Initiation au développement - SDA : TP7b

DUT/INFO/R1_01
version 2021-2022 (PN BUT 2021)

Table des matières

[1. Utilisation du Type Abstrait MatriceEntier](#)
  [1.1. Type abstrait JeuDeLaVie](#)
[2. Exercice 1 : Implémentation de l'Enregistrement JeuDeLaVie](#)
[3. Exercice 2 : Implémentation des opérations du type JeuDeLaVie](#)
[4. Exercice 3 : Ecriture du programme principal](#)
[5. Exercice 4 : Opération grilleConnue() pour le type JeuDeLaVie](#)
[6. Exercice 5 : Opération toHTML() pour le type JeuDeLaVie](#)
[7. Avant de partir](#)

Préambule

SI votre Type Abstrait **MatriceEntier** NE permet PAS de faire passer toutes les assertions de [MatriceEntierTest.java](#)

ALORS utiliser **ProgrammeMatriceEntier.java** fournis dans *Matériel TP8*.

1. Utilisation du Type Abstrait **MatriceEntier**

Objectif

Utiliser le type abstrait **MatriceEntier** pour implémenter une application *Jeu de la vie*.

Définition

Une application *Jeu de la vie* est un automate cellulaire où des cellules naissent et meurent sur une grille selon les règles suivantes inventées en 1970 par John Horton Conway.

1. Une cellule morte possédant exactement trois voisines vivantes devient vivante.
2. Une cellule vivante possédant deux ou trois voisines vivantes reste vivante, sinon elle meurt.

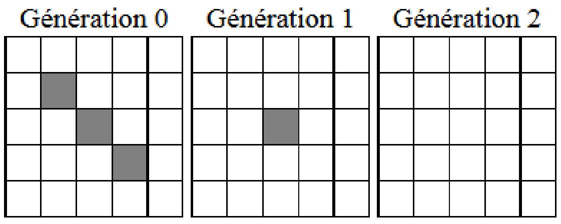

Figure 1. Exemple

1.1. Type abstrait **JeuDeLaVie**

Champs du type **JeuDeLaVie**

```
  int nbL ;
  int nbC ;
  MatriceEntier grille ;
  MatriceEntier grilleSuivante ;
  int generation = 0 ;
```

`grille` et **`grilleSuivante`** sont de taille **`(nbL+2)*(nbC+2)`**

Opérations du type **JeuDeLaVie**

```
  JeuDeLaVie() : Entier x Entier -> JeuDeLaVie
  estVivante() : JeuDeLaVie x Entier x Entier ->
Booleen
  getNbVoisinsVivants() : JeuDeLaVie x Entier x
Entier -> Entier
  seraVivante() : JeuDeLaVie x Entier x Entier ->
Booleen
  generationSuivante() : JeuDeLaVie -> JeuDeLaVie
```

Préconditions du type **JeuDeLaVie**

```
 JeuDeLaVie(l,c) valide SI ET SEULEMENT SI ( l > 0 )
ET ( c > 0 )
 estVivante(jeu,i,j) valide SI ET SEULEMENT SI (
(jeu.nbL + 2) > i >= 0 ) ET ( (jeu.nbC + 2) > j >= 0
)
 getNbVoisinsVivants(jeu,i,j) valide SI ET SEULEMENT
SI ( jeu.nbL >= i > 0 ) ET ( jeu.nbC >= j > 0 )
 seraVivante(jeu,i,j) valide SI ET SEULEMENT SI (
jeu.nbL >= i > 0 ) ET ( jeu.nbC >= j > 0 )
```

Les bords de la grille ne sont pas candidats pour les fonctions **`getNbVoisinsVivants`** et **`seraVivante`**.

Axiomes du type JeuDeLaVie

```
 estVivante(JeuDeLaVie(l,c),i,j) = FAUX qqs i et j
valides
 getNbVoisinsVivants(JeuDeLaVie(l,c),i,j) = 0 qqs i
et j valides
 seraVivante(JeuDeLaVie(l,c),i,j) = VRAI SI ET
SEULEMENT SI
        getNbVoisinsVivants(JeuDeLaVie(l,c),i,j) = 3
        OU
        ( estVivante(JeuDeLaVie(l,c),i,j) = VRAI ET
getNbVoisinsVivants(JeuDeLaVie(l,c),i,j) = 2)
```

2. Exercice 1 : Implémentation de

l’Enregistrement JeuDeLaVie

1. Ecrire une classe `JeuDeLaVie` qui permette de créer des valeurs du type **JeuDeLaVie** contenant les champs `nbL`, `nbC`, `grille`, `grilleSuivante` et `generation`.
2. Compléter cette classe avec un constructeur `JeuDeLaVie(int pfNbLignes, int pfNbColonnes)` qui lève une exception si `pfNbLignes` ou `pfNbColonnes` sont négatifs ou nuls, sinon initialise `nbL`, `nbC` et crée une `grille` de taille `(nbL+2)*(nbC+2)`.

3. Exercice 2 : Implémentation des opérations du type JeuDeLaVie

1. Implémenter dans une classe `ProgrammeJeuDeLaVie` les opérations et préconditions du type Abstrait **JeuDeLaVie**.

4. Exercice 3 : Ecriture du programme principal

1. Ecrire dans la classe `ProgrammeJeuDeLaVie` un programme principal `main()` qui :
    a. crée un JeuDeLaVie de taille 3 x 3
    b. initialise la génération 0 ainsi :

```
0 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 0
```

    c. puis affiche l'état des 2 générations suivantes.

Le résultat attendu est celui-ci :

```
>java -cp . ProgrammeJeuDeLaVie
Generation 0
0 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 0

Generation 1
0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0

Generation 2
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

>Exit code: 0
```

5. Exercice 4 : Opération `grilleConnue()` pour le type JeuDeLaVie

On souhaite disposer d’une opération `grilleConnue()` qui retourne VRAI si la grille de la génération courante a déja été rencontrée au cours d’une des générations précédentes.

Indications

1. Comment sauver l’historique des générations ?

    Dans un tableau de chaînes de caractères où chaque chaîne représente la grille d’une génération.

    > le tableau historique sera donc un champ supplémentaire de nom `histoire` dans l’Enregistrement `JeuDeLaVie` et de taille fixée à 10 éléments.

2. Comment transformer une grille en chaine ?

    `toString(jeu.grille)` le fait très bien !

3. Comment comparer 2 générations ?

    En comparant les chaînes de l’historique avec la grille courante transformée en chaîne.

1. Ajouter l’opération `grilleConnue` à la classe `ProgrammeJeuDeLaVie`.
2. Dans le programme `main`, coder son utilisation dans une boucle où le calcul de la génération suivante s’arrête si la grille obtenue a déja été rencontrée.

    Vous initialiserez la grille avec la configuration suivante appelée *Grenouille* :

```
        // Initialisation Grenouille
        // la génération 2 sera identique à la
génération 0
        jeu = new JeuDeLaVie(5,5) ;
        setElement(jeu.grille,2,2,1);
        setElement(jeu.grille,2,3,1);
        setElement(jeu.grille,2,4,1);
        setElement(jeu.grille,3,1,1);
        setElement(jeu.grille,3,2,1);
        setElement(jeu.grille,3,3,1);
        // Affichage Génération 0
        System.out.println("Generation " +
jeu.generation);
        System.out.println(toString(jeu.grille));
```

3. Le résultat attendu est celui-ci :

```
>java -cp . ProgrammeJeuDeLaVie


Generation 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 1 1 1 0 0
0 1 1 1 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0

Generation 1
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 0 0 1 0 0
0 1 0 0 1 0 0
0 0 1 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0

Generation 2
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 1 1 1 0 0
0 1 1 1 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0

>Exit code: 0
```

6. Exercice 5 : Opération `toHTML()` pour le type **JeuDeLaVie**

On souhaite disposer d'une opération **`toHTML()`** qui retourne une chaîne contenant la grille du jeu au format HTML.

```
    toHTML() : JeuDeLaVie -> String
```

Pour une grille contenant :

```
0 1 0
1 0 1
0 1 0
```

On souhaite obtenir le résultat suivant :

```
<table border="1"><caption>Génération 0</caption>
<tr><td></td><td class='on'></td><td></td></tr>
<tr><td class='on'></td><td></td><td class='on'></td>
</tr>
<tr><td></td><td class='on'></td><td></td></tr>
</table>
```

Indications pour l'opération **`toHTML()`**

1. Comment fabriquer la table HTML attendue ?

   Vous pourriez écrire **`toHTML()`** en vous inspirant de **`toHTML(matrice)`**,

   MAIS on peut aussi atteindre le résultat attendu en modifiant le résultat de **`toHTML()`**

   où la balise `<table border="1">`

   doit être remplacée par celle ci :

   ```
   <table><caption>Génération
   "+jeu.generation+"</caption>
   ```

   et où les balises `<td>0</td>` et `<td>1</td>` doivent être remplacées par `<td></td>` et `<td class='on'></td>` respectivement.

2. Comment faire en **Java** ?

   ```
   toHTML(jeu.grille).replace("<table
   border=\"1\">"

   , "<table><caption>Génération
   "+jeu.generation+"</caption>");
   ```

1. Ajouter l'opération **`toHTML()`** à votre classe **ProgrammeJeuDeLaVie**
2. Ajouter à votre **`main()`** les instructions permettant d'afficher les versions HTML des grilles du Jeu de La Vie.
3. On souhaite placer toutes les versions HTML des grilles du Jeu de La Vie dans un fichier résultat qu'on puisse ensuite ouvrir dans un navigateur.

   En utilisant la fonction suivante (que vous placerez dans **ProgrammeJeuDeLaVie.java**), modifier votre **`main()`** pour que toutes les grilles soient accumulées dans un fichier enregistré sur votre disque.

   ```
     /*
       Fonctions (de librairie) fournies par la
   classe
     */
     static void saveTo(String nomFichier, String
   chaine) throws Exception {
       String css = "<style>";
       css += "table";
       css += "{";
       css += "border-collapse:collapse;";
       css += "border: 1px solid black;";
       css += "width:100px;";
       css += "display: inline-table;";
       css += "}";
       css += "tr,td { border: 1px solid black;
   height:20px;}";
       css += ".on { background-color:grey; }";
       css += "</style>";

       String debutPage = "<html><head>" ;
       debutPage += "<title>TP: Jeu de la
   vie</title>" ;
       debutPage += "<meta http-equiv='Content-Type'
   content='application/xhtml+xml; charset=UTF-8'
   />" ;
       debutPage += css ;
       debutPage += "</head><body>" ;

       String finPage = "</body></html>" ;

       PrintStream out = new
   PrintStream(nomFichier);
       out.print(debutPage);
       out.print(chaine);
       out.print(finPage);
       out.close();
     }
   ```

4. Ouvrir le fichier HTML produit dans un navigateur.

   Le résultat attendu est celui-ci.

7. Avant de partir

1. Enregistrer vos programmes sous **webetud2**.
2. N'oubliez-pas de vous déconnecter.