

1- Print-screens Calculator

The screenshot shows an IDE with a project structure on the left and a terminal window at the bottom. The project structure includes a package `Applications.distribues.devoirs` with sub-packages `Implementation_d'un_service_distribue_Java_RMI` and `Calculator`. The `Calculator` package contains `Calculator.class`, `CalculatorClient.class`, `CalculatorServer.class`, `CalculatorImpl.class`, `CalculatorImpl.java`, `CalculatorServer.class`, `CalculatorServer.java`, and `SimpleBankingSystem`. The terminal window shows the following output:

```
PS C:\Users\marie\OneDrive\Desktop\Atos> git clone git@github.com:mariecloudsma/Applications.distribues.devoirs.git
Cloning into 'Applications.distribues.devoirs'...
remote: transferring objects: 76%, done.
remote: counting objects: 100% (76/76), done.
remote: compressing objects: 100% (51/51), done.
remote: Total 76 (delta 9), reused 63 (delta 6), pack-reused 0 (from 0)
Receiving objects: 100% (76/76), 1.54 MiB | 108.00 KiB/s, done.
Resolving deltas: 100% (0/0), done.
PS C:\Users\marie\OneDrive\Desktop\Atos> cd .\Applications.distribues.devoirs\
PS C:\Users\marie\OneDrive\Desktop\Atos> cd .\Applications.distribues.devoirs> cd .\Implementation_d'un_service_distribue_Java_RMI\
PS C:\Users\marie\OneDrive\Desktop\Atos> cd .\Applications.distribues.devoirs> cd .\Implementation_d'un_service_distribue_Java_RMI> cd .\Calculator>
PS C:\Users\marie\OneDrive\Desktop\Atos> cd .\Applications.distribues.devoirs> cd .\Implementation_d'un_service_distribue_Java_RMI> cd .\Calculator> javac *.java
PS C:\Users\marie\OneDrive\Desktop\Atos> cd .\Applications.distribues.devoirs> cd .\Implementation_d'un_service_distribue_Java_RMI> cd .\Calculator>
```

The screenshot shows an IDE with a project structure on the left and a terminal window at the bottom. The project structure is the same as the previous screenshot. The terminal window shows the following output:

```
PS C:\Users\marie\OneDrive\Desktop\Atos> cd .\Applications.distribues.devoirs> cd .\Implementation_d'un_service_distribue_Java_RMI> cd .\Calculator> rmiregistry
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release
[]
```

The screenshot shows an IDE with a project structure on the left and a terminal window at the bottom. The project structure is the same as the previous screenshots. The terminal window shows the following output:

```
PS C:\Users\marie\OneDrive\Desktop\Atos> cd .\Applications.distribues.devoirs> cd .\Implementation_d'un_service_distribue_Java_RMI> cd .\Calculator>
PS C:\Users\marie\OneDrive\Desktop\Atos> cd .\Applications.distribues.devoirs> cd .\Implementation_d'un_service_distribue_Java_RMI> cd .\Calculator> java CalculatorServer
```

The screenshot shows an IDE with a project structure on the left and a terminal window at the bottom. The project structure is the same as the previous screenshots. The terminal window shows the following output:

```
PS C:\Users\marie\OneDrive\Desktop\Atos> cd .\Applications.distribues.devoirs> cd .\Implementation_d'un_service_distribue_Java_RMI> cd .\Calculator>
PS C:\Users\marie\OneDrive\Desktop\Atos> cd .\Applications.distribues.devoirs> cd .\Implementation_d'un_service_distribue_Java_RMI> cd .\Calculator> java CalculatorClient
```

Start a phone book (rmiregistry) so the client can find the server.

Start the server and register the calculator in the phone book.

Start the client so it can look up the calculator and use it.

2- Print-screens SimpleBankingSystems et exercice III

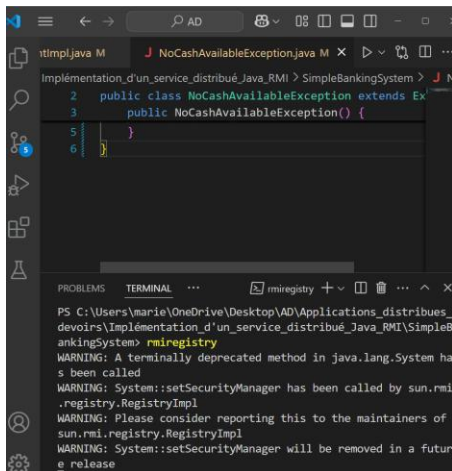
Partie III- Mise en œuvre d'un Simple Banking System

1. Au niveau de cet exercice aussi vous avez les classes relatives à la mise en place d'un Simple Banking System avec Java RMI. Téléchargez le folder correspondant sur moodle, **faites les corrections nécessaires au niveau du code**, compilez et exécutez.
2. Essayez de comprendre l'implémentation qui vous est proposée.
3. Est-ce que tous les objets Remote sont des objets enregistrés au niveau du RMI registry ?
4. Est-ce que tous les objets Remote sont dérivés de UnicastRemoteObject ?

1.

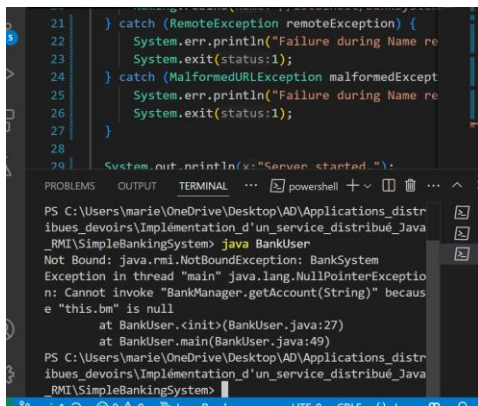
private Hashtable<String, Account> accounts;

private Hashtable<String, Client> clients;



```
ntimpl.java M NoCashAvailableException.java M X
Implémentation d'un service distribué Java RMI > SimpleBankingSystem > N
2 public class NoCashAvailableException extends Ex
3 public NoCashAvailableException() {
5 }
6 }

PROBLEMS TERMINAL ... rmi registry + v
PS C:\Users\marie\OneDrive\Desktop\AD\Applications_distribues_devoirs\Implémentation d'un service distribué Java RMI\SimpleBankingSystem> rmi registry
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release
```



```
21 } catch (RemoteException remoteException) {
22     System.err.println("Failure during Name re
23     System.exit(status:1);
24 } catch (MalformedURLException malformedExcept
25     System.err.println("Failure during Name re
26     System.exit(status:1);
27 }
28
29 System.out.println(y: "Server started.");

PROBLEMS OUTPUT TERMINAL ... powershell + v
PS C:\Users\marie\OneDrive\Desktop\AD\Applications_distribues_devoirs\Implémentation d'un service distribué Java RMI\SimpleBankingSystem> java BankUser
Not Bound: java.rmi.NotBoundException: BankSystem
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "BankManager.getAccount(String)" because "this.bm" is null
    at BankUser.<init>(BankUser.java:27)
    at BankUser.main(BankUser.java:49)
PS C:\Users\marie\OneDrive\Desktop\AD\Applications_distribues_devoirs\Implémentation d'un service distribué Java RMI\SimpleBankingSystem>
```

Its not working and i cant fix it

2. Compréhension de l'implémentation

Le système est composé des classes suivantes :

- Account et AccountImpl : Représentent un compte bancaire avec des méthodes pour obtenir le solde, retirer de l'argent, etc.
- BankManager et BankManagerImpl : Gèrent les comptes et les clients. Ils permettent de récupérer un compte ou un client par son identifiant.
- Client et ClientImpl : Représentent un client avec des informations telles que son nom et le gestionnaire de banque associé.
- BankSystemServer : Le serveur RMI qui expose le BankManager aux clients distants.
- BankUser : Un client qui se connecte au serveur RMI pour interagir avec le système bancaire.

3. Est-ce que tous les objets Remote sont des objets enregistrés au niveau du RMI registry ?

Non, tous les objets distants (implémentant l'interface Remote) ne sont pas nécessairement enregistrés dans le registre RMI. Seuls les objets qui doivent être accessibles directement par les clients via un nom (comme BankManager) sont enregistrés dans le registre RMI.

Par exemple, dans ce système, seul BankManager est enregistré dans le registre RMI avec le nom "//localhost/BankSystem". Les objets Account et Client sont accessibles via des méthodes de BankManager, mais ils ne sont pas enregistrés directement dans le registre RMI.

4. Est-ce que tous les objets Remote sont dérivés de UnicastRemoteObject ?

Non, tous les objets distants n'ont pas besoin d'être dérivés de UnicastRemoteObject.

Cependant, pour qu'un objet soit accessible via RMI, il doit être exporté. Cela peut être fait de deux manières :

- En héritant de UnicastRemoteObject : Cela exporte automatiquement l'objet lors de sa création.
- En utilisant UnicastRemoteObject.exportObject() : Cela permet d'exporter manuellement un objet distant sans hériter de UnicastRemoteObject.

Dans le code fourni, les objets distants (AccountImpl, ClientImpl, BankManagerImpl) sont exportés manuellement via UnicastRemoteObject.exportObject(), donc ils n'héritent pas de UnicastRemoteObject.

3.

Comprendre ce que le programme fait (côté client et côté serveur)

- **Côté Serveur (RMIServer) :**

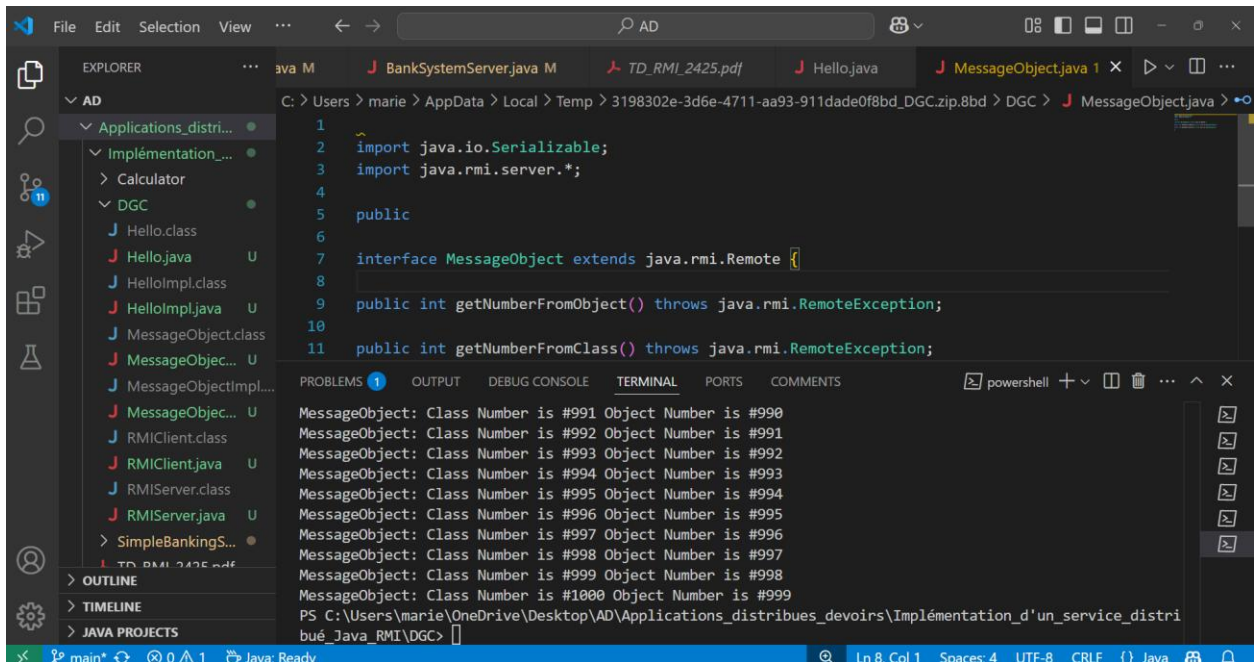
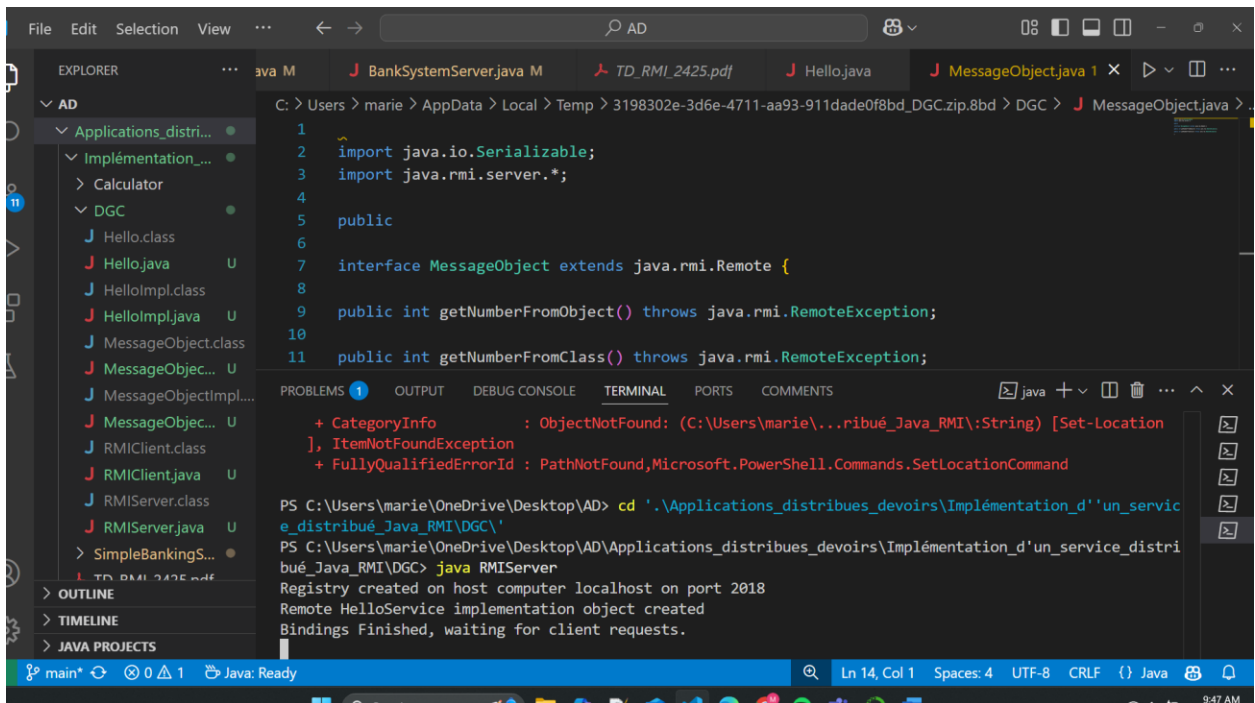
- Le serveur crée un registre RMI sur le port 2018.
- Il instancie un objet HelloImpl qui implémente l'interface Hello.
- Cet objet est ensuite enregistré dans le registre RMI sous le nom "Hello".
- Le serveur attend les requêtes des clients pour fournir des services RMI.

- **Côté Client (RMIClient) :**

- Le client se connecte au serveur RMI en utilisant l'URL "rmi://localhost:2018/Hello".
- Il appelle la méthode sayHello() pour obtenir un message de bienvenue.
- Ensuite, il appelle la méthode getMessageObject() 1000 fois pour créer des objets MessageObject.
- Le client affiche les numéros de classe et d'objet pour chaque MessageObject créé.
- Les objets MessageObject sont ensuite déréférencés (mo = null), ce qui déclenche le mécanisme de garbage collection.

The screenshot shows an IDE with the following components:

- EXPLORER:** A file tree on the left showing a project structure with folders like 'Applications_distri...' and 'DGC', and files like 'Hello.class', 'HelloImpl.class', 'MessageObject.class', etc.
- EDITOR:** The main window displays the code for 'MessageObject.java'. It includes imports for 'java.io.Serializable' and 'java.rmi.server.*', and defines an interface 'MessageObject' with two methods: 'getNumberFromObject()' and 'getNumberFromClass()'.
- PROBLEMS:** A panel on the right shows warnings. One warning is visible: 'MessageObjectImpl.java:31: warning: [removal] finalize() in Object has been deprecated and marked for removal'. Below this, there are four warnings related to deprecated methods in 'java.lang.System'.
- TERMINAL:** A terminal window at the bottom shows the command 'PS C:\Users\marie\OneDrive\Desktop\AD\Applications_distribues_devoirs\Implémentation_d'un_service_distribué_Java_RMI\DGC> rmiregistry' and the output of the command.



- The server prints messages when it creates MessageObject instances.
- The clients print the details of the MessageObject instances they receive.
- When a MessageObject is no longer referenced by a client, the unreferenced() method is called, and a message is printed.

- When the local garbage collector is about to destroy a MessageObject, the finalize() method is called, and a message is printed.
- The local garbage collector does not destroy objects immediately but waits until near the end of the run.
- After the clients terminate, the server waits for the lease term (10 minutes) before cleaning up the unreferenced objects.

```

U      MessageObject: Class Number is #1000 Object Number is #999
●      PS C:\Users\marie\OneDrive\Desktop\AD\Applications_distribues_
      bué_Java_RMI\DGC> java -Xmx1m RMIServer
      Error occurred during initialization of VM
      Too small maximum heap
      PS C:\Users\marie\OneDrive\Desktop\AD\Applications_distribues_
      bué_Java_RMI\DGC> 

```

```

U      Error occurred during initialization of VM
●      Too small maximum heap
      PS C:\Users\marie\OneDrive\Desktop\AD\Applications_distribues_devoirs\Imp
      bué_Java_RMI\DGC> java -Xmx10m RMIServer
      Registry created on host computer localhost on port 2018
      Remote HelloService implementation object created
      Bindings Finished, waiting for client requests.
      

```

Partie I- Personnalisation de la couche transport

- 1. Vous devez commencer par vous assurer que le service RMI sur lequel vous allez faire la personnalisation s'exécute bien dans un seul folder. Un folder contenant le service et les classes nécessaires vous est fourni sur moodle. Compilez les classes qui vous sont fournies, démarrez trois terminaux puis testez votre service, en faisant attention à la configuration nécessaire à la ligne de commande. Expliquez.**
- 2. Vous devez par la suite lire et comprendre comment s'est faite la personnalisation.**
- 3. Quel est le design pattern utilisé pour l'implémentation des XOR Input et OutputStreams?**
- 4. Au niveau de quelle instruction s'est fait le choix de nos Sockets et non des Sockets TCP par défaut?**
- 5. Vous devez modifier la classe HelloImpl pour que l'on ait une classe serveur à part.**

It did not work i tried