Exercice d'application en .NET Remoting

Nous disposons d'un service qui permet d'offrir des opérations permettant la gestion de son compte courant. Voici le code des méthodes offertes par ce service :

```
void debiter(double montant) {

      }

void crediter(double montant) {

      }

 double lireSolde() {

      }
```

1. On souhaite rendre chacune de ces méthodes accessibles à distance de manière à ce qu'elles définissent l'interface entre le client et le serveur. Ecrire cette interface.

2. Déduire la classe qui implémente l'interface.

3. Compléter le fichier Serveur.cs pour permettre l'enregistrement du service *Compte* auprès du CLR sous le nom de « Transaction ». Si l'objet *Compte* gère le même compte bancaire, à votre avis, faudra-t-il gérer l'objet en mode *SingleCall* ou en mode *Singleton* ?

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using nameSpaceCompte;


namespace nameSpaceServeur {
   // Implementation de la classe des objets serveurs.
   class Serveur {
      static void Main() {

         // A compléter

         // 3) Maintient du processus courant.
         Console.WriteLine("Pressez une touche pour stopper le serveur.");
         Console.Read();
      }
   }
}
```

4. Lancer le serveur.

5. Compléter le programme du client qui doit être lancé à partir d'une nouvelle instance de Visual C# 2008.

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using nameSpaceInterface;

namespace namespaceClient
{
    class Client
    {
        static void Main(string[] Argument)
        {

            // A compléter

            int op = int.Parse(Argument[0]);
            double m = double.Parse(Argument[1]);

            // Appel d'une methode sur l'objet distant.
            try
            {
                switch (op)
                {
                    case 1: obj.debiter(m);
                        break;
                    case 2: obj.crediter(m);
                        break;
                }

                Console.WriteLine("solde courant : {0}" + obj.lireSolde());
            }
            catch (RemotingException e) {
                  Console.WriteLine("erreur : {0}" + e.GetBaseException());
            }
            Console.Read();
            }
    }
}
```

Remarques :
- Pour tester ces programmes, télécharger gratuitement *Visual C# 2008 Express Edition* et installez-le, à partir du lien suivant :
http://msdn.microsoft.com/fr-fr/express/aa975050.aspx
Pour exécuter correctement le client et le serveur, rajouter dans l'onglet *References* la librairie *System.RunTime.Remoting*.
- Les arguments du programme du client doivent être saisis dans un champ accessible à partir du menu Projet/Proproriétés/Deboguer. Une valeur réelle doit être saisie avec une virgule (par exemple 13,5) et non avec un point.

Solution

1. La structure de la classe interface `CompteInterface.cs` est la suivante.

```
namespace nameSpaceInterface
{

    public interface CompteInterface
    {

        void debiter(double montant);

        void crediter(double montant);

        double lireSolde();

    }
}
```

2. La structure de la classe d'implémentation `Compte.cs` est la suivante.

```
using System;
using nameSpaceInterface;

namespace nameSpaceCompte {
public class Compte : MarshalByRefObject, CompteInterface {
      private double solde=15;
      public Compte() {
         Console.WriteLine("Compte Constructeur");
      }
      public void debiter(double montant) {
         Console.WriteLine("debiter( {0} )", montant);

         solde=solde-montant;
      }
      public void crediter(double montant) {
         Console.WriteLine("crediter( {0} )", montant);

         solde=solde+montant;
      }
       public double lireSolde() {
         Console.WriteLine("LireSolde : {0}", solde);
         return solde;
      }

   }
}
```

3. La classe Serveur.cs est la suivante :

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using nameSpaceCompte;


namespace nameSpaceServeur {
   // Implementation de la classe des objets serveurs.
```

```
   class Serveur {
      static void Main() {
       // 1)Creation d'un canal Tcp sur le port 65000
       //enregistre ce canal dans le domaine d'application courant.
         TcpChannel canal = new TcpChannel(65000);
         ChannelServices.RegisterChannel(canal, false);

       // 2) Ce domaine d'application presente un objet de type
       //CompteInterface associe au point terminal 'Transaction'.
         RemotingConfiguration.RegisterWellKnownServiceType(
            typeof(nameSpaceCompte.Compte),
            "Transaction",
            WellKnownObjectMode.Singleton);

         // 3) Maintient du processus courant.
         Console.WriteLine("Pressez une touche pour stopper le serveur.");
         Console.Read();
      }
   }
}
```

4. Voici un exemple de programme client qui doit disposer dans son contexte du fichier CompteInterface.cs :

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using nameSpaceInterface;


namespace namespaceClient
{


    class Client
    {
        static void Main(string[] Argument)
        {
            // Creer un canal TCP puis enregistre
            // ce canal dans le domaine d'application courant.
            // (la valeur 0 pour le numero de port du client signifie
            // que ce numero de port est choisi automatiquement
            // par le CLR).
            TcpChannel canal = new TcpChannel(0);
            ChannelServices.RegisterChannel(canal, false);

            // Obtient un proxy transparent sur l'objet distant a partir de
            // son URI, puis transtype le proxy distant en CompteInterface.
            MarshalByRefObject objRef =
(MarshalByRefObject)RemotingServices.Connect(typeof(nameSpaceInterface.Comp
teInterface),
                "Tcp://localhost:65000/Transaction");
            CompteInterface obj = objRef as CompteInterface;

            int op = int.Parse(Argument[0]);
            double m = double.Parse(Argument[1]);

            // Appel d'une methode sur l'objet distant.
```

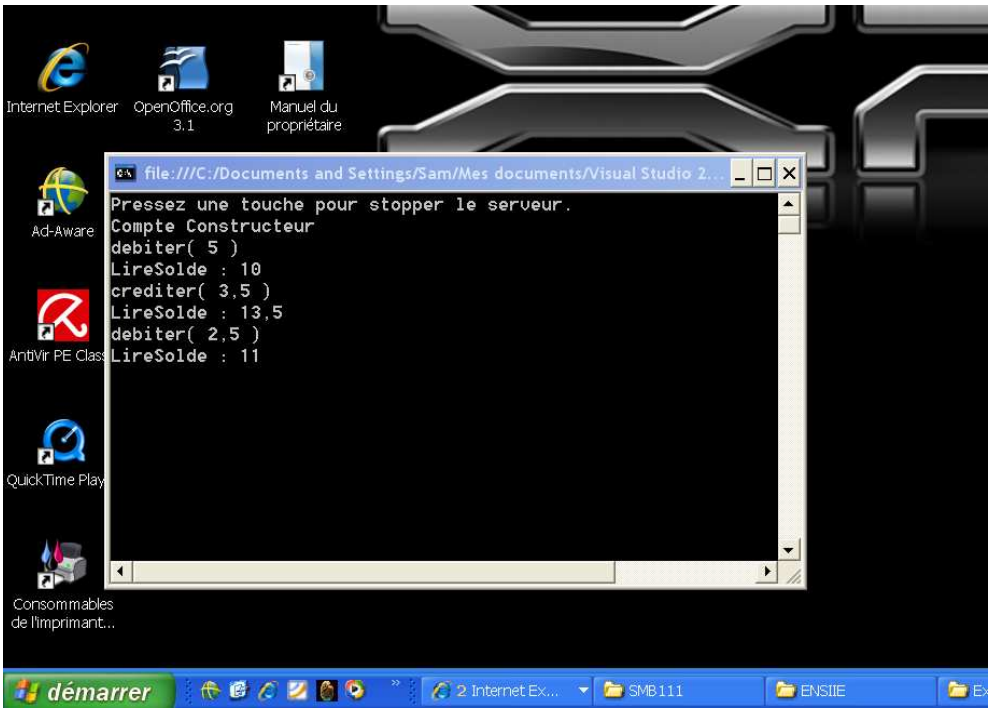
```
            try
            {
                switch (op)
                {
                    case 1: obj.debiter(m);
                        break;
                    case 2: obj.crediter(m);
                        break;
                }

                Console.WriteLine("solde courant : {0}" + obj.lireSolde());
            }
            catch (RemotingException e) { Console.WriteLine("erreur : {0}"
+ e.GetBaseException()); }

            Console.Read();


           }
    }
}
```

Interface côté client : notons que les noms du namespace comme celui de l'interface doivent être identiques côté client et côté serveur.

```
namespace nameSpaceInterface
{

    public interface CompteInterface
    {

        void debiter(double montant);

        void crediter(double montant);

        double lireSolde();

    }
}
```

5. Exemple d'exécution

Serveur



Client