Examen terminal
Architectures n-tiers : Intergiciels à objets et services web

Tous documents autorisés.
Le barème est donné à titre indicatif.

1 RMI et Corba (8 points)

On reprend une version très simplifiée du TP RMI, dont le code vous est donné aux listings 1, 2, 3, 4, 5, et 6

Listing 1 – CabinetVeterinaire.java

```
1  package miniVetoExam2012;
2  import java.rmi.Remote;
3  import java.rmi.RemoteException;
4  import java.util.ArrayList;
5  public interface CabinetVeterinaire extends Remote{
6      ArrayList<Dossier> getDossierByOwnerName(String ownerName) throws RemoteException;
7  }
```

Listing 2 – CabinetVeterinaireImpl.java

```
1  package miniVetoExam2012;
2
3  import java.rmi.RemoteException;
4  import java.rmi.server.UnicastRemoteObject;
5  import java.util.ArrayList;
6
7  public class CabinetVeterinaireImpl extends UnicastRemoteObject  implements CabinetVeterinaire
       {
8
9      private ArrayList<Dossier> dossiers=new ArrayList<Dossier>();
10
11     public CabinetVeterinaireImpl() throws RemoteException{
12
13     }
14
15     public void addDossier(DossierImpl d){
16         dossiers.add(d);
17     }
18
19     public ArrayList<Dossier> getDossierByOwnerName(String ownerName) throws RemoteException{
20         ArrayList<Dossier> result=new ArrayList<Dossier>();
21         for (Dossier d:dossiers){
22             if (d.getOwnerName().equals(ownerName)){
23                 result.add(d);
24             }
25         }
26         return result;
27         }
28
29 }
```

Listing 3 – Dossier.java

```
1  package miniVetoExam2012;
2  import java.rmi.Remote;
3  import java.rmi.RemoteException;
4  public interface Dossier extends Remote{
5      public String getOwnerName() throws RemoteException;
6      public String getAnimalName() throws RemoteException;
7  }
```

Listing 4 – DossierImpl.java

```
1  package miniVetoExam2012;
2
3  import java.rmi.RemoteException;
4  import java.rmi.server.UnicastRemoteObject;
5
6  public class DossierImpl extends UnicastRemoteObject  implements Dossier {
7      private String ownerName;
8      private String animalName;
9
10     public DossierImpl(String ownerName, String animalName)
11             throws RemoteException {
12         super();
13         this.ownerName = ownerName;
14         this.animalName = animalName;
15     }
16
17     public String getOwnerName() throws RemoteException{
18         return ownerName;
19     }
20     public String getAnimalName() throws RemoteException{
21         return animalName;
22     }
23 }
```

Listing 5 – Client.java

```
1  package miniVetoExam2012;
2
3  import java.rmi.registry.LocateRegistry;
4  import java.rmi.registry.Registry;
5  import java.util.ArrayList;
6
7  public class Client {
8      private Client() {}
9
10     public static void main(String[] args) {
11         try {
12             Registry registry = LocateRegistry.getRegistry(2000);
13             CabinetVeterinaire miniCab = (CabinetVeterinaire) registry.lookup("miniCab");
14             ArrayList<Dossier> dossiersTintin=miniCab.getDossierByOwnerName("Tintin");
15             for (Dossier d:dossiersTintin){
16                 System.out.println(d.getAnimalName());
17             }
18         } catch (Exception e) {
19             System.err.println("Client exception: " + e.toString());
20             e.printStackTrace();
21         }
22     }
23 }
```

Listing 6 – Server.java

```
1  package miniVetoExam2012;
2  import java.rmi.registry.Registry;
3  import java.rmi.registry.LocateRegistry;
4
5  public class Server {
6
7      public Server() {}
8
9
10     public static void main(String args[]) {
11
12         try {
13             CabinetVeterinaireImpl miniCab = new CabinetVeterinaireImpl();
14             DossierImpl milou=new DossierImpl("Tintin", "Milou");
15             DossierImpl idefix=new DossierImpl("Obelix", "Idefix");
16             miniCab.addDossier(milou);
17             miniCab.addDossier(idefix);
18             Registry registry = LocateRegistry.getRegistry(2000);
19             if (registry==null){
20                 System.out.println("RmiRegistry not found");
```

```
21              }else{
22                  registry.bind("miniCab", miniCab);
23                  System.out.println("Server ready");
24              }
25          } catch (Exception e) {
26              System.out.println("Server exception: " + e.toString());
27              e.printStackTrace();
28          }
29      }
30  }
```

Question 1. Ce code permet-il de gérer un seul cabinet vétérinaire ou plusieurs ? Justifier.

Question 2. La méthode `getDossierByOwnerName` retourne une ArrayList de DossierImpl. Lors de l'appel de cette méthode dans le main du client, de quel type dynamique est le résultat obtenu ? Expliquer pourquoi le type reçu n'est pas le type envoyé et à quel endroit a pu avoir lieu le "changement de type".

Question 3. Si on remplace la ligne 9 du listing 2 par la ligne :

```
1 private ArrayList<DossierImpl> dossiers = new ArrayList<DossierImpl>();
```

l'application continue-t-elle de compiler normalement ? De s'exécuter normalement ?

Question 4. On souhaite réécrire cette application avec Corba au lieu de RMI.

a- Donner l'idl nécessaire.

b- Donner l'ensemble du code Java nécessaire. S'il y a des classes très peu modifiées, vous pouvez les corriger directement sur l'énoncé (ne pas oublier alors de rendre l'énoncé, avec votre numéro d'étudiant dessus), ou juste expliquer sur votre copie les modifications à apporter. Vous supposerez disposer de tout ce qui est généré par la commande `idlj -fall` appelée sur votre idl.

2 Services Web et WCF (5,5 points)

Soit l'extrait de fichier WSDL donné au listing 7.

Listing 7 – extrait d'un fichier WSDL

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://
      microsoft.com/wsdl/mime/textMatching/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/
      encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://services.
      aonaware.com/webservices/" xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http:
      //schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
      targetNamespace="http://services.aonaware.com/webservices/" xmlns:wsdl="http://schemas.
      xmlsoap.org/wsdl/">
3   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Word Dictionary Web
        Service</wsdl:documentation>
4   <wsdl:types>
5     <s:schema elementFormDefault="qualified" targetNamespace="http://services.aonaware.com/
          webservices/">
6 (...)
7       <s:element name="DictionaryList">
8         <s:complexType />
9       </s:element>
10      <s:element name="DictionaryListResponse">
11        <s:complexType>
12          <s:sequence>
13            <s:element minOccurs="0" maxOccurs="1" name="DictionaryListResult" type="
                  tns:ArrayOfDictionary" />
14          </s:sequence>
15        </s:complexType>
16      </s:element>
17      <s:complexType name="ArrayOfDictionary">
18        <s:sequence>
19          <s:element minOccurs="0" maxOccurs="unbounded" name="Dictionary" nillable="true"
                type="tns:Dictionary" />
20        </s:sequence>
21      </s:complexType>
```

```
22      <s:complexType name="Dictionary">
23        <s:sequence>
24          <s:element minOccurs="0" maxOccurs="1" name="Id" type="s:string" />
25          <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string" />
26        </s:sequence>
27      </s:complexType>
28 (...)
29      <s:complexType name="WordDefinition">
30        <s:sequence>
31          <s:element minOccurs="0" maxOccurs="1" name="Word" type="s:string" />
32          <s:element minOccurs="0" maxOccurs="1" name="Definitions" type="
                tns:ArrayOfDefinition" />
33        </s:sequence>
34      </s:complexType>
35      <s:complexType name="ArrayOfDefinition">
36        <s:sequence>
37          <s:element minOccurs="0" maxOccurs="unbounded" name="Definition" nillable="true"
                type="tns:Definition" />
38        </s:sequence>
39      </s:complexType>
40      <s:complexType name="Definition">
41        <s:sequence>
42          <s:element minOccurs="0" maxOccurs="1" name="Word" type="s:string" />
43          <s:element minOccurs="0" maxOccurs="1" name="Dictionary" type="tns:Dictionary" />
44          <s:element minOccurs="0" maxOccurs="1" name="WordDefinition" type="s:string" />
45        </s:sequence>
46      </s:complexType>
47      <s:element name="DefineInDict">
48        <s:complexType>
49          <s:sequence>
50            <s:element minOccurs="0" maxOccurs="1" name="dictId" type="s:string" />
51            <s:element minOccurs="0" maxOccurs="1" name="word" type="s:string" />
52          </s:sequence>
53        </s:complexType>
54      </s:element>
55      <s:element name="DefineInDictResponse">
56        <s:complexType>
57          <s:sequence>
58            <s:element minOccurs="0" maxOccurs="1" name="DefineInDictResult" type="
                  tns:WordDefinition" />
59          </s:sequence>
60        </s:complexType>
61      </s:element>
62 (..)
63      <s:element name="string" nillable="true" type="s:string" />
64      <s:element name="ArrayOfDictionary" nillable="true" type="tns:ArrayOfDictionary" />
65      <s:element name="WordDefinition" nillable="true" type="tns:WordDefinition" />
66      <s:element name="ArrayOfStrategy" nillable="true" type="tns:ArrayOfStrategy" />
67      <s:element name="ArrayOfDictionaryWord" nillable="true" type="tns:ArrayOfDictionaryWord"
            />
68    </s:schema>
69  </wsdl:types>
70   (...)
71  <wsdl:message name="DictionaryListSoapIn">
72    <wsdl:part name="parameters" element="tns:DictionaryList" />
73  </wsdl:message>
74  <wsdl:message name="DictionaryListSoapOut">
75    <wsdl:part name="parameters" element="tns:DictionaryListResponse" />
76  </wsdl:message>
77   (...)
78  <wsdl:message name="DefineInDictSoapIn">
79    <wsdl:part name="parameters" element="tns:DefineInDict" />
80  </wsdl:message>
81  <wsdl:message name="DefineInDictSoapOut">
82    <wsdl:part name="parameters" element="tns:DefineInDictResponse" />
83  </wsdl:message>
84  (...)
85  <wsdl:portType name="DictServiceSoap">
86    (...)
87    <wsdl:operation name="DictionaryList">
88      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Returns a list of
            available dictionaries</wsdl:documentation>
89      <wsdl:input message="tns:DictionaryListSoapIn" />
90      <wsdl:output message="tns:DictionaryListSoapOut" />
91    </wsdl:operation>
92 (...)
93    <wsdl:operation name="DefineInDict">
```

```
94        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Define given word,
              returning definitions from specified dictionary</wsdl:documentation>
95        <wsdl:input message="tns:DefineInDictSoapIn" />
96        <wsdl:output message="tns:DefineInDictSoapOut" />
97      </wsdl:operation>
98  (...)
99    </wsdl:portType>
100 (...)
101     <wsdl:operation name="DictionaryList">
102       <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Returns a list of
              available dictionaries</wsdl:documentation>
103       <wsdl:input message="tns:DictionaryListHttpGetIn" />
104       <wsdl:output message="tns:DictionaryListHttpGetOut" />
105     </wsdl:operation>
106 (...)
107     <wsdl:operation name="DefineInDict">
108       <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Define given word,
              returning definitions from specified dictionary</wsdl:documentation>
109       <wsdl:input message="tns:DefineInDictHttpGetIn" />
110       <wsdl:output message="tns:DefineInDictHttpGetOut" />
111     </wsdl:operation>
112 (...)
113   </wsdl:portType>
114 (...)
115   <wsdl:binding name="DictServiceSoap" type="tns:DictServiceSoap">
116     <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
117     (...)
118     <wsdl:operation name="DictionaryList">
119       <soap:operation soapAction="http://services.aonaware.com/webservices/DictionaryList"
              style="document" />
120       <wsdl:input>
121         <soap:body use="literal" />
122       </wsdl:input>
123       <wsdl:output>
124         <soap:body use="literal" />
125       </wsdl:output>
126     </wsdl:operation>
127     (...)
128     <wsdl:operation name="DefineInDict">
129       <soap:operation soapAction="http://services.aonaware.com/webservices/DefineInDict" style
              ="document" />
130       <wsdl:input>
131         <soap:body use="literal" />
132       </wsdl:input>
133       <wsdl:output>
134         <soap:body use="literal" />
135       </wsdl:output>
136     </wsdl:operation>
137   </wsdl:binding>
138   <wsdl:binding name="DictServiceSoap12" type="tns:DictServiceSoap">
139     <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
140     (...)
141     <wsdl:operation name="DictionaryList">
142       <soap12:operation soapAction="http://services.aonaware.com/webservices/DictionaryList"
              style="document" />
143       <wsdl:input>
144         <soap12:body use="literal" />
145       </wsdl:input>
146       <wsdl:output>
147         <soap12:body use="literal" />
148       </wsdl:output>
149     </wsdl:operation>
150     (...)
151     <wsdl:operation name="DefineInDict">
152       <soap12:operation soapAction="http://services.aonaware.com/webservices/DefineInDict"
              style="document" />
153       <wsdl:input>
154         <soap12:body use="literal" />
155       </wsdl:input>
156       <wsdl:output>
157         <soap12:body use="literal" />
158       </wsdl:output>
159     </wsdl:operation>
160     (...)
161   </wsdl:binding>
162   (...)
163   <wsdl:service name="DictService">
```

```
164     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Word Dictionary Web
            Service</wsdl:documentation>
165     <wsdl:port name="DictServiceSoap" binding="tns:DictServiceSoap">
166       <soap:address location="http://services.aonaware.com/DictService/DictService.asmx" />
167     </wsdl:port>
168     <wsdl:port name="DictServiceSoap12" binding="tns:DictServiceSoap12">
169       <soap12:address location="http://services.aonaware.com/DictService/DictService.asmx" />
170     </wsdl:port>
171     (...)
172   </wsdl:service>
173 </wsdl:definitions>
```

Question 5. Donnez sous forme de signatures de méthodes les opérations décrites dans l'extrait de ce fichier.

Question 6. Le service reçoit ce message SOAP :

```
1 POST /DictService/DictService.asmx HTTP/1.1
2 Host: services.aonaware.com
3 Content-Type: application/soap+xml; charset=utf-8
4 Content-Length: length
5
6 <?xml version="1.0" encoding="utf-8"?>
7 <soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.
      w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
8   <soap12:Body>
9     <DefineInDict xmlns="http://services.aonaware.com/webservices/">
10      <dictId>vera</dictId>
11      <word>wsdl</word>
12    </DefineInDict>
13  </soap12:Body>
14 </soap12:Envelope>
```

Quel message SOAP renvoit-il sachant que `vera` est l'identifiant du dictionnaire de nom `Virtual Entity of Relevant Acronyms (Version 1.9, June 2002)` ? On ne donnera pas l'entête http mais juste le contenu de l'enveloppe.

Question 7. Ce service vous semble t-il pratique à utiliser ? Expliquer.

Question 8. On souhaite utiliser ce service dans une application développée en C# et faire un appel du genre : `monServiceDico.defineInDict(''vera'', ''wsdl'')`.

a- Comment procéder ? Soyez précis : expliquer quel(s) outil(s) utiliser, dans quel but, et la nature de ce qu'il(s) produi(sen)t. De quel type sera `monServiceDico` ?

b- Supposons que ce service ait été développé avec WCF. Comment aurait-on procédé ? Donner le genre de contrats de service et de données à développer, ainsi que les bindings à utiliser.

3 Questions courtes (3 points)

Question 9. Côté serveur, on dispose d'un objet o. Un client distant veut manipuler cet objet o. Le serveur peut lui transmettre par valeur ou par référence.

a- Qu'est-ce que cela signifie ? Quelles sont les conséquences de choisir l'une ou l'autre des solutions ?

b- Peut-on mettre en place les deux solutions avec des services WCF ? Si oui comment, sinon pourquoi ?

Question 10. .net remoting vous semble t-il plus proche de WCF ou de RMI ? Justifier.

4 WCF (3,5 points)

Soit le service WCF donné aux listings 8 et 9. Ce service est hébergé comme montré au listing 10 avec la configuration du listing 11. On a le programme client donné au listing 12 et la configuration du client donnée au listing 13.

Question 11. Combien d'instances de la classe d'implémentation du service seront créées si l'on lance 2 clients ? Comment le savez-vous ?

Question 12. On exécute le programme d'hébergement puis celui du client. Quels sont les affichages entraînés côté client par les lignes se terminant par `// a`, `// b`, `// c`, `// d`, et `// e` du listing 12 ?

Listing 8 – Contrat de service et contrat de données WCF

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.ServiceModel;
6  using System.Text;
7
8  namespace Foobar_service
9  {
10     [ServiceContract]
11     public interface IFoo
12     {
13         [OperationContract]
14         Bar GetBar();
15
16         [OperationContract]
17         void SetBar(Bar bar);
18     }
19
20     [DataContract]
21     public class Bar
22     {
23
24         public Bar(int bar)
25         {
26             this.bar = bar;
27         }
28
29         [DataMember]
30         public int bar
31         {
32             get;
33             set;
34         }
35     }
36 }
```

Listing 9 – Implémentation de service WCF

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.ServiceModel;
6  using System.Text;
7
8  namespace Foobar_service
9  {
10     public class Foo : IFoo
11     {
12         private Bar bar;
13
14         public Foo()
15         {
16             bar = new Bar(0);
17         }
18         public Bar GetBar()
19         {
20             return bar;
21         }
22         public void SetBar(Bar bar)
23         {
24             this.bar = bar;
25         }
26     }
27 }
```

Listing 10 – Hébergement de service WCF

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ServiceModel;
using Foobar_service;
namespace hebergementService
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ServiceHost host = new ServiceHost(typeof(Foobar_service.Foo)))
            {
                host.Open();
                Console.WriteLine("The service is ready.");
                Console.WriteLine("Press <Enter> to stop the service.");
                Console.ReadLine();

                // Close the ServiceHost.
                host.Close();
            }
        }
    }
}
```

Listing 11 – Configuration de service WCF

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service behaviorConfiguration="Foobar_service.Service1Behavior"
        name="Foobar_service.Foo">
        <endpoint address="" binding="wsHttpBinding" contract="Foobar_service.IFoo">
          <identity>
            <dns value="localhost" />
          </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8732/Design_Time_Addresses/Foobar_service/
                Service1/" />
          </baseAddresses>
        </host>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="Foobar_service.Service1Behavior">
          <serviceMetadata httpGetEnabled="True"/>
          <serviceDebug includeExceptionDetailInFaults="False" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

Listing 12 – Client WCF

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Foobar_service;
using System.ServiceModel;

namespace ClientFooBarServiceConsole
{
    class Program
    {
```

```
12          static void Main(string[] args)
13          {
14              ChannelFactory<IFoo> myChannelFactory = new ChannelFactory<IFoo>("configClient");
15              IFoo proxy = myChannelFactory.CreateChannel();
16              Bar bar =proxy.GetBar();
17              Console.WriteLine("bar.bar"+bar.bar); // a
18              bar.bar = 12;
19              Console.WriteLine("bar.bar" + bar.bar); // b
20              IFoo proxybis = myChannelFactory.CreateChannel();
21              Bar barbis = proxy.GetBar();
22              Console.WriteLine(barbis.bar); // c
23              proxy.SetBar(new Bar(3));
24              barbis = proxybis.GetBar();
25              bar = proxy.GetBar();
26              Console.WriteLine("bar.bar : " + bar.bar); // d
27              Console.WriteLine("barbis.bar : "+barbis.bar); // e
28              Console.ReadLine();
29          }
30      }
31  }
```

Listing 13 – Config client WCF

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <configuration>
3      <system.serviceModel>
4          <client>
5            <endpoint address="http://localhost:8732/Design_Time_Addresses/Foobar_service/
                 Service1/"
6                binding="wsHttpBinding" bindingConfiguration="" contract="Foobar_service.IFoo"
7                name="configClient">
8            </endpoint>
9          </client>
10     </system.serviceModel>
11 </configuration>
```