Rapport Programmation Concurrente: Files d'entiers

BENARD Thomas, L3 INFORMATIQUE

13 November 2018

A travers ce projet nous allons mettre en place une file d'entiers de longueur maximale 20. Cette même file sera partager par deux threads qui vont répéter des opérations que nous allons détailler dans la suite dans ce rapport. Tout le fonctionnement sera modéliser à travers une interface graphique.

1 Introduction

Pour la bonne réalisation de ce projet, il est important de bien séparer chaque partie de celui-ci. Il faudra tout d'abord effectuer un travail préliminaire afin de bien comprendre les étapes à suivre et de poser les éventuels problèmes. Cette première étape sera évoquée dans une première partie. Par la suite nous allons nous interesser à l'architecture du programme. Dans cette partie nous détaillerons le fonctionnement de chaque classe.

2 Préliminaires

Avant de commencer la réalisation du projet, il faut tout d'abord décomposer toutes les fonctionnalités présentes dans celui-ci. L'architecture choisie sera donc la suivante :

- une classe servira à mettre en place l'interface graphique
- une classe gérera la gestion de la file
- un Thread pour l'ajout des nombres dans la file
- un Thread qui enlève les nombres de la file

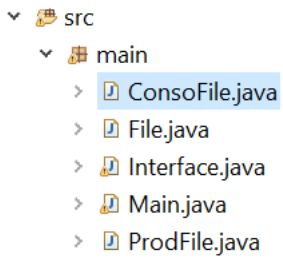

FIGURE 1 – Différentes classes java du programme

3 Architecture du programme

3.1 Threads

Le programme est constitué de deux threads. Le premier,ProdFile, est chargé d'ajouter un nombre entier aléatoire choisi entre 1 et 100 à la fin de la ligne. Le deuxième, ConsoFile, va lui enlever le premier nombre de cette même file.

3.1.1 ProdFile : Ajout de nombre dans la file

La classe ProdFile fournit des entiers aléatoires à la file. Elle appelle la fonction add de la classe File. Un nombre aléatoire entre 1 et 100 va être ajouter à la fin de la file.

```
1
2  public void run() {
3                  while(!isInterrupted()) {
4                          if(this.pause) {
5                                  try {
6                                          synchronized(this) {
7                                                  wait();
8
9                                          }
10                                 }catch(InterruptedException e) {
11
12                                 }
13                         }
14                         last= file.add();
15                         try {
16                                 sleep(sleep);
17                         }catch(InterruptedException e) {
18                                 this.interrupt();
19                         }
20                 }
21         }
```

3.1.2 ConsoFile : Retrait du dernier nombre de la file

La classe ConsoFile va faire l'inverse de ProdFile, elle va retirer un nombre au début de la file. Elle va faire appelle à la fonction remove de la classe File pour se faire.

```
1  public void run() {
2
3                  while(!isInterrupted()) {
4                          if(this.pause) {
5                                  try {
6                                          synchronized(this) {
7                                                  wait();
8
9                                          }
10                                 }catch(InterruptedException e) {
11
12                                 }
13                         }
14                         last=file.remove();
15                         try {
16                                 sleep(sleep);
17                         }catch(InterruptedException e) {
18                                 this.interrupt();
19                         }
20                 }
21         }
```

Si on met le programme en pause les threads vont arrêter d'ajouter ou de retirer des nombres de la file et vont être mis en attente jusqu'à ce que l'utilisateur relance l'opération.

3.2 Gestion de la file : File.java

La fonction add permet dé générer un nombre aléatoire entre 1 et 100. Elle vérifie aussi si la file a atteint sa taille maximale. Si c'est le cas, le thread ProdFile va être mis en attente. Sinon, le nombre généré est ajouté dans la file. Elle permet aussi d'afficher sur la console le nombre généré qui va être ajouté à la file.

```
public synchronized int add() {

                int x= (int)Math.round(Math.random()*(100-1)+1);

                while(liste.size()==maxSize) {
                        try {
                                wait();
                        }catch(InterruptedException e) {
                                e.printStackTrace();
                        }
                }

                this.liste.add(x);
                String file=affiche();
                System.out.println("Add"+file);
                notifyAll();
                return x;
        }
```

La fonction remove vérifie si la file est remplie. Si la file est vide alors le thread ConsoFile va être mis en attente. Dans le cas inverse elle va enlever le premier nombre de la file. Elle permet aussi d'afficher sur la console le nombre qui va être retiré.

```
public synchronized int remove() {

                while(liste.isEmpty()) {
                        try {
                                wait();
                        }catch(InterruptedException e) {
                                e.printStackTrace();

                        }
                }
                int rm = liste.get(0);
                this.liste.remove(0);
                if(!liste.isEmpty()) {
                        String file= affiche();
                        System.out.println("Rm : "+file);
                }
                else {
                        System.out.println("Rm : []");
                }

                notifyAll();
                return rm;
        }
```

3.3 L'interface graphique

La classe Interface est chargé de modéliser l'interface graphique de notre programme. La construction de celle-ci est divisée en 3 JPanels. Le premier pour l'affichage de la file, un autre pour l'affichage des nombres ajoutés et enlevés et enfin un dernier JPanel pour l'affichage du bouton en bas.

```
while(true) {
                        txt2.setText(file.affiche());
                        txtC1.setText(prod.getNom()+", ajout de : "+prod.getLast());
                        txtC2.setText(cons.getNom()+", retrait de : "+cons.getLast());
                }
```

L'interface se met à jour dès que la file et les éléments ajoutés et retirés sont mis à jour.

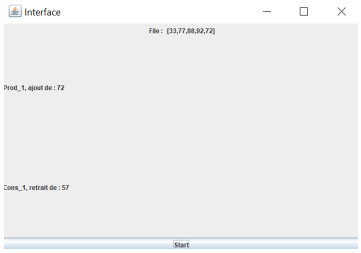

FIGURE 2 – Différentes classes java du programme

4 Conclusion

Ce projet était une bonne occasion de se perfectionner dans la programmation ainsi que de se perfectionner dans les languages. Par ailleurs, la réaction de se compte rendu a permis de découvrir le fonctionnement de latex qui nous sera utile pour la suite de nos études.Nous regrettons cependant le fait de ne pas avoir pu rendre le travail demandé en Python.