Institut Galilée LEE
Année 2015-2016 Master 2

T.P. 1

Familiarisation avec Eclipse / Netbeans Java

1 Exercice

1. Définir une classe Periode contenant 2 champs : un champ Date pour la date d'arrivée, un champ de type Date pour la date de départ.
2. Définir une classe Location ayant 4 champs :
   — String lieu
   — int tarif (tarif par semaine)
   — List<Periode> reservations
   — String genre : une énumération pour studio, appartement, maison
   On suppose qu'il n'y a qu'une location par lieu.
3. Définir un serveur permettant une connexion socket sur le port 9999, et ayant un ensemble (que vous définirez) de locations, comme liste d'instances de la classe Location.
4. Définir un client qui va servir de test pour la connexion au serveur.
5. Définir côté serveur la méthode permettant de renvoyer à un client le tarif quand le client envoie un nom de lieu, modifiez le client pour tester cette méthode.
6. Définir une méthode côté serveur pour renvoyer la moyenne des tarifs par semaine des locations quand un client le demande (fixez le protocole correspondant à cette demande).
7. Définir une méthode côté serveur pour renvoyer le nombre de lieux dont les tarifs par semaine sont compris entre 100 et 200.
8. Définir un pool de taille fixe (10) pour le serveur pour répondre aux requêtes de clients. *Utilisez un pool prédéfini.* Faire donc des classes pour le traitement des requêtes.
9. Définir des classes exception permettant de gérer les cas d'erreurs (i.e. des exceptions sont retournées par la classe traitant les requêtes, le serveur gérant l'exception)
10. Définir des méthodes permettant d'ajouter, ou de supprimer, ou de modifier une location, une réservation. Testez via un client.
11. Définir un pool à taille 1 pour l'administration, c'est à dire pour les méthodes de la question précédente.
12. Définir une méthode qui teste si une réservation a été faite pour une certain lieu (et testez via un client)
13. Modifiez la tâche précédente pour que ce soit un Future, et testez toutes les 10 secondes si le résultat vrai/faux est obtenu. Utilisez un *Callable*, testez si le *future* est terminé avec une méthode *isDone*, attention à bien gérer les exceptions !
14. En supposant que le client envoie 10 lieux dont on veut tester lesquels ont des réservations, utilisez un *CompletionService* pour récupérer (et afficher) les résultats dès qu'ils sont disponibles.
15. Modifiez les méthodes de calcul précédentes pour y mettre une gestion par stream avec lambda-expressions.
16. Définissez un format XML pour les données de location. Définissez les méthodes permettant lecture/écriture entre fichier XML et instances de la classe Location.
17. Définissez une annotation de type booléen qui, quand sa valeur est *true*, permet d'enregistrer les opérations d'administration (i.e. de sauvegarder dans un fichier "log_admin.txt" le nom de la méthode appelée et les paramètres d'appel).

Quelques éléments sur la classe CompletionService :
— Création : `CompletionService<Boolean> completionService = new ExecutorCompletionService<Boolean>(executor);` (quand executor est votre 'exécuteur de service' et Boolean est le type de résultat d'un callable)
— Lancement d'un callable t via le CompletionService : `completionService.submit(t));`
— récupération d'un callable terminé (le prochain appel à take() renverra le callable terminé suivant) : `res = completionService.take();`

2 Environnements Java

Il existe plusieurs environnements Java. Il est ESSENTIEL d'utiliser la "bonne" version, et un environnement d'exécution identique à la version de compilation (principalement en cas de chargement de librairies à chaud). Le système que vous avez à l'Institut Galilée a les environnements java suivants :

- java 1.6.0_36 GNU (IcedTea6 1.8.13) (`/usr/bin/java`) : c'est la version open source présente par défaut. Elle a l'inconvénient majeure de présenter de nombreux problèmes de compatibilité avec le java standard de Sun.
- java jdk1.6.0_27, 1.6.0_33, jdk1.7.0_07, jdk1.7.0_15, jdk1.7.0_25, jdk-1.8 (`/usr/java/jdk1.../bin/java`) : versions 1.6, 1.7, 1.8 de Sun/Oracle (donc propriétaire même si les sources sont publiques).

Accès à Java 1.6 Modifier votre fichier `.bashrc` afin que la version 1.6 de java soit accessible. Pour cela, décommenter dans votre fichier `.bashrc` la ligne

```
source /export/home/users/COMMUN/.jdkrc
```

Attention : ce changement ne prend effet qu'en lançant un nouveau shell bash ou en effectuant la commande

```
. ~/.bashrc
```

Vérifier que le "bon" java est accessible avec la commande `java -version`

Accès à Java 1.7 Modifier votre fichier `.bashrc` afin que la version 1.7 de java soit accessible. Pour cela, mettre en fin de fichier .bashrc les lignes :

```
export JAVA_HOME=/usr/java/jdk
export PATH=${JAVA_HOME}/bin:$PATH
```

Terminologie et versions

- Java SE (Standard Edition) : noyau des API (lang, collections, ..., jar, ...) sans machine virtuelle.
- JRE (Java Runtime Environment) : JSE + machine virtuelle + plugin.
- JDK (Java Development Toolkit) : JRE + logiciels d'utilisation (java, javac, jar, ...).

3 Environnement Eclipse

Eclipse est un environnement de développement intégré (*IDE*) libre, extensible et polyvalent, permettant potentiellement de créer des projets de développement pour tout langage de programmation. Eclipse est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions. La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plug-in (en conformité avec la norme OSGi) : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plug-in. Plusieurs logiciels commerciaux sont basés sur ce logiciel libre, comme par exemple IBM Lotus Notes 8, IBM Symphony ou WebSphere Studio Application Developer.

Téléchargements, plugins, tutoriaux : `www.eclipse.org`

3.1 Initialisation

Modifier votre fichier `.bashrc` afin qu'Eclipse soit accessible (c'est la version 3.4.1 contrairement à ce que le nom ci-dessous pourrait faire penser). Pour cela, décommenter dans votre fichier `.bashrc` la ligne

```
source /export/home/users/COMMUN/.eclipse-3.2rc
```

Attention : ce changement ne prend effet qu'en lançant un nouveau shell bash ou en effectuant la commande

```
. ~/.bashrc
```

Remarque : En fait, c'est la version 'indigo' de Eclipse qui est lancée (donc la 3.7 et pas la 3.2). Il existe maintenant la version 'kepler' (4.3) qui a fait suite à la version 'juno' (4.2).

3.2 Projet

On prendra comme exemple le code de partage de mémoire avec thread (p. 27), classe TestRunnable.

Création d'un projet `File -> New -> Java Project`
`Project Name` : mettre `Thread_Share`
Vérifier que le JRE (*Java Runtime Environment*) est `jdk1.6.xx` ou `jdk1.7.xx`. Si ce n'est pas le cas, utiliser `Configure JREs`, puis sélectionner ou ajouter (via search si nécessaire).

Une sous-fenêtre `Project Explorer` vous permet de parcourir les projets et sources créés. Si la sous-fenêtre n'est pas visible, utiliser `Window -> Show view -> Project Explorer`.

Accès aux fichiers Un Workspace contient l'ensemble des projets créés. Le workspace est un répertoire dans votre environnement de travail, il est défini au premier lancement de Eclipse. Vous pouvez retrouver le nom du répertoire par `File -> Switch Workspace`. Chaque projet est un sous-répertoire contenant un sous-répertoire `bin` pour les exécutables, et un sous-répertoire `src` pour les sources.

Création de sources, exécution Sélectionner `Thread_Share/src` dans le `Project Explorer`, puis bouton droit et `New -> Class`. Entrer le nom de la classe `TestRunnable`, cocher `main` (puisque cette classe contient la méthode main). Puis entrer le code, sauvegarder.

Exécution par `Run`. Une fenêtre `Console` vous donne l'affichage.

4 Environnement Netbeans

NetBeans est aussi un environnement de développement intégré (IDE) pour Java, placé en open source par Sun. En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web). Il permet en outre :

— de créer et déployer des environnements complets web + java
— d'intégrer la machinerie des beans java (annotations adaptées)
— de supporter en plugin des bases de données (GlassFish en Open Source)
— de contrôler le déploiement sur un ensemble de machines

4.1 Initialisation

Modifier votre fichier `.bashrc` afin que Netbeans soit accessible. Pour cela, décommenter dans votre fichier `.bashrc` la ligne

```
source /export/home/users/COMMUN/.netbeansrc
```

dans votre `.bashrc`
Attention : ce changement ne prend effet qu'en lançant un nouveau shell bash ou en effectuant la commande

```
. ~/.bashrc
```

Remarque : c'est la version 7.3 qui sera lancée, la version 8.0.1 est aussi installée (`/usr/java/netbeans-8.0.1/bin/netbeans`). La version la plus récente est la 8.0.2

4.2 Projet et Groupe

Implanter et tester les 4 exemples sur les threads entre la page 31 et la page 40. Créer un projet pour chacun d'entre eux. Les projets peuvent être organisés en groupes. Un groupe est un ensemble de projets et est créable avec `FIle->Project Group -> New Group`. Donner un nom : `TP0`.

Création d'un projet `File -> New Project -> Java -> Java Application`
`Project Name` : mettre `Thread1`

Accès aux fichiers Un Workspace contient l'ensemble des projets créés. Le workspace est un répertoire dans votre environnement de travail, il est défini au premier lancement de Netbeans. Vous pouvez retrouver le nom du répertoire en regardant les propriétés (bouton droit, Properties) d'un projet particulier.

Création de sources, exécution Sélectionner le projet puis (bouton droit), `New -> Java Class`. Entrer le nom de la classe `CbDigest`, entrer un nom de package (par exemple `Ex2-Thread1`). Puis entrer le code, sauvegarder. Faire de même pour tous les codes.

Exécution par `Run`. Une fenêtre `Output` vous donne l'affichage.