Contrôle SLR201

9 novembre 2015

– 1h30 –

Barème indicatif, **avec document**

Les téléphones portables doivent être éteints et rangés dans vos sacs

Le sujet comporte 5 pages

Les deux parties de l'examen sont indépendantes.

Merci de traiter chaque partie sur des feuilles séparées.

1ère Partie – Java Avancée (10 points)

1.1 Exercice général

(0,5 point) Choisir la bonne phrase et la cocher :

☐ Une classe peut implémenter plusieurs interfaces mais doit étendre une seule classe.

☐ Une classe peut implémenter plusieurs classes mais doit étendre une seule interface.

☐ Une classe peut implémenter plusieurs classes et peut étendre plusieurs interfaces.

☐ Une classe doit implémenter une seule interface et étendre une seule classe

1.2 Exercice sur les threads.

Soit la classe suivante représentant une baignoire :

```
class Baignoire
{
     private int capacite ;
     private int quantite ;

     public Baignoire(int capacite) {
          this.capacite = capacite ;
          this.quantite = 0;
     }

     public void ajouter(int quantite){
     this.quantite += quantite ;
     if (this.quantite > this.capacite)
          this.quantite = this.capacite ;
     }

     public void retirer(int quantite)
     {
     this.quantite += quantite ;
     if (this.quantite < 0)
          this.quantite = 0 ;
     }

     public int getQuantite()
     {
     return quantite ;
     }
}
```

On imagine que le programme main crée une baignoire avec une certaine capacité.

- Puis qu'elle lance une thread `Fuite` qui vide la baignoire d'une certaine quantité (3 dans l'exemple ci-dessous) toutes les 100 ms.
- Elle lancera une thread `Robinet` qui remplit la baignoire d'une autre quantité (5 dans l'exemple ci-dessous) toutes les 100 ms.
- Enfin, elle lancera aussi une thread `Voir` qui affichera le contenu de la baignoire toutes les 100ms.

Soit :

```
public void main()
{
      Baignoire baignoire = new Baignoire(100) ;

      Fuite     fuite     = new Fuite(baignoire,3) ;
      Robinet   robinet   = new Robinet(baignoire,5) ;
      Voir      voir      = new Voir(baignoire) ;

      fuite.start() ;
      robinet.start() ;
      voir.start() ;
}
```

1) *(1 points)* Ecrire le code de la thread `Fuite`.
2) *(1 points)* Ecrire le code de la thread `Robinet`.
3) *(1 points)* Ecrire le code de la thread `Voir`.
4) *(0,5 point)* Ce programme fait-il ce qu'on en attend ?

1.3 Exercice sur la sérialisation

Soit une classe `Point`, que l'on définit, ne contenant que deux attributs `x` et `y` (ses coordonnées). Soit une classe `PolyLine` contenant uniquement une liste de `Point` sous la forme d'une `ArrayList<Point>`.

1) *(2 points)* Ecrire les classes `Point` et `Polyline`. On n'écrira que ce qui est nécessaire aux questions suivantes.
2) *(2 points)* Ecrire une méthode de la classe `Polyline` qui permet de sauvegarder une `PolyLine` dans un fichier donné.
3) *(2 points)* Ecrire une méthode de la classe `PolyLine` qui permet d'aller rechercher la sauvegarde précédente l'objet de la classe `PolyLine`.

2ème Partie – Intergiciel, RMI, JMS (10 points)

2.1 Intergiciel

Question 0 (1 point) : l'objectif d'un intergiciel est de :
(Sélectionner la réponse correcte parmi les options suivantes)

a) faciliter le développement d'applications réparties ;

b) faciliter la gestion des réseaux de communication ;

c) faciliter le développent des registres de nommage ;

d) faciliter la compilation des applications réparties.

2.2 RMI

Etant donné une interface `Magasin` :

```
public interface Magasin extends java.rmi.Remote
{
      //retourne le prix de l'ingrédient donné en paramètre
      float renvoyerPrix (String ingredient)
            throws java.rmi.RemoteException;
}
```

Et une classe `GerantMagasin` qui implante cette interface :

```
public class GerantMagasin implements Magasin
{
      float renvoyerPrix (String ingredient)
            throws java.rmi.RemoteException
      {
            …
            return prix ;
      }
}
```

Question 1 (2 points) : écrire le code d'une méthode `main` qui crée un objet de type `GerantMagasin` et qui enregistre cet objet auprès d'un registre RMI (rmiregistry). On considère que le registre RMI sera lancé sur une machine dont on connait l'adresse IP (<adresse>) et qui écoutera sur un port connu (<port>).

Question 2 (1 point) : indiquer les contraintes sur l'endroit où le rmiregistry doit être lancé (ex : ordinateur, JVM, répertoire) par rapport à l'endroit où la méthode `main` (développée pour la Question 1) sera exécutée.

Question 3 (1 point) : lors de la compilation de l'application définie ci-haut, via javac et ensuite rmic : combiens de classes de type souche (« stub ») seront générées et avec quel nom ? justifier votre réponse en indiquant comment le compilateur sait pour quelles classes il doit générer des classes stub ?

Question 4 (1 point) : quel est le rôle du paramètre `-keepgenerated` utilisé lors de la compilation ?

2.2 JMS

Question 5 (1 point) : écrire le code permettant à un client JMS d'obtenir la référence vers un service (ou registre) de nommage JNDI (où seront enregistrées les références vers des fabriques de connexion et/ou des destinations JMS). Indiquer la façon dont l'adresse et le port du service de nommage JNDI peuvent être indiqués pour ce faire.

Question 6 (2 points) : quelle sont les modalités via lesquelles un client JMS (Receiver) peut obtenir des messages à partir d'une destination JMS ? pour chaque option, que se passe-t-il lors qu'il n'y a pas de message dans la destination visée ?

Question 7 (1 point) : combiens de clients JMS peuvent recevoir un message envoyé dans une destination de type fille (« queue ») ? la même question pour une destination de type sujet (« topic ») ?