esprit
Ecole Supérieure Privée d'Ingénierie et de Technologies

EXAMEN

Semestre : 1■ 2□

Session : Principale■ Rattrapage□

Module : SOA (Architectures Orientée Service)
Enseignants: *BENLAZREG I, MAKNI M, MARZOUK E, ABED M, MESSOUD M.*
Classes : 4SIM1,2,3, 4INFOGL1,2,3,4 4ERP-BI1,2,3
Documents autorisés : □OUI ■NON Nombre de pages : 4
Date : 15/01/2014 Heure: 11H15 Durée : 1H30

NB : Répondre à toutes les questions sur la feuille d'examen

Exercice 1 : JAX-WS (6 pts)

L'objectif de cet exercice est de mettre en place un service web étendu pour inscription en ligne des étudiants dans une université.

Travail demandé : A partir du fichier WSDL fourni ci-dessous :

1. Identifier les types XSD des données utilisées par le service web.
2. Identifier les signatures des opérations fournies par ce service en précisant les annotations JAX-WS si elles existent.

```
<wsdl:definitions name="InscriptionEnLigne"
targetNamespace="http://inscription.universite.com/"
xmlns:tns="http://inscription.universite.com/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
<xs:schema elementFormDefault="unqualified"
targetNamespace="http://inscription.universite.com/" version="1.0"
xmlns:tns="http://inscription.universite.com/" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Inscription" type="tns:Inscription"/>
<xs:element name="InscriptionResponse" type="tns:InscriptionResponse"/>
<xs:element name="Liste_Etudiants_Inscrits" type="tns:Liste_Etudiants_Inscrits"/>
<xs:element name="Liste_Etudiants_InscritsResponse"
type="tns:Liste_Etudiants_InscritsResponse"/>
<xs:element name="Recherche_Etudiant" type="tns:Recherche_Etudiant"/>
<xs:element name="Recherche_EtudiantResponse" type="tns:Recherche_EtudiantResponse"/>
<xs:complexType name="Inscription">
    <xs:sequence>
      <xs:element minOccurs="0" name="etudiant" type="tns:etudiant"/>
    </xs:sequence>
  </xs:complexType>
<xs:complexType name="etudiant">
    <xs:sequence>
      <xs:element name="age" type="xs:int"/>
      <xs:element name="cin" type="xs:int"/>
      <xs:element name="id" type="xs:int"/>
      <xs:element minOccurs="0" name="nom" type="xs:string"/>
      <xs:element minOccurs="0" name="prenom" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
<xs:complexType name="InscriptionResponse">
```

```
    <xs:sequence>
      <xs:element name="Inscription_return" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
<xs:complexType name="Liste_Etudiants_Inscrits">
    <xs:sequence/>
  </xs:complexType>
<xs:complexType name="Liste_Etudiants_InscritsResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="EtudiantsInscrits_return"
type="tns:etudiant"/>
    </xs:sequence>
  </xs:complexType>
<xs:complexType name="Recherche_Etudiant">
    <xs:sequence>
      <xs:element name="id" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
<xs:complexType name="Recherche_EtudiantResponse">
    <xs:sequence>
      <xs:element minOccurs="0" name="Recherche_return" type="tns:etudiant"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
  </wsdl:types>
  <wsdl:message name="Liste_Etudiants_InscritsResponse">
    <wsdl:part element="tns:Liste_Etudiants_InscritsResponse" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="InscriptionResponse">
    <wsdl:part element="tns:InscriptionResponse" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="Recherche_EtudiantResponse">
    <wsdl:part element="tns:Recherche_EtudiantResponse" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="Inscription">
    <wsdl:part element="tns:Inscription" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="Recherche_Etudiant">
    <wsdl:part element="tns:Recherche_Etudiant" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="Liste_Etudiants_Inscrits">
    <wsdl:part element="tns:Liste_Etudiants_Inscrits" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="ServiceInscription">
    <wsdl:operation name="Inscription">
      <wsdl:input message="tns:Inscription" name="Inscription">
    </wsdl:input>
      <wsdl:output message="tns:InscriptionResponse" name="InscriptionResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="Liste_Etudiants_Inscrits">
      <wsdl:input message="tns:Liste_Etudiants_Inscrits" name="Liste_Etudiants_Inscrits">
```

```
    </wsdl:input>
      <wsdl:output message="tns:Liste_Etudiants_InscritsResponse"
name="Liste_Etudiants_InscritsResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="Recherche_Etudiant">
      <wsdl:input message="tns:Recherche_Etudiant" name="Recherche_Etudiant">
    </wsdl:input>
      <wsdl:output message="tns:Recherche_EtudiantResponse"
name="Recherche_EtudiantResponse">
    </wsdl:output>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="InscriptionEnLigneSoapBinding" type="tns:ServiceInscription">
  </wsdl:binding>
  <wsdl:service name="InscriptionEnLigne">
    <wsdl:port binding="tns:InscriptionEnLigneSoapBinding" name="ServiceInscriptionPort">
      <soap:address location="http://localhost:2000/examen2014/InscriptionEnLigne"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Exercice 2 : JAX-RS (6 pts)

L'objectif de cet exercice est de décrire un service web RESTful pour la gestion des comptes bancaires. Chaque compte a un identificateur (int), un propriétaire (String) et un solde (float). Le service web doit effectuer les opérations suivantes :

- Créer un compte.
- Créditer un compte.
- Afficher un compte.

Créer un compte

URL: http://localhost:8080/examen/Compte/Create

Requête : le corps de la requête contient une représentation XML du compte à créer.

```
<Compte>
  <ID>08997976</ID>
  <Proprietaire >Foulen Ben Foulen</Proprietaire >
  <Solde>567.600</Solde>
</Compte >
```

Réponse : retourne une chaîne de caractères pour indiquer le succès ou l'échec de la création.

```
Votre compte est ajouté avec succès.
```

Créditer un compte

URL: http://localhost:8080/examen/Compte/*[ID]*/addToSolde

Réponse : retourne une représentation XML du compte modifié ayant l'identifiant passé en paramètre.

```
<Compte>
  <ID>08997976</ID>
  <Proprietaire >Foulen Ben Foulen</Proprietaire >
  <Solde>788.908</Solde>
</Compte >
```

Afficher un compte

URL : http://localhost:8080/examen/ Compte/display?ID=[valeur]

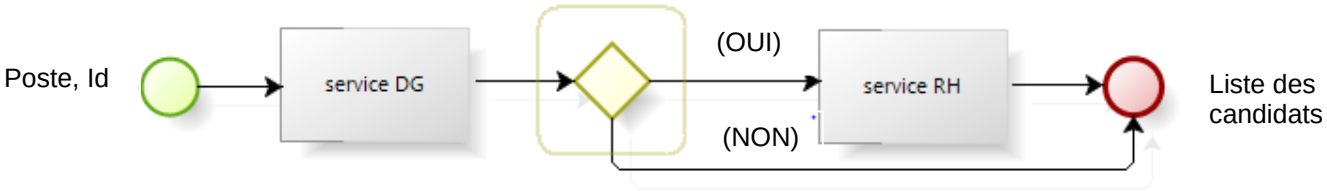
Réponse : retourne une représentation XML du compte ayant l'identifiant passé en paramètre.

```
<Compte>
  <ID>08997976</ID>
  <Proprietaire >Foulen Ben Foulen</Proprietaire >
  <Solde>788.908</Solde>
</Compte >
```

Travail demandé : Proposer un squelette du code avec les annotations appropriées.

Exercice3 : BPEL (8 pts)

Nous souhaitons modéliser avec BPEL un processus métier de recrutement pour l'ouverture d'un nouveau poste dans une entreprise :



1- Le responsable de recrutement soumet une demande pour un poste vacant en fournissant le **nom du poste** et son **id**.
2- La demande doit être traitée en premier lieu auprès du service de la Direction Générale en lui fournissant le nom et l'id du poste, et qui retourne une variable booléenne qui indique si la demande est approuvée ou pas.
3- Si la demande est approuvée, le nom du poste sera transmis à un deuxième service web des Ressources humaines pour chercher la liste des candidats.
4- La liste des candidats trouvés sera retournée au responsable de recrutement sous forme de chaine de caractère.
5- Si la demande n'est pas approuvée, un message de rejet sera transmis au responsable de recrutement.

Questions :

1. Définir les services partenaires, ainsi que les opérations à invoquer. Nous ne nous intéressons pas aux activités internes des différents partenaires.
2. Modéliser le processus de recrutement avec le langage BPEL en détaillant les différentes activités utilisées (les variables ainsi que les données qu'elles encapsulent).