Examen 2008 - 2 h

PAR : Programmation d'Applications Réparties
Troisième année

Tous documents autorisés. Rédigez les parties RMI et EJB sur des copies séparées.

1 RMI

★ **Exercice 1. Questions de cours**

▷ **Question 1.** (1 pt) Dans le cadre d'une communication distante, qu'est ce qu'une souche et un squelette ? Quels en sont les avantages ?

▷ **Question 2.** (1pt) Lors de l'invocation d'une méthode distante Java RMI, quelle est la différence entre le passage d'un objet par valeur et le passage d'un objet par référence ?

★ **Exercice 2.** On propose les interfaces et classes suivantes :

One
```
1  public class One implements java.io.Serializable {
2    public void one() {
3      System.out.println("One.one()");
4    }
5  }
```

ITwo
```
1  import java.rmi.*;
2  public interface ITwo extends Remote {
3    public void two() throws RemoteException;
4  }
```

Two
```
1  import java.rmi.*
2  public class Two extends UnicastRemoteObject
3                   implements ITwo {
4    public Two() throws RemoteException { }
5    public void two() throws RemoteException {
6      System.out.println("Two.two()");
7    }
8  }
```

IServeur
```
1  import java.rmi.*;
2  public interface IServeur extends Remote {
3    public void a(One o) throws RemoteException;
4    public One b() throws RemoteException;
5    public void c(ITwo o) throws RemoteException;
6    public ITwo d() throws RemoteException;
7  }
```

Serveur
```
1  import java.rmi.*;
2  public class Serveur extends UnicastRemoteObject implements IServeur {
3    public Serveur() throws RemoteException { }
4    public void a(One o) throws RemoteException {
5      o.one();
6    }
7    public One b() throws RemoteException {
8      return new One();
9    }
10   public void c(ITwo o) throws RemoteException {
11     o.two();
12   }
13   public ITwo d() throws RemoteException {
14     return new Two();
15   }
16 }
```

▷ **Question 1.** (2pt) On suppose un client exécutant le code ci-dessous (avec `s` étant une variable de type `IServeur` initialisée à une référence distante sur l'objet serveur).

Pour chacun des appels, indiquez si l'appel considéré se déroule dans la machine virtuelle du client (ie, s'il s'agit d'un appel local) ou dans la machine virtuelle du serveur (ie, s'il s'agit d'un appel à distance).

De plus, Quand on démarre le serveur et le client, les JVM correspondantes créent chacune un thread correspondant à la méthode main. Dans chaque cas, indiquez également si l'appel se déroule dans le thread principal ou dans un autre Thread.

Code du client

```
1 One un = new One();
2 s.a(un);
3 s.b().one();
4 s.c(un);
5 s.d().two();
```

Méthode considérée	Exécution sur JVM		Thread d'exécution	
	serveur	client	Principal	Autre
l2 : `a()`	☐	☐	☐	☐
l2 : `one()` par `a()`	☐	☐	☐	☐
l3 : `one()`	☐	☐	☐	☐
l4 : `two()` par `c()`	☐	☐	☐	☐

▷ **Question 2.** (1pt) Que se passe-t-il si dans l'interface `IServeur` (et dans la classe `Serveur`), on remplace toutes les occurrences de `ITwo` par `Two` ?

★ **Exercice 3.** Un serveur doit fournir le service décrit par l'interface suivante :

```
1 interface extends java.rmi.Remote {
2   int addition(int a,int b) throws java.rmi.RemoteException;
3 }
```

▷ **Question 1.** (1pt) Écrivez une implémentation simple du serveur rendant le service en question, ainsi qu'un client utilisant ce service.

▷ **Question 2.** (2pt) Écrivez le code des applications client et serveur qui permet d'invoquer la méthode addition de manière asynchrone différée grâce à un *callback*.

2 EJB

▷ **Question 3.** (1pt) Quel est le nom du service de permettant de retrouver un objet distant à partir de son nom dans l'environnement J2EE ?

▷ **Question 4.** (1pt) Quelle est la méthode de ce service, appliquée à quelle classe qui permet de retrouver une référence sur un Session Bean ?

▷ **Question 5.** (1pt) Pour quelle raison faut-il parfois faire une différence entre l'implantation d'une méthode pour un appel en local et pour un appel distant ?

▷ **Question 6.** (1pt) Quel service de l'environnement J2EE permet la transparence de la sécurité ?

▷ **Question 7.** (1pt) Ecrivez un Stateless Session Bean qui fournit une méthode de conversion de dollar en euro (taux de change 1 pour 1.45)

▷ **Question 8.** (1pt) Décrivez les étapes permettant l'appel d'un Service Web décrit en WSDL depuis un client écrit en Java. Quel est le service utilisé ?

▷ **Question 9.** (1pt) Lors du traitement d'un message JMS par un Message Driven Bean qu'arrive-t-il si un exception est levée ?

▷ **Question 10.** (1pt) Qu'est ce qui est affiché lors de l'appel de la méthode myMethod par un client ?

```
1 public class MyInterceptor {
2     @AroundInvoke
3     public Object test(InvocationContext ctx) {
4        System.out.println("World");
5     }
6 }
7
8 @Stateless
```

```
@Interceptors({MyInterceptor.class})
public class MyStateless {
    public void myMethod() {
       System.out.println("Hello");
    }
}
```

▷ **Question 11.** (1pt) Quels sont les tables créées dans la base de données selon la stratégie Single Table ou Join Strategy

```

public class Courrier {
  private String destinataire;
  private String adresse;
}

public class Recommande extends Courrier {
  private Date envoi;
  private Date recu;
  private boolean signature;
}

public class Suivi extends Courrier {
  private boolean arrivé
}

```

▷ **Question 12.** (1pt) Ecrivez le code permettant de créer une instance de la classe Courrier et de la rendre persistente

▷ **Question 13.** (1pt) Ecrivez le code permettant de retrouver toutes les instances de Courrier adressés à une personne

▷ **Question 14.** (1pt) Une transaction est démarrée. Que se passe-t-il si à l'intérieur de cette transaction une méthode d'un session bean ayant l'attribut transactionnel required est appelée ?