TP Linda/TSpaces

1 Linda/TSpaces

Un **Tuple** de valeurs est un n-uplet ordonné de valeurs. Les éléments sont de types quelconques (entier, booléen,... ou même tuple). Par exemple, les données suivantes sont des tuples : `[10 'A' true]`, `[1 2 3]`, `["azerty" 1]`, `['A' [10000 false] 2.3]`.

Un tuple **motif** (**template**) est un n-uplet dont certains des composants sont des types, qui représentent « n'importe quelle valeur de ce type ». Par exemple `[?int true]` est un motif et les tuples `[6 true]` et `[2 true]` correspondent au motif.

Un **espace de tuples** (TupleSpace) est une zone partagée de stockage de tuples. Sur un TupleSpace ts, les méthodes suivantes existent :

- `ts.write(tuple)` : ajoute un tuple dans l'espace partagé.
- `ts.take(template)` : cherche dans l'espace un tuple qui correspond au motif ; ce tuple est retiré de l'espace et renvoyé à l'appelant ; renvoie `null` si aucun tuple ne correspond.
- `ts.waitToTake(template)` : comme take, mais bloquant si aucun tuple ne correspond.
- `ts.read(template)` : comme take, sans le retirer de l'espace.
- `ts.waitToRead(template)` : comme read, mais bloquant si aucun tuple ne correspond.
- `ts.scan(template)` : comme read, en renvoyant l'ensemble des tuples qui correspondent au motif.
- `ts.delete(template)` : enlève *tous* les tuples correspondant au motif.
- `ts.eventRegister(command, template, callback)` : enregistre un callback qui sera appelé chaque fois que la commande (qui peut être `TupleSpace.WRITE` ou `TupleSpace.DELETE`) est effectuée pour un tuple correspondant au motif indiqué.

2 TSpaces en java

Dans l'implantation utilisée (IBM TSpaces), la zone de tuples est centralisée (1 serveur unique) et accessible à distance (multiples clients situés sur des machines distantes).

- Création d'un tuple valeur : cf classe `com.ibm.tspaces.Tuple` :

```
Integer myval = new Integer(45);
Tuple tt = new Tuple("foo", myval);
```

- Création d'un motif

```
Tuple t1 = new Tuple("foo", new Field(Integer.class));
Tuple t2 = new Tuple(new Field(String.class), new Field(Integer.class));
Tuple t3 = new Tuple(new Field(String.class), new Integer(45));
```

- Accès à un champ :

```
String s = (String) (tt.getField(0).getValue());
Integer val = (Integer) (tt.getField(1).getValue());
```

3 À faire

L'objectif est de réaliser un tableau blanc partagé (whiteboard) sur lequel plusieurs utilisateurs (clients) peuvent dessiner. Tout ce que fait l'un est visible par tous les autres. Un utilisateur peut dessiner des traits ou des points de diverses couleurs qui sont représentés par un `ColoredShape` (qui est une paire `java.awt.Shape` et `java.awt.Color`). Un utilisateur peut aussi tout effacer ou imposer une rotation à tous les tuples. Enfin un client peut demander un accès exclusif pour interdire aux autres de diffuser des rectangles.

Les tuples manipulés sont :

— `["Whiteboard", Command.DRAW, shape]` où `shape` est un `ColoredShape Color`.
— `["Whiteboard", Command.ERASEALL ]`.
— `["Whiteboard", Command.LOCK ]`.
— `["Whiteboard", Command.ROTATE, angle ]` où l'angle est un `Integer`.

Tous les clients sont abonnés au dépôt de l'un de ces tuples, pour réaliser l'action correspondante à la réception. En outre, un nouveau client qui arrive en cours de session doit aussi récupérer toutes les commandes `DRAW` ayant été précédemment effectuées.

4 Compilation/exécution

— La documentation de TSpaces est dans
`html/index.html`
(classes `Tuple` et `TupleSpace`).
— Eclipse : ajouter `tspaces.jar` pour compiler (après avoir créé le projet, faire menu **Project** / **Properties** / **Java Build Path** / **Libraries** / **Add external JARs** et sélectionner `tspaces.jar`).
— Compilation : utiliser le projet Eclipse ou faire `make`.
— Démarrage d'un serveur sur la machine locale :
`./runserver` ou `./runserver -p numéro-port`
— Démarrage d'un client :
  — `make run` (équivalent au suivant)
  — `java -classpath .:./tspaces.jar whiteboard.Whiteboard`
  — `java -classpath .:./tspaces.jar whiteboard.Whiteboard machine-serveur`
  — `java -classpath .:./tspaces.jar whiteboard.Whiteboard machine-serveur numéro-port`
  — Démarrage d'un client solution (implantation complète) :
  `cd Solution; make run`

Bien évidemment, le TP n'a d'intérêt qu'avec plusieurs clients !

5 Déroulement optimal de la séance

1. Traiter le dépôt et la récupération des `DRAW` : `TODO1` dans `Whiteboard.addShape`, constructeur `Whiteboard`, callback `Whiteboard.call`.
2. Traiter l'effacement (erase) : `TODO2` dans `Whiteboard.eraseAll`, constructeur `Whiteboard`, callback `Whiteboard.call`
3. Ajouter l'accès exclusif : le tuple particulier `["Whiteboard", Command.LOCK]` doit être présent pour pouvoir diffuser un rectangle. Un client peut (attendre de) prendre ce tuple (`waitToTake`) et interdire ainsi à tous les autres de diffuser des rectangles (`TODO3`).
4. S'occuper de la récupération de l'état courant au démarrage : `TODO4` dans constructeur.
5. Ajouter une action permettant de retourner tous les rectangles (`TODO5`).