université PARIS DIDEROT PARIS 7

INITIATION À LA PROGRAMMATION (IP2)

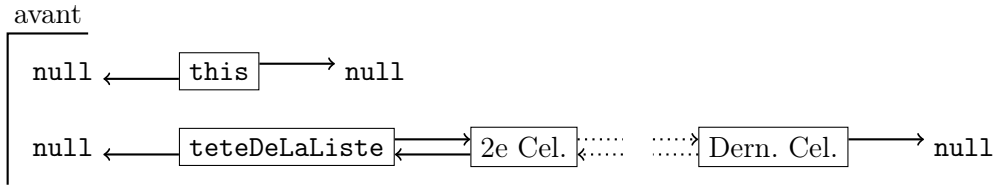
TP6 : Liste doublement chaînée et automate cellulaire

Le but de ce TP est d'implémenter une version simplifiée d'automate Cellulaire, vue comme une liste doublement *chaînée de booléens, qui indique si la cellule est vivante ou morte.*

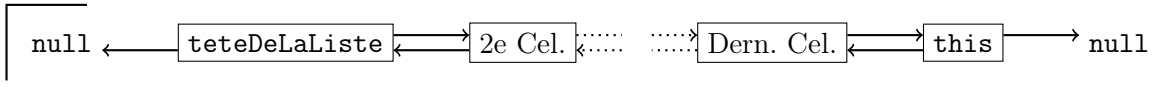
Exercice 1 [Liste doublement chaînée]

1. Créer une classe `Cellule` contenant :
   (a) Trois attributs privés : `precedente` et `suivante` de type `Cellule`, `noire` de type `boolean`.
   (b) Les accesseurs pour ces attributs.
   (c) Un constructeur `Cellule(boolean noire)` qui initialise l'attribut `noire` avec l'argument, et les deux autres attributs à `null`.
   (d) Une méthode `void affiche()` qui imprime, *sans retourner à la ligne*, un dièse `#` si `noire==true` et un tiret `-` si `noire==false`.
2. Ajouter un constructeur `Cellule(boolean noire, Cellule teteDeLaListe)` qui ajoute la cellule en création (this) *à la fin* de la liste dont la première cellule est `teteDeLaListe`.

avant



après

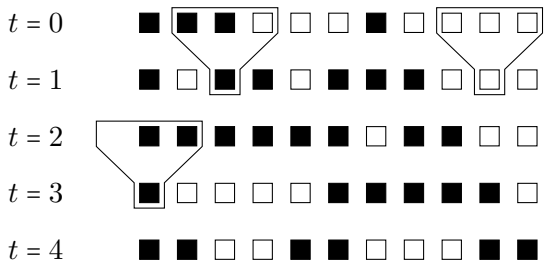


3. Tester tout ceci dans une classe principale `AutomateMain`, par exemple avec le code suivant :

```
Cellule liste = new Cellule(true); // crée une liste contenant une seule Cellule
new Cellule(true,liste);           // ajoute une Cellule à la fin de la liste
new Cellule(false,liste);           // ajoute une Cellule à la fin de la liste
liste.affiche();                              // affiche #
liste.getSuivante().affiche();                // affiche #
liste.getSuivante().getSuivante().affiche(); // affiche -
```

Il doit s'afficher : `##-` .

Exercice 2 [Automate] La suite décrit le fonctionnement d'un automate cellulaire.
A chaque instant t, chaque cellule est soit noire (`noire==true` dans les programmes) soit blanche (`noire==false` dans les programmes). L'état d'une cellule donnée à l'instant $(t+1)$ dépend de l'état de ses voisines à l'instant t, ainsi que de son propre état à l'instant t.
Considérons la règle de l'unanimité, c'est-à-dire que la cellule d'indice i à l'étape $(t+1)$ est blanche si les cellules $(i-1)$, i et $(i+1)$ portent le même état à l'étape t, voir figure ci-dessous. (Par convention, la cellule à gauche de la première (resp. à droite de la dernière) est toujours considérée comme blanche).



Sont encadrés des exemples d'application de la règle de l'unanimité. À $t = 1$, la 3ème cellule est noire car, à $t = 0$, les états de cellules 2, 3 et 4 ne sont pas identiques. De façon analogue, l'avant-dernière cellule devient blanche à $t = 1$ car ses deux voisines ont le même état que le sien à $t = 0$. Le troisième cadre souligne le fait que la première cellule considère que sa voisine de gauche est blanche.

1. Faire une classe `Automate` contenant :
   (a) Un attribut `Cellule premiereCellule`.
   (b) Un constructeur `Automate()` qui initialise à la liste vide.
   (c) Une methode `void initialisation()` qui initialise la liste comme à la figure précédente, à $t = 0$.
   (d) Une méthode `void affiche()` qui affiche les cellules en utilisant la méthode du même nom de la classe `Cellule`, puis retourne à la ligne. Nous pouvons écrire une méthode auxiliaire dans la classe `Cellule`.
2. Dans la méthode `main` dans une autre classe `Teste`, créer un `Automate`, l'initialiser (grâce à la méthode de la question 1c), puis l'afficher. Il doit s'afficher : `###---#----` .

Exercice 3 [Mise à jour] On veut créer une fonction qui change le statut `noire` des cellules en fonction du temps. Il n'est pas possible de faire ceci avec un seul parcours de la liste : la mise à jour prématurée d'une cellule peut changer le résultat de la mise à jour de sa voisine !

1. Ajouter à la classe `Cellule` un attribut `prochainEtat` de type `boolean`. Changer les constructeurs pour que celui-ci soit toujours initialisé à `false`.
2. Ajouter dans la classe `Cellule` une méthode `prochaineEtape()` qui met `prochainEtat=true` si la cellule sera noire à l'instant suivant (et `prochainEtat=false` si elle sera blanche) en suivant la règle de l'unanimité.
3. Ajouter ensuite une méthode `void miseAJour()` qui met à jour la valeur de `noire` à celle stockée dans `prochainEtat`.
4. Créer dans la classe `Automate` la méthode `uneEtape()` qui parcourt la liste *deux fois*, la première fois en appelant `prochaineEtape()` sur chaque cellule, la seconde fois en appelant `miseAJour()` sur chaque cellule.
5. Tester : vous devez obtenir le résultat de $t = 1$ de la figure précédente.

Exercice 4 [Quelques améliorations]

1. Ajouter une méthode `void nEtapes(int n)` qui affiche d'abord l'état courant, puis effectue n étapes successives, en affichant les étapes intermédiaires.
2. Tester avec $n = 4$ et comparer avec la figure de la page précédente.
3. Ajouter un constructeur `Automate(String str)` qui crée l'automate où, initialement, la cellule i est noire si `str.charAt(i)=='#'`, ou est blanche si `str.charAt(i)=='-'`.
   Par exemple, si l'on donne en argument la chaîne `###---#----`, on doit obtenir exactement le même état initial que précédemment.
4. Tester avec diverses chaînes de caractères, surtout celle avec un seul `#` au milieu.