Combats d'automates cellulaires

Les automates cellulaires

Un *automate cellulaire* est un ensemble de *cellules* disposées suivant une grille rectangulaire théoriquement infinie.

Les cellules ne peuvent prendre qu'un nombre fini d'états et évoluent au cours du temps suivant une *règle* immuable. L'état d'une cellule au temps $t + 1$ dépend à la fois :

- de son état au temps t,
- de l'état au même temps t d'un nombre fini d'autres cellules formant son *voisinage*.

Dans un automate cellulaire, la forme du voisinage et la règle d'évolution sont communes à toutes les cellules de l'automate.

Le plus célèbre des automates cellulaires est le *jeu de la vie* décrit par exemple sur Wikipedia.

La grille

La grille où évolue l'automate est théoriquement infinie. Pour permettre de définir le voisinage des cellules du bord de la grille, il est nécessaire de définir la méthode d'extension de la grille.

Plusieurs méthodes existent :

- par répétition : $\mid a\ b\ c \ldots x\ y\ z \mid \rightarrow a\ a\ a \mid a\ b\ c \ldots x\ y\ z \mid z\ z\ z$ (à appliquer sur chaque ligne et colonne)
- par périodicité : $\mid a\ b\ c \ldots x\ y\ z \mid \rightarrow x\ y\ z \mid a\ b\ c \ldots x\ y\ z \mid a\ b\ c$
- par symétrie-1 : $\mid a\ b\ c \ldots x\ y\ z \mid \rightarrow c\ b\ a \mid a\ b\ c \ldots x\ y\ z \mid z\ y\ x$
- par symétrie-2 : $\mid a\ b\ c \ldots x\ y\ z \mid \rightarrow c\ b \mid a\ b\ c \ldots x\ y\ z \mid y\ x$
- constant(e) : $\mid a\ b\ c \ldots x\ y\ z \mid \rightarrow e\ e\ e \mid a\ b\ c \ldots x\ y\ z \mid e\ e\ e$ où e est l'un des états de l'automate.

Notez qu'il n'est pas indispensable d'implanter toutes ces méthodes mais votre application devra permettre d'ajouter facilement de nouvelles méthodes.

Les états

Dans le cadre de ce projet A31, l'application que vous développerez ne fera appel qu'à des automates à deux états, MORT et VIVANT.

Cependant, la conception de votre application doit anticiper une future extension aux automates ayant un seul état associé à la mort mais plusieurs états associés à la vie.

Les combats

Les combats se déroulent entre deux automates à deux états.

Phase préliminaire

Les joueurs s'entendent sur :

1. les dimensions effectives de la grille ;
2. la méthode d'extension la grille ;
3. le nombre initial de cellules vivantes pour chacun des deux automates ;
4. le nombre d'itérations de la phase de jeu.

Phase d'initialisation

Chaque joueur choisit un type d'automate dans la liste proposée par l'application. Les deux joueurs ne peuvent pas utiliser le même type d'automate : les règles d'évolution seront donc différentes pour chaque joueur.

L'application choisit aléatoirement quel est le premier joueur à sélectionner son automate.

Ensuite, chaque joueur place à tour de rôle une cellule vivante de son automate sur la grille pour atteindre le nombre initial de cellules vivantes choisi. Il est interdit de placer une cellule vivante sur un emplacement déjà occupé par une autre cellule vivante. Le premier joueur à placer une cellule vivante sur la grille est celui qui a choisi en second son automate.

Phase de jeu

Une fois cette première phase terminée, les automates sont démarrés.

À la fin de chaque itération, pour chaque emplacement de la grille occupé par deux cellules vivantes, l'application provoque un combat aléatoire entre les deux cellules et l'une des deux meurt.

Le jeu s'arrête :
- soit lorsque toutes les cellules d'un automate sont mortes — l'autre automate est alors déclaré vainqueur,
- soit lorsque le nombre maximal d'itérations est atteint. Dans ce dernier cas, le joueur dont l'automate a le plus de cellules vivantes est déclaré vainqueur.

Consignes

Le projet est à faire en binôme.

L'application doit :
- reposer sur une architecture MVC ;
- faire appel à plusieurs patrons de conception vus en cours. Ces patrons devront être utilisés à bon escient et non uniquement pour satisfaire cette consigne ;
- utiliser des énumérations pour modéliser les ensembles d'états des automates ;
- définir la règle d'évolution d'un automate à l'aide d'une implantation d'interface, cette implantation garantissant l'existence d'une opération associant à plusieurs états (passés) un état (futur). À titre d'exemple, la déclinaison en langage Java de cette interface pourrait ressembler à cela :

```
public interface Rule< State extends Enum<State> > {
    State next( State ... states );
}
```

- implanter au moins deux automates :
  — le jeu de la vie,
  — l'automate de Fredkin.
- offrir une interface graphique permettant de
  — choisir les valeurs des paramètres du jeu,
  — afficher la grille sur laquelle évoluent les automates.

Conseils pour l'affichage

L'évolution des automates doit pouvoir être visualisée en « temps réel ». Pour obtenir un rafraîchissement régulier de l'affichage, vous devrez exécuter des instructions dans un fil d'exécution (« thread ») différent de l'Event Dispatch Thread (EDT) utilisé par `AWT` et `Swing`.

Ainsi, l'instruction

```
new Thread( () -> myMethod( my args ) ).start( )
```

est équivalente à l'instruction `myMethod( my args )` mais avec une exécution sur un autre fil que le fil courant.

Pour permettre à l'EDT de mettre à jour l'affichage, vous devrez mettre en pause votre fil d'exécution avec la méthode `sleep( long )` de la classe `Thread`.

Rendu

Vous rendrez une archive exécutable au format `.jar` contenant en outre :

- Un rapport au format .pdf,
- un diagramme de classes de votre application (sous forme d'image),
- les fichiers sources avec une arborescence reflétant les packages (le dossier `src` de votre appli) documentés par des balises javadoc.

Le rapport doit contenir :

- une description succincte de votre application (2 à 10 lignes),
- un mode d'emploi,
- les patrons de conception mis en œuvre, en expliquant pour chacun à quel problème de conception il répond,
- la manière d'étendre cette application avec :
  - de nouveaux automates à 2 états,
  - de nouvelles méthodes d'extension de la grille,
  - de nouveaux voisinages autres que ceux de Moore ou de Von Neumann,
  - des automates à plus de deux états.

Pour créer une archive `.jar`, vous pouvez regarder :

- le tutoriel sur Moodle dédié à JDev,
- la page Wikipedia,
- la documentation d'Oracle.