3D - TD Java 2022 - 2023 - 4 heures

Antoine Tauvel, ENSEA

Décembre 2023

1 Introduction

Vous aurez besoin pour ces deux séances d'un PC avec IntelliJ installé. L'installation de JavaFX (indispensable en TP) n'est pas obligatoire.

2 Concurence

2.1 Une classe qui ne sert à rien

Créez une classe Service composé d'une chaîne de caractère nom et d'une méthode nommée traitement qui :

- Affiche dans la console "Bonjour, je suis le processus *Nom du processus*"
- Tire un nombre aléatoire nommé durée compris entre 250 et 2000.
- Affiche dans la console "Mon traitement prendra *nombre aléatoire* milliseconde".
- S'endort pour ce nombre aléatoire de milliseconde.
- Affiche dans la console "Ici le processus *Nom du processus*, j'ai finis de travailler".

Fabriquez un constructeur à cet objet puis testez dans une classe à part votre classe Service.

2.2 Lancement de plusieurs processus

Modifiez la classe Service de manière à la transformer en thread (aux choix : héritage de thread ou implémentation de runnable). Modifiez votre classe de test de manière à lancer 10 Services en parallèle. Les références vers Services sont stockés dans un ArrayList A l'aide de la méthode join, synchronisez les processus entre eux.

2.3 Pose de verrou sur une ressource

Lorsque plusieurs processus utilisent la même ressource, il est nécessaire de synchroniser l'usage de cette ressource en spécifiant un accès unique à la ressource. Cela peut se faire entre autre par le biais de la syntaxe synchronized. Le code entre crochet derrière synchronized attend qu'une ressource soit libérer. Voici un exemple :

```
Object lock = new Object();

new Thread() {
  public void run() {
    System.out.println("Locking ");
    synchronized (lock) {
      System.out.println("locked");
    }
    System.out.println("Released");
  }
}.start();
```

Modifiez la classe Service de manière à ce qu'elle ait une référence vers un objet File Writer. L'instance est créer dans la classe de test à l'aide de la méthode FileWriter (String fileName, boolean append). Ici append est faux.
A l'issue de l'attente aléatoire, écrivez dans le fichier de manière synchronisé le nom de la méthode.
Ouvrez le fichier texte passé en argument et observez la synchronisation.

3 Création d'un petit serveur

Créez dans votre projet précédent une nouvelle classe nommé SwitchBoard. Cette classe est constitué d'une ArrayList de Service.

Vous aurez besoin pour tester cet exercice d'un outil de type netcat.

Voici le code de test :

```
public class SwitchBoard {
    static ArrayList<Service> serviceList =new ArrayList<Service>();

    public SwitchBoard(){
    }

    public static void main(String[] arg) {
        final int port = 6666;
        ServerSocket listener;
        Socket socket;

        try {
            listener = new ServerSocket(port);
            while (true) {
                socket = listener.accept();
                serviceList.add(new Service( /* Quels paramètres ? */));
                System.out.println("New connection on the port");
            }
        } catch (IOException exc) {
            System.out.println("Connexion problem");
        }
    }
}
```

En vous inspirant du code ci-dessus, fabriquez un serveur. On écrira dans le socket les informations qui auparavant allaient sur la console.

4 Optionnel : création d'un deadlock

Un deadlock peut avoir lieu lorsque deux thread veulent prendre deux ressources, une dans l'ordre ressource1-¿ressource2 et l'autre dans l'ordre ressource2-¿ressource1.

Pour assurer un deadlock, il faut être sur qu'il y ait un changement de processus entre les deux prises de verrous.

Dans une unique classe Main, créez deux objets Object et deux objets Thread, prenez dans le corps de chaque thread un sémaphore puis endormez les processus.
L'affichage pourrait ressembler à celui ci-dessous.

```
1.1 Thread 01 got lock 1
2.1 Thread 02 got lock 2
1.2 Thread 01 waiting for lock 2
2.2 Thread 02 waiting for lock 1
```

5 Lambda expression

```
Soit le code suivant :
interface StringFunction {
  String run(String str);
}

public class Main {
  public static void main(String[] args) {
    // To complete : definition of two lambda expression
    // named exclaim and ask of the type StringFunction
    printFormatted("Hello", exclaim);
    printFormatted("Hello", ask);
  }
  public static void printFormatted(String str, StringFunction format) {
    String result = format.run(str);
    System.out.println(result);
  }
```

Complétez le code ci-dessus de manière à ce que exclaim renvoie la chaine suivi d'un point d'exclamation, et ask d'un point d'interrogation. Il faut bien sur utiliser des lambdas expression.

6 Usage d'API

A l'aide de l'API ouverte de la SNCF, déterminez la liste des gares de voyageur en France.

Pour cela, on devra :

Créez une classe Gare contenant un identifiant unique, le nom de la gare, et le numéro du département.
Créez dans le main une ArrayList de Gare.
Remplissez cette ArrayList à l'aide de l'exemple ci-dessous. Il faudra importer d'internet la classe Json
A l'aide de la classe JsonReader à importer de chez Oracle, analyser le paquet réponse et remplir le tableau de Gare.

```
HttpClient client = HttpClient.newHttpClient();

HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create("http:/api/records/1.0/search/?dataset=referentiel-gares-voyageurs&q=&
    sort=gare_alias_libelle_noncontraint&facet=departement_libellemin&facet=segmentdrg_libelle&
    facet=gare_agencegc_libelle&facet=gare_regionsncf_libelle&facet=gare_ug_libelle")
    .build();
HttpResponse response = client.send(request, HttpResponse.BodyHandlers.ofString());
System.out.printkn(response.body().toString());
```