SI4 Polytech'Nice Sophia – UNS

ECUE « Middleware Et SOC», Partie « Middleware » - Epreuve écrite individuelle

7 Décembre 2022, durée 1h/1h10 environ

Documents cours/TP NON autorisés

Comme d'habitude, il est conseillé de lire le sujet en entier avant de commencer !

F. Baude

Exercice 1 (barème approximatif : 9 points) : RMI

Le but de cet exercice est, grâce à une série de questions de synthèse, d'évaluer votre compréhension du système Java RMI. Quelques lignes de réponse (4 / 5 **maximum**) par question.

1. En Java RMI, quel est l'équivalent de ce qu'est un fichier WSDL dans la technologie des web services : vous discuterez en vous focalisant sur les opération et sur leurs paramètres
2. Dans quelle circonstance (typique), est ce que le RMI registry va avoir besoin de générer dynamiquement la classe d'un proxy/stub
3. Même un appel de méthode RMI sans valeur de retour est qualifié d'appel de méthode « synchrone ». Expliquez.
4. Un appel de méthode RMI avec ou sans valeur de retour pourrait ne pas réussir : expliquer les raisons qui font que le comportement du modèle de programmation répartie JavaRMI se résume par le terme « at most once ».
5. Pourquoi du coté d'un serveur RMI il y a l'utilisation d'un thread pool (ou plus généralement, d'un ensemble de threads) ? Quel serait l'effet si une seule thread au plus pouvait être utilisée dans l'exécution des appels entrants côté Serveur RMI ? Quel est l'impact sur la manière de programmer le code qui sera donc potentiellement exécuté par plusieurs threads du côté du serveur RMI.
6. Le RMI Registry a été lancé et écoute sur le port 2000 de la machine localhost, par exemple. Un objet RMI implémentant l'interface RMI HelloWorld écoute sur le port 10000 de cette même machine localhost. Lorsque le stub de cet objet doit être enregistré auprès du registre RMI, on peut le faire en utilisant une URL, mais, celle qu'on vous propose ci-dessous est incomplète, complétez-la. Remarquez qu'on a utilisé l'API de java.rmi.Naming (non vue en TP, mais réfléchissez !) et la clé d'enregistrement est HW

```
Naming.rebind("rmi://localhost:xxx/HW", hello);
```

L'URL débute par '`rmi://`' signifiant que le protocole utilisé est celui qui est propre à Java RMI. Pourquoi ? Pour répondre, expliquez quel est le processus, et sa nature que l'on veut contacter quand on utilise cette URL

Exercice 2 (barème approximatif : 7points) : MOM

Le but de cet exercice est, grâce à une série de questions de synthèse, d'évaluer votre compréhension de JMS/ActiveMQ. Quelques lignes de réponse (4 / 5 **maximum**) par question, à l'exception de la dernière, où il vous faudra probablement plus de lignes pour discuter les deux situations proposées.

1. En JMS, avec ActiveMQ, on a manipulé deux URLs, que sont http://localhost:8161 et tcp://localhost :61616 . Expliquer quelle différence entre les 2 URLs, et à quoi chacune d'elle nous sert.
2. Expliquer en bref ce qu'est une transaction dans le cadre d'un MOM, quelle est la nature de la ressource dite transactionnelle ; est-ce que une transaction englobe du code qui consiste à envoyer des messages, recevoir des messages, ou expliquer si ce concept de transaction s'applique ou fait aux deux.

3. Je propose ici une/ma définition de messagerie que je qualifie d'instantanée et personnalisée: on peut la définir comme le fait que l'émetteur d'un message l'envoie et que le récepteur (uniquement identifié, c'est-à-dire, pas n'importe quel récepteur) ne peut le réceptionner que si il est actuellement connecté au système (dit autrement, si il n'est pas connecté au système, il ne doit jamais recevoir le message, même plus tard). Pour l'aspect privé/personnalisé, on suppose par la suite que la paire de personnes communiquant connait par avance le nom d'une destination (au sens JMS) de messages, qui ne change pas avec le temps.

   Un MOM au standard JMS (comme ActiveMQ) permet-il de réaliser un tel mode de messagerie instantanée ? Vous répondrez en discutant selon deux situations, et cela en considérant l'utilisation de queue ou de topic JMS (vous aurez donc au maximum 4 cas de figure à considérer) :
   première situation : le consommateur du message est actuellement connecté au MOM. Dans ce cas là, quel mode de réception de message utiliser dans le code du consommateur, d'autant plus que le consommateur peut être en train de réaliser une autre tâche dans l'application ? Vous préciserez si une Destination de type Queue ou de type Topic peut être indifféremment utilisée
   Seconde situation : le consommateur du message n'est pas actuellement connecté au MOM. Dans ce cas là, quand il finit par se reconnecter, qu'est ce qu'il va se passer ? Aurez vous ou pas de l'instantané (selon la définition d'instantané posée au début du texte de la question) ? Et surtout, dites si c'est possible ou non, selon que vous utilisez queue ou topic, d'empêcher la consommation du message en attente ?

Exercice 3 (4 points) : petit scénario distribué

L'objectif de cet exercice est de comprendre ce que réalise le code proposé et de répondre à deux questions ci après.

```
import java.rmi.Remote;
import java.rmi.RemoteException;


public interface Bank extends Remote {

	public Account createAccount(String nom) throws RemoteException;
	public Account accessAccount(String nom) throws RemoteException;

}
```

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.Hashtable;

public class BankImpl extends UnicastRemoteObject implements Bank {

    protected BankImpl() throws RemoteException {
            super();
    }

    Hashtable<String,Account> accountsList = new Hashtable<String,Account>();
    @Override
    public Account accessAccount(String name) throws RemoteException {
            System.out.println("account access requested by "+ name);
            return accountsList.get(name);
    }

    @Override
```

```
	public Account createAccount(String name) throws RemoteException {
		Account newCount = new AccountImpl(name);
		accountsList.put(name, newCount);
		System.out.println("new account created for "+ name);
		return newCount;
	}

}
```

```
import java.rmi.Remote;
import java.rmi.RemoteException;


public interface Account extends Remote {
	public String getOwner() throws RemoteException;
	public void transfer(double d) throws RemoteException;
	public String currentAmount() throws RemoteException;

}
```

```
public class AccountImpl extends UnicastRemoteObject implements Account {
	String holder=null;
	double solde=0.0;
	protected AccountImpl(String owner) throws RemoteException {
		super();
		holder=owner;
	}

	@Override
	public void transfer(double d) throws RemoteException {
		solde+=d;
	}



	@Override
	public String getOwner() throws RemoteException {
		return holder;
	}

	@Override
	public String currentAmount() {
		return "["+holder+" has  " + solde+" Euros ]";
	}

}
```

Q1) Expliquer pourquoi nous voyons cette exception dans le code d'un client

MaBanque ▸ src ▸ (default package) ▸ BankClient ▸ main(String[]) : void

```
1  import java.rmi.Naming;
2
3  public class BankClient {
4      public static void main(String[] args) {
5
6          Bank maBanque;
7          Account myCount, anotherCount;
8          try{
9              maBanque = (Bank) Naming.lookup("Bank");
10             System.out.println("I have access to the e-bank :)= ");
11             myCount = maBanque.createAccount("Mireille");
12             myCount.transfer(1200);
13             System.out.println("The amount of available funds of "+ myCount.getOwner() + " is "+ myCount.currer
14             myCount.transfer(-100);
15             System.out.println("The amount of available funds now is "+ myCount.currentAmount());
16             anotherCount=maBanque.accessAccount("toto");
17             System.out.println("The amount of available funds is "+ anotherCount.currentAmount() );
18         }
19         catch (Exception e){
20             e.printStackTrace();
```

Javadoc Declaration Console

<terminated> BankClient [Java Application] C:\Program Files\Java\jdk-11.0.15.1\bin\javaw.exe (6 déc. 2022 à 14:04:24 – 14:04:24) [pid: 19264]

```
I have access to the e-bank :)=
The amount of available funds of Mireille is [Mireille has  1200.0 Euros ]
The amount of available funds now is [Mireille has  1100.0 Euros ]
java.lang.NullPointerException
        at BankClient.main(BankClient.java:17)
```

Q2) Expliquer ce qu'il se produirait si deux clients de nom identique créent un compte.