UPMC/Licence/Info/2I013
Flowdroid – Le moteur

Janvier 2015

Revue de code

Modéliser

ATTENTION

- coordonnées matrices : ligne/colonne
- coordonnées écran : abscisses/ordonnées, origine en haut, à gauche

Prévoir le changement d'unité : les coordonnées des événements tactiles sont des coordonnées écran en «pixels» alors que les coordonnées de la grilles sont en «cases»

Modéliser

Abstraitement : un *chemin* est un ensemble de *cases*

TOUTEFOIS

Pourquoi une «case» devrait-elle contenir un «chemin» ?

Abstraire

Le *modèle* est une grille à 2 dimensions
$\Rightarrow$ toutes les fonctions sur les cases ont 2 paramètres.

```
Point p5 = new Point(4,1);
[...]
fm.getTab().get(2).ajoutPoint(p5);
```

est à proscrire au profit de

```
fm.addPoint(p5, 4, 1)
```

voire

```
fm.addPoint(p5)
```

puisque p5 connaît sa position

Abstraire

Éviter `g.getGrille()[x][y]` lui préférer `g.get(x,y)`

Éviter

```
plateau.getCase(x,y).getPUB()
   .caseEstUneExtremite(plateau.getCase(x, y))
```

préférez `plateau.estExtremite(x,y)`
(ici, les cases contiennent leur chemin)

Abstraire

Sans excès

Sachez ce que vous faites :

```
this.aUnVoisin(plateau.getCase(x,y).getX(),
               plateau.getCase(x,y).getY())
```

n'est autre que
`this.aUnVoisin(x,y)`

Placer les fonctionnalités (méthodes) au bon endroit
`aUnVoisin` est plutôt une méthode de `plateau` (qui est la `Grille`) pas de `this` (ici, `FlowModel`)
Ça se discute

Abstraire

Soyez économes : ne faites pas

```
new Point(currentPoint.getX(), currentPoint.getY()+1).equals(p) ||
new Point(currentPoint.getX(), currentPoint.getY()-1).equals(p) ||
new Point(currentPoint.getX()+1, currentPoint.getY()).equals(p) ||
new Point(currentPoint.getX()-1, currentPoint.getY()).equals(p)
```

faites plutôt

```
p.estVoisin(currentPoint)
```

1. pas d'allocation inutile (`new Point`)
2. la relation de voisinage est une relation entre points

Abstraire

Utilisez des constantes symboliques, des variables ou des méthodes :

- pas de : `if (a.getVal() != -1)`
- encore moins de : `if (!(cheminFini.size() == 3))`

Abstraire

C'est une mauvaise idée de considérer que le chemin courant est toujours le premier (cf. infra `get(0)`)

Utiliser une variable ou une méthode

Ça évite les codes cryptiques

```
for ( int i = 0 ; i < p.size() ; i++){
   for ( int j = 0 ; j < p.get(i).getpath().size(); j+
      if ([...]){
      PathUnderBuilding a = p.get(0);
       p.set ( 0 , p.get(i));
       p.set ( i , a);
      }
   }
}
```

144 caractères

```
((c.getX() == p.get(i).getpath().get(j).getX())&&
(c.getY() == p.get(i).getpath().get(j).getY())&&
(c.getC() == p.get(i).getpath().get(j).getC()))
```

149 caractères

```
(A.getCheminCase()!=this && A.getCheminCase().getColor()==this.color &&
A.getCheminCase().getPath().indexOf(A)==A.getCheminCase().getPath().size()-1)
```

166 caractères

```
(neSortPasDuPlateau(x-1,y) &&
pastilleInitiale.getPUB().getListeChemin().get(pastilleInitiale.getPUB().
getListeChemin().size()-1).equals(plateau.getCase(x-1,y)))
```

216 caractères

```
(cour.getChemin().getcheminPath().get(cour.getChemin().getcheminPath().
size()-1).getCouple().getX()+1 == x &&
cour.getChemin().getcheminPath().get(cour.getChemin().getcheminPath().
size()-1).getCouple().getY()+1 == y)
```

251 caractères

```
((this.getPathesUnderBuilding().get(0).getpath().
get(this.getPathesUnderBuilding().get(0).getpath().size()-1).getX() == x ) &&
(this.getPathesUnderBuilding().get(0).getpath().
get(this.getPathesUnderBuilding().get(0).getpath().size()-1).getY()+1 == y))
```

609 caractères !

```
((allPath.get(cheminStarter).getDernierElement()==
allPath.get(indiceCheminRegroup
e).getDernierElement())&&
(((x-1)==allPath.get(cheminStarter).getDernierElement().getX())&&
(y==allPath.get(cheminStarter).getDernierElement().getY()))||
(((x+1)==allPath.get(cheminStarter).getDernierElement().getX())&&
(y==allPath.get(cheminStarter).getDernierElement().getY()))||
((x==allPath.get(cheminStarter).getDernierElement().getX())&&
((y-1)==allPath.get(cheminStarter).getDernierElement().getY()))||
((x==allPath.get(cheminStarter).getDernierElement().getX())&&((y+1)==
allPath.get(cheminStarter).getDernierElement().getY())))
```

Programmation objet

Préférer

```
public class PathUnderBulding extends ArrayList<Case> {
    private int couleur;
```

à

```
public class PathUnderBulding {
    private ArrayList <Case> CheminsEnConstructions;
    private int couleur;
```