République Algérienne Démocratique & Populaire

Ministère de l'enseignement Supérieure et de la Recherche Scientifique
Université Ibn Khaldoun de Tiaret

Faculté des Mathématiques et de l'Informatique
Département d'Informatique

Promotion : 3ème année Licence ISIL - **Semestre :** 5 - **Module :** SID. **Date :** ... / 06 / 2025 - **Durée :** 01H30Mn

Solution Type de l'Examen de Rattrapage en Systèmes

Nom et Prénom	

Restrictive Importante : mentionner votre **Nom et Prénom** et joindre cette feuille avec votre copie d'examen.

Exercice 1 : (6 Points) **(voir la page 3)**

- En utilisant le langage Java, compléter la partie mentionnée en pointillée dans les Class **RMIServer** et **RMIClient** destinées pour une application **Client/Serveur** RMI exécutée en **Localhost** sur le port **= 1099,** citée en **verso (***page 2/2***)**.

Exercice 2 : (8 Points)

- Définir un module IDL (Interface Definition Language) « **medical** » permettant de modéliser les services d'un système de gestion de rendez-vous médicaux dans un environnement distribué CORBA comme suit :

```
module Medical { //  (0.5 point)
    struct Patient {
      string id;
      string nom;
      string prenom;
      string date_naissance;
      string telephone;
    };
    struct Doctor {
      string id;
      string nom;
      string specialite;
      string disponibilite;
    };
    struct Appointment {
      string patient_id;
      string doctor_id;
      string date;
      string heure;
    };
    struct HistoryEntry {
      string doctor_nom;
      string date;
      string heure;
    };          // (2 point)

    typedef sequence<Appointment> AppointmentList;
    typedef sequence<HistoryEntry> HistoryEntryList;        //(1 point)

    interface AppointmentService {      // (0.5 point)

      void register_patient(in string nom, in string prenom, in string
                            date_naissance, in string telephone);

      void add_doctor(in string nom, in string specialite, in string
                      disponibilite);

      boolean book_appointment(in string patient_id, in string doctor_id,
                               in string date, in string heure);

      boolean cancel_appointment(in string patient_id, in string date);

      AppointmentList list_appointments_by_doctor(in string doctor_id);

      HistoryEntryList get_patient_history(in string patient_id);//(4 points)
    };
};
```

Questions : (03 Points)[01 point pour chaque bonne réponse]

1. Décrire brièvement les notions **http** suivantes.

« **PUT** » : est employée pour envoyer un fichier ou des données qui seront directement stockés sous l'URL spécifiée.	**« Content-Length** » : La quantité de données transmises, exprimée en octets indiquant au serveur le nombre exact d'octets à lire.

2. Citer les avantages d'un « **Middleware** » ?

- ✓ Fournir une plateforme pour faire tourner les composants cotés serveur.
- ✓ Equilibrer leur charge
- ✓ Assurer l'intégrité des transaction (Atomoticité Cohérence Isolation Durabilité). (permettre d'assurer l'intégrité des données quand plusieurs utilisateurs tentent d'écrire au sein de la base de données.
- ✓ Maintenir une disponibilité élevée et sécurité d'environnement.
- ✓ Responsable des conduites de communication client/serveur

3. Citer un exemple de **Middleware** de type « **MOM** » ?

MOM (Message **O**riented **M**iddleware) :
Java Message Service (**JMS**) est une interface de programmation pour middleware orienté message peut être utilisé avec un serveur de messagerie (broker) **ActiveMQ** en prenant en charge plusieurs protocoles de messagerie et en offrant des fonctionnalités avancées de gestion des messages.

Abréviations : (03 Points)[0.5 pour chacune]

- Désabréger les abréviations suivantes :

MQTT	**DCOM**	**EJB**	**XML**	**ORB**	**GIOP**
Message Queuing Telemetry Transport	Distributed Component Object Model	Enterprise Java Beans	eXtensible Markup Language	Object Request Broker	General Iner ORB Protocol

Server

```
package rmi_claculator_thread;
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
public class RMIServer {
   public static void main(String[] args) {
      try {
         LocateRegistry.createRegistry(1099);
         CalculatorImpl calculator = new
                        CalculatorImpl();
         Naming.rebind("Calculator", calculator);
         System.out.println("Calculator RMI Service is
                              running");
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

// (0.5 Point pour chaque cas)

Client

```
package rmi_claculator_thread;
import java.rmi.Naming;
import java.util.Scanner;
public class RMIClient {
   public static void main(String[] args) {
      try {
          Calculator calculator = (Calculator)
          Naming.lookup("rmi://localhost/Calculator");
        Scanner scanner = new Scanner(System.in);
        boolean exit = false;
        while (!exit) {
          System.out.println("\nChoose an operation:");
          System.out.println("1. Addition");
          System.out.println("2. Subtraction");
          System.out.println("3. Multiplication");
          System.out.println("4. Division");
          System.out.println("5. Exit");
          System.out.print("Enter your choice: ");
          int choice = scanner.nextInt();
          if (choice == 5) {
            System.out.println("Exiting the program...");
            exit = true;
            break;
          }
          System.out.print("Enter the first number: ");
          int a = scanner.nextInt();
          System.out.print("Enter the second number: ");
          int b = scanner.nextInt();
          int result = 0;
          switch (choice) {
            case 1: // Addition
              result = calculator.add(a, b);
              break;
            case 2: // Soustraction
              result = calculator.subtract(a, b);
              break;
            case 3: // Multiplication
              result = calculator.multiply(a, b);
              break;
            case 4: // Division
              result = calculator.divide(a, b);
              break;
            default:
              System.out.println("Invalid operation. Please try
                                  again.");
              continue; // Recommencer le menu
          }
          System.out.println("Result: " + result);
        }
        scanner.close();
      } catch (Exception e) {
        e.printStackTrace();
      }
   }
}
```

----------***FINI***----------