.

void setNom(String n)

Cet accesseur en écriture permet de changer le nom du joueur.

void setScore(long s)

Cet accesseur en écriture permet de changer la valeur du score du joueur par la valeur passée en paramètre en la stockant dans l'attribut 'm\_score'.

String toString()

Cette méthode permet de retourner le nom et le score du joueur sous forme d'une chaîne de caractères.



## Main.java

Cette classe sert uniquement à contenir le point d'entrée du programme afin de le séparer des autres classes.

```
void main(String[] args)
```

Il s'agit du point d'entrée du programme. Cette méthode demande toutes les informations nécessaires à la création d'une nouvelle partie, lance cette dernière et recommence jusqu'à ce que l'utilisateur refuse de faire une nouvelle partie.

## Niveau.java

Cette énumération sert à représenter un niveau de difficulté de jeu. Les différents niveaux, par ordre croissant de difficulté, sont les suivants : 'DEBUTANT', 'FACILE', 'MOYEN', 'DIFFICILE', 'EXTREME' et 'IMPOSSIBLE'.

String m\_niveau

Cet attribut sert à contenir le nom du niveau de difficulté.

int m\_largeur

Cet attribut sert à contenir le nombre de case de la largeur du plateau.

int m\_hauteur

Cet attribut sert à contenir le nombre de case de la hauteur du plateau.

int m\_valeurMax

Cet attribut sert à contenir la valeur maximale que peut prendre une case du plateau.

boolean m\_indicesVisibles

Cet attribut sert à représenter si les indices des cases doivent être visibles ou non depuis le début de la partie ou non.

Niveau(String niveau, int largeur, int hauteur, int valeurMax, boolean indicesVisibles)

Il s'agit du constructeur de l'énumération demandant en paramètre toutes les valeurs des différents attributs.

int getHauteur()

Cet accesseur en lecture renvoie la valeur contenue dans l'attribut 'm\_hauteur'.

int getLargeur()

Cet accesseur en lecture renvoie la valeur contenue dans l'attribut 'm\_largeur'.

String getNiveau()

Cet accesseur en lecture renvoie la référence à la chaîne de caractères correspondant au nom du niveau de difficulté contenue dans l'attribut 'm\_niveau'.

int getValeurMax()

Cet accesseur en lecture renvoie la valeur contenue dans l'attribut 'm\_valeurMax'.

boolean isIndicesVisibles()

Cet accesseur en lecture renvoie la valeur du booléen contenu dans l'attribut 'm\_indicesVisibles'.

void setHauteur(int h)

Cet accesseur en écriture permet d'attribuer la valeur passée en paramètre à l'attribut 'm\_hauteur'.

void setIndicesVisibles(boolean iv)

Cet accesseur en écriture permet d'attribuer la valeur du booléen passé en paramètre à l'attribut 'm\_indicesVisibles'.

void setLargeur(int l)

Cet accesseur en écriture permet d'attribuer la valeur passée en paramètre à l'attribut 'm\_largeur'.

`void setNiveau(String n)`

Cet accesseur en écriture permet de changer le nom du niveau de difficulté avec la chaîne de caractères passée en paramètre.

`void setValeurMax(int vm)`

Cet accesseur en écriture permet d'attribuer la valeur passée en paramètre à l'attribut 'm\_valeurMax'.

`String toString()`

Cette méthode permet de retourner le nom du niveau de difficulté sous forme d'une chaîne de caractères.

## Plateau.java

Cette classe permet de représenter un plateau de jeu et de le manipuler.

`Case[][] m_cases`

Cet attribut sert à contenir l'ensemble des cases du plateau dans un tableau bidimensionnel.

`int m_largeur`

Cet attribut sert à contenir le nombre de cases que contient le plateau en largeur.

`int m_hauteur`

Cet attribut sert à contenir le nombre de cases que contient le plateau en hauteur.

`Plateau(int largeur, int hauteur, int valeurMax, boolean indicesVisibles)`

Ce constructeur permet de créer un nouveau plateau ayant la largeur et la hauteur spécifiées en paramètre. A noter que les deux autres paramètres servent à préciser l'état initial des cases du plateau, respectivement la valeur maximale que peut prendre une case et un booléen valant 'true' si les indices des cases doivent être visibles.

`void affiche()`

Cette méthode permet d'afficher le plateau.

`Case getCase(int x, int y)`

Cet accesseur en lecture permet d'obtenir la référence à une des cases du plateau en précisant ses coordonnées.

`Case[][] getCases()`

Cet accesseur en lecture permet d'obtenir la référence à l'ensemble des cases du plateau.

`int getHauteur()`

Cet accesseur en lecture permet d'obtenir la valeur de la hauteur du plateau.

`int getLargeur()`

Cet accesseur en lecture permet d'obtenir la valeur de la largeur du plateau.

`void setCase(int x, int y, Case c)`

Cette méthode sert à attribuer une case à l'attribut 'm\_cases' en précisant ses coordonnées.

`void setCases(Case[][] c)`

Cet accesseur en écriture sert à attribuer un ensemble de cases à l'attribut 'm\_cases'.

`void setHauteur(int h)`

Cet accesseur en écriture permet de modifier la valeur de l'attribut 'm\_hauteur'.

`void setLargeur(int l)`

Cet accesseur en écriture permet de modifier la valeur de l'attribut 'm\_largeur'.

`String toString()`

Cette méthode permet de renvoyer le contenu du plateau sous la forme d'une chaîne de caractères.

`boolean toutesLesCasesSontDecouvertes()`

Cette méthode renvoie la valeur 'false' tant que toutes les valeurs des cases n'ont pas toutes été trouvées.