Ministère de l'Enseignement Supérieur
Direction Générale des Etudes Technologiques
Institut Supérieur des Etudes Technologiques de Sfax

Classes	:	L3 DSI	**Date**	:	Janvier-2024
Matière	:	SOA	**Documents**	:	Non autorisés
Durée	:	1 heure 30 min	**Nbre pages**	:	6
Enseignants	:	Mohamed ZAYANI & Nissen MASMOUDI	**Barème**	:	6+14

**Il sera tenu compte de la présentation*

DEVOIR SE SYNTHESE

Nom : ..... **Prénom:**.....**Groupe :**.....

Le code donné en annexe représente le code d'une classe JAVA dans un projet Maven (**examen-rest-client**) de type «**maven-archetype-quickstart**».

Cette classe est considérée comme un client JAVA d'un service web REST écrit aussi dans un autre projet Maven (**examen-rest-serveur**) de type «**maven-archetype-webapp**».

PartieA : Côté Client REST (6points :2+1+1+2)

1. Dans le code donné en annexe, la classe **Etudiant** (définie dans le projet **examen-rest-serveur**) n'est pas reconnue. Pourquoi ? Citer les étapes à réaliser pour corriger cette erreur. (**2pt**)

.....

.....

.....

.....

.....

2. Le code donné en annexe utilise une API **Gson**. Indiquer l'utilité de cette API : (**1pt**)

.....

.....

3. L'exécution du code donné en annexe déclenche l'exception suivante :
com.google.gson.stream.MalformedJsonException: Use JsonReader.setLenient(true) to accept malformed JSON

Cette exception est déclenchée suite à l'exécution de l'instruction suivante :

```
JsonObject jo = new  JsonParser().parse(reponseGET).getAsJsonObject();
```

Expliquer l'origine de cette exception et proposer une correction du code (**indiquer seulement l'instruction à corriger**) : (**2pt**)

.....

.....

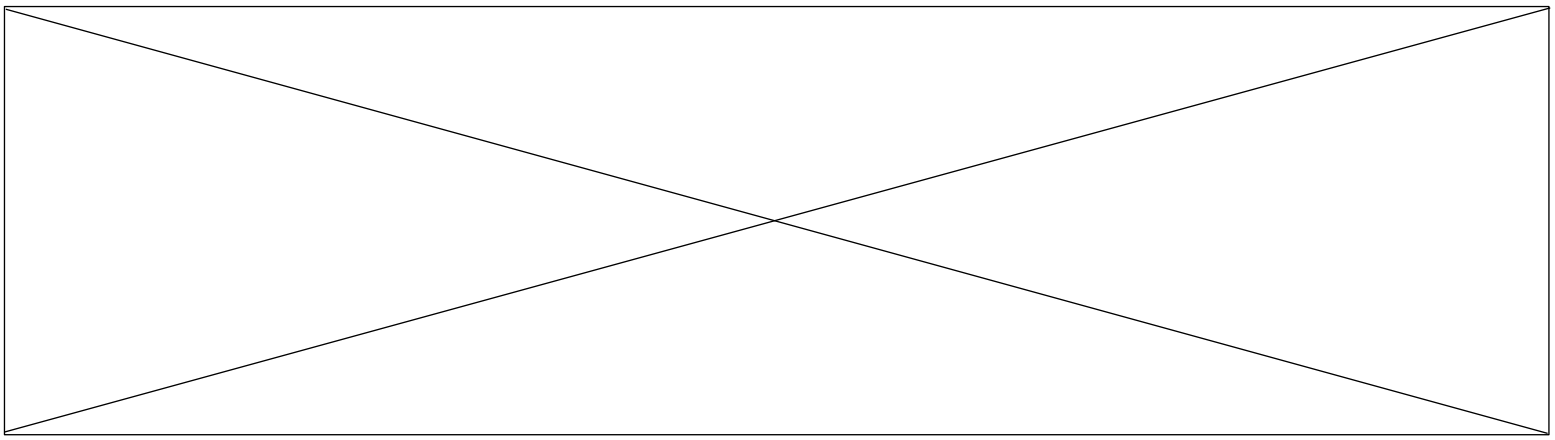
.....

.....

4. Le projet client (**examen-rest-client**) utilise deux dépendances Maven **JAXB** et **JACKSON**. Expliquer le rôle de chacune : (**2pt**)

JAXB :.....

JACKSON :.....



PartieB: Côté Serveur REST (14points :1+2+11)

En s'appuyant sur le code donné en annexe (code de la classe «**ClientMainApp**»), il est demandé de compléter quelques éléments du projet serveur (**examen-rest-serveur**).

NB : Toutes les classes java du «**projet examen-rest-serveur**» sont définies dans un même package.

5. Compléter le code de déclaration du plugin **tomcat7** dans le fichier **pom.xml** : (**1pt**)

```
<plugin>
  <groupId>org.apache.tomcat.maven</groupId>
  <artifactId>tomcat7-maven-plugin</artifactId>
  <version>2.2</version>
  <configuration>
    <path>……</path>
    <port>……</port>
  </configuration>
</plugin>
```

6. Compléter le code du fichier **web.xml** : (**2pt**)

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >
<web-app>
  <display-name>Application de gestion des étudiants</display-name>
  <servlet>
    <servlet-name>jerseyServlet</servlet-name>
    <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>com.sun.jersey.config.property.packages</param-name>
      <param-value>……</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>……</servlet-name>
    <url-pattern>……</url-pattern>
  </servlet-mapping>
</web-app>
```

7. **Donner le code d'une classe JAVA qui définit le service web REST** (consommé dans la classe «**ClientMainApp**» donnée en annexe) et qui implémente l'interface suivante en utilisant les annotations **JERSEY.** Cette classe permet de gérer des objets « **Etudiant** » à travers un objet «**HashMap**». (**11pt**)

```
package org.soa.examen.serveur;
public interface EtudiantServiceInterface {
      Reponse ajouterEtudiant(Etudiant e);
      Reponse supprimerEtudiant(int id);
      Reponse modifierEtudiant(Etudiant e);
      Etudiant getEtudiant(int id);
      Etudiant[] getAllEtudiants();
}
```

NB :

- La classe « **Reponse** » est définie par les attributs suivants :
  - etat : boolean
  - message : String
- La classe « **Etudiant** » est définie par les attributs suivants :
  - id : int
  - nom : String
  - prenom : String
- Ne pas donner les instructions « **import** ».

Bon Travail

Annexe

```
package org.soa.examen.client;

import com.google.gson.*;
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;
import org.soa.examen.serveur.Etudiant;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.UriBuilder;
import java.net.URI;

public class ClientMainApp
{
 public static void main( String[] args )
 {
  System.out.println("Application client....");
  // Objet de configuration
  ClientConfig config = new DefaultClientConfig();
  //objet client
  Client client = Client.create(config);
  //créer l'uri
  URI uri = UriBuilder.fromUri("http://localhost:7777/examen/soa/etudiants").build();
  //obtenir une resource correspondante à l'uri du service web
  WebResource service = client.resource(uri);

  System.out.println("* 1 ****************************************" );
  System.out.println( "Insérer deux étudiants...." );
  Etudiant etd1 =new Etudiant(1, "Ben Mohamed","Ali");
  String reponsePOST1= service.path("/")
                              .accept(MediaType.APPLICATION_JSON)
                              .post(String.class, etd1) ;
  System.out.println(reponsePOST1) ;

  Etudiant etd2 =new Etudiant(2, "Sallemi","Sami");
  String reponsePOST2= service.path("/")
                              .accept(MediaType.APPLICATION_JSON)
                              .post(String.class, etd2) ;
  System.out.println(reponsePOST2) ;

System.out.println( "* 2 ****************************************" );
 System.out.println( "Afficher la liste des étudiants...." );
 String reponseGET= service.path("/")
                           .accept(MediaType.APPLICATION_JSON)
                           .get(String.class) ;
 System.out.println(reponseGET) ;
 System.out.println( "* 3 ****************************************");
 System.out.println("Modifier le nom de l'étudiant ayant un id=1...");
 etd1.setNom("Ayyadi");
 String reponsePUT= service.path("/")
                           .accept(MediaType.APPLICATION_XML)
                           .put(String.class,etd1) ;
 System.out.println(reponsePUT) ;
 System.out.println( "* 4 ****************************************");
 System.out.println( "Supprimer l'étudiant ayant un id=2...." );
 String reponseDELETE= service.path("/2")
                              .accept(MediaType.APPLICATION_XML)
                              .delete(String.class) ;
 System.out.println(reponseDELETE) ;
```

```
System.out.println( "* 5 ****************************************");
// Récupérer des objets de type "Etudiant" en utilisant l'API gson de Google
Gson gson = new GsonBuilder().create();
reponseGET= service.path("/")
                   .accept(MediaType.APPLICATION_XML)
                   .get(String.class) ;
JsonObject jo = new JsonParser().parse(reponseGET).getAsJsonObject();
if (!reponseGET.equals("null")) // s'il existe au moins un objet "Etudiant"
  {
   if (jo.get("etudiant").isJsonArray())  // en cas de plusieurs etudiants
    {
     JsonArray jsonArray = jo.getAsJsonArray("etudiant");
     Etudiant[] listeEtudiants=gson.fromJson(jsonArray,Etudiant[].class);
     System.out.println("Liste des étudinats (en utilisant l'API Gson)....");
     for (Etudiant e: listeEtudiants)
        {
           System.out.println(e);
        }
     }else
     {  // en cas d'un seul objet "Etudiant"
       JsonObject jsonObject = jo.getAsJsonObject("etudiant");
       Etudiant etd = gson.fromJson(jsonObject, Etudiant.class);
       System.out.println("Un seul étudiant (en utilisant l'API gson)....");
       System.out.println(etd);
     }
   }else
    {
      System.out.println("Pas d'étudiants..");
    }
 }
}
```