POLYTECH – ELEC5-TR 2016–2017

# Contrôle de Langage Java

**Durée :** 1h Aucun document autorisé

*Note : la qualité des commentaires, avec notamment la présence d'affirmations significatives, ainsi que les noms donnés aux variables, et la bonne indentation rentreront pour une part importante dans l'appréciation du travail.*

**Vous choisirez de répondre à 2 questions parmi les 3 suivantes**

## Question 1

On veut représenter des villes, des capitales et des pays du monde, avec :
— pour les villes leur nom, leur nombre d'habitants et le pays auquel elles appartiennent ;
— pour les pays leur nom et leur capitale.

Trois classes seront à définir : `Pays`, `Capitale` et `Ville`.

1. Faites un schéma qui établit les relations entre ces classes. Vous indiquerez clairement la nature des relations (héritage ou composition) qui lient les classes.
2. Écrivez en Java ces trois classes, avec leurs constructeurs, leurs accesseurs et mutateurs, et leurs méthodes `toString`.
3. Écrivez la méthode `main` d'une classe `Test`, dans laquelle vous initialiserez un `Vector` de `Villes` avec des villes et des capitales de pays de votre choix de votre choix, puis vous écrirez sur la sortie standard son contenu.

Le résultat de l'exécution de la classe `Test` peut être par exemple :

```
[Nice(France) - 300000h, Xian(Chine) - 8000000h, *Rome(Italie) - 2500000h]
```

l'étoile indique que la ville est une capitale.

## Question 2

On souhaite écrire un programme qui affiche sur la sortie standard le nombre d'occurrences d'un entier `x` qui apparaît dans un tableau d'entiers. La recherche dans le tableau est effectuée par un thread. Toutefois, si le tableau possède une taille supérieure à une valeur `TAILLE_MAX`, l'espace de recherche dans le tableau est divisé en 2, et un premier thread effectue la recherche du nombre d'occurrences de `x` dans la première partie du tableau, alors qu'un second thread effectue la recherche en parallèle dans la second partie. Vous écrirez 3 classes :
— `ThreadChercheur`, un thread qui effectue la recherche du nombre d'occurrences de `x` dans une partie de tableau. À chaque fois qu'il trouve une nouvelle occurrence de `x`, il incrémente un compteur de type `Compteur`.
— `Compteur` qui gère le compteur de nombre d'occurrences partagé par les threads.
— `CompteurOccurrences`, la classe principale qui contient la méthode `main` dans laquelle vous créerez un tableau, et effectuerez la création du ou des threads qui effectue la recherche du nombre d'occurrences. La valeur `x` à rechercher est passée en paramètre programme. À la fin, la méthode `main` affiche le résultat.

## Question 3

Écrivez le client UDP, `DayTimeUDPClient.java`, permettant d'interroger le démon DayTime qui est accessible sur le port 13 d'une machine dont l'adresse est passée en paramètre au programme.

Lorsque le démon DayTime reçoit un datagramme (quel qu'il soit), il retourne au client un datagramme qui contient la date et l'heure courante. Vous afficherez le contenu de ce datagramme de retour sur la sortie standard.

Par exemple, l'exécution du programme qui interroge le serveur `server.unice.fr` donnera le résultat suivant :

```
$ javac DayTimeUDPClient.java && java DayTimeUDPClient server.unice.fr

Création de la socket locale attachée :
à l'adresse 0.0.0.0/0.0.0.0
au port 53917

Émission du datagramme
Attente de la réponse...
Réponse reçue de (server.unice.fr:13) :
16 NOV 2016 08:12:57 CET
```