CENTRE SCOLAIRE SAINTE-JULIENNE

TA 14 – Les sockets et les threads

Exercices Java – Série 10 – Enoncés

I- Mise en situation

Tu es analyste-programmeur dans une société et tu dois passer un test en langage Java. A travers une série d'exercices, tu dois comprendre et maitriser le langage Java pour obtenir la prime salariale.

II- Objets d'apprentissage

Appliquer	Transférer
• Modéliser une logique de programmation orientée objet • Déclarer une classe • Instancier une classe (objet) • Utiliser les méthodes de l'objet instancié • Traduire un algorithme dans un langage de programmation • Commenter des lignes de codes. • Tester le programme conçu	• Développer une classe sur la base d'un cahier des charges en respectant le paradigme de la programmation orientée objet (POO) • Programmer en recourant aux classes nécessaires au développement d'une application orientée objet • Corriger un programme défaillant • Améliorer un programme pour répondre à un besoin défini

Connaître

- Différencier la programmation impérative de la programmation orientée objet
- Caractériser une classe
- Décrire la création d'un objet (instanciation)
- Identifier l'instance d'une classe
- Caractériser les attributs dans une classe (encapsulation)
- Caractériser les méthodes dans une classe (encapsulation)
- Décrire la création d'un constructeur
- Différencier les types de visibilités

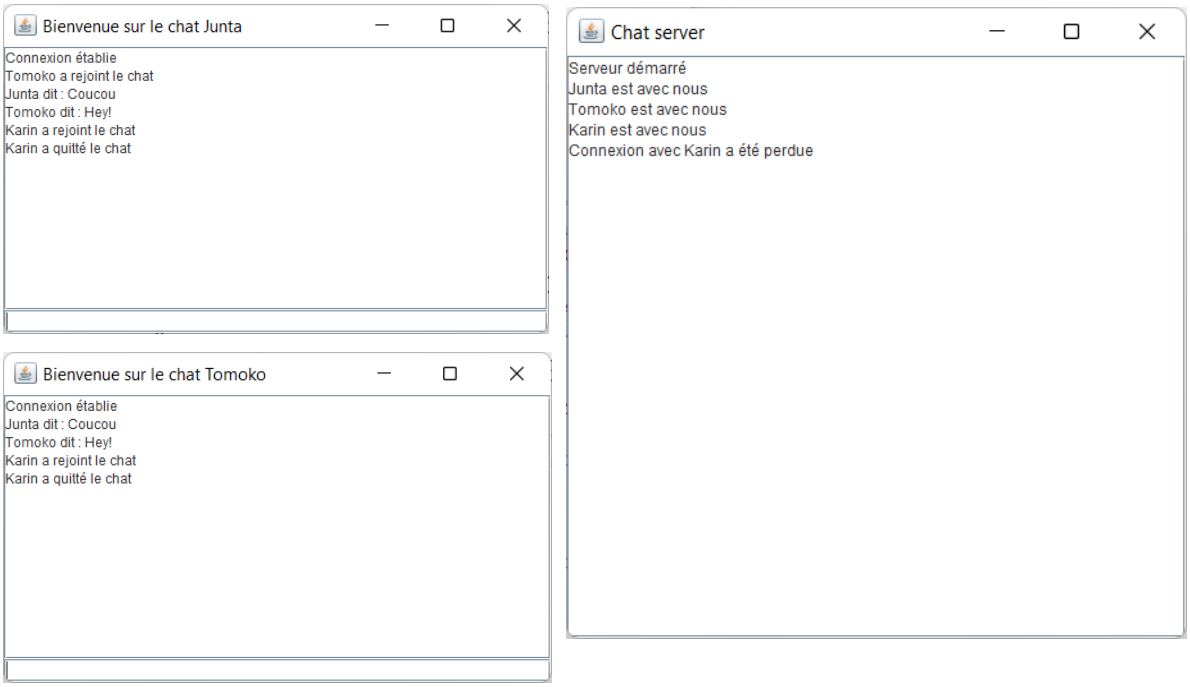
III- Travail à accomplir

1. Analyser l'énoncé du point IV correspondant au numéro de l'exercice demandé.
2. Modéliser en diagramme de classes l'exercice.
3. Réaliser l'exercice.
4. Commenter le travail.
5. Visualiser le travail.
6. Sauvegarder le document suivant les instructions données.
7. Imprimer le(s) document(s)

IV- Enoncés

1. Ex01 - Messenger

Dessiner les interfaces graphiques suivantes:



Ecrire une classe **ChatServer** étendue JFrame qui requiert comme variables d'instance:

- une zone d'affichage de type JTextArea;
- un numéro de port de type int initialisé à 2001;
- un socket de type ServerSocket;
- un socket de type Socket;
- un booléen pour l'état de marche du serveur.

Cette classe contient la méthode *startServer()* qui:

- crée le socket du serveur;
- accepte un client;
- crée un flux entrant pour connaître le nom du client;
- met à jour la zone d'affichage;
- crée une instance de ChatManager en lui passant le socket du serveur, le nom du client et la zone d'affichage;
- démarre le manager du chat.

Le constructeur de cette classe est chargé de créer l'interface graphique.

Ecrire une classe **ChatManager** étendue Thread qui requiert comme variables d'instance:

- une zone d'affichage de type JTextArea;
- un socket de type Socket;
- un nom pour client qui s'est connecté;
- un flux entrant de type DataInputStream;
- un flux sortant de type DataOutputStream.

Cette classe nécessite des variables de classe:
- un vecteur de type Vector qui contient les chats du manageur;
- un booléen pour l'état de marche du manageur.

Cette classe contient les méthodes:
- *demarrer()* qui:
    - instancie les flux de communication;
    - démarre le thread.
- *run()* qui
    - appelle la méthode *sendMessage* pour informer les personnes de la venue d'un nouveau sur le chat;
    - ajoute le chat courant dans le vecteur;
    - écoute les messages entrants et les envoie via la méthode *sendMessage* aux connectés;
    - retire un client déconnecté du vecteur, mentionne au serveur la perte de connexion avec ce dernier et informe les connectés via la méthode *sendMessage* quand un client se déconnecte.
- *sendMessage()* qui envoie de manière synchronisée via une énumération un message aux connectés et stoppe le thread en cas d'erreur.

Le constructeur de cette classe est chargé d'instancier les variables d'instance et les méthodes accesseurs se limitent aux get().

Ecrire une classe **ChatClient** étendue JFrame qui requiert comme variables d'instance:
- une zone d'affichage de type JTextArea;
- une zone de saisie de type JTextField;
- un numéro de port de type int initialisé à 2001;
- un nom pour le client;
- une ip pour l'adresse du serveur;
- un socket de type Socket;
- un booléen pour l'état de marche du client.

Cette classe nécessite des variables de classe:
- un flux entrant de type DataInputStream;
- un flux sortant de type DataOutputStream.

Cette classe contient la méthode *startClient()* qui:
- crée le socket du client;
- crée les flux de communication;
- communique le nom du client via le flux sortant;
- affiche tous les messages reçus dans la zone d'affichage;
- ferme les flux et le socket en cas d'arrêt du client.

Le constructeur de cette classe est chargé de créer l'interface graphique en sachant qu'un texte, qui est saisi dans la zone de saisie, est validé par la touche « enter ». L'événement envoie le message via le flux sortant au chat manager.