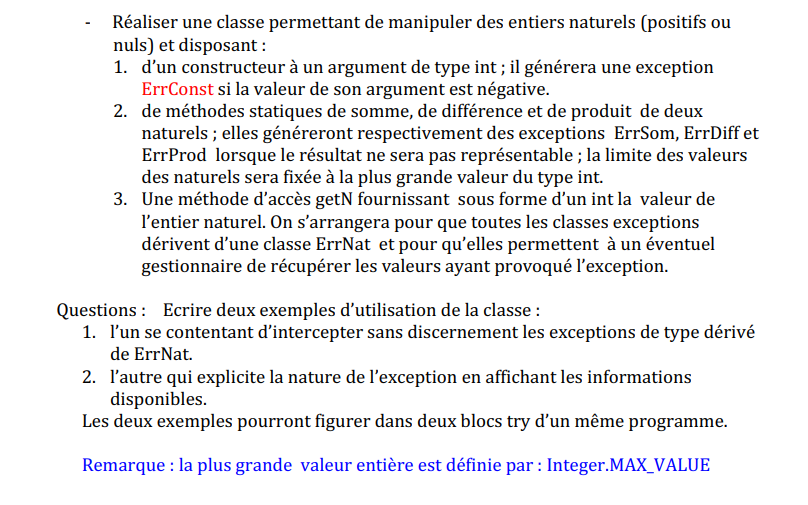
**Exam JAVA :**

**Exercice 1 : (Exception)**

****

Exercice 2 : (Collection)

Exercice : Gestion d'un Groupe de Participants

**Partie 1 : Modélisation d'un Participant**

1. Créez une classe Participant dans le package gestionParticipants avec les attributs suivants :

Nom du participant (String)

Liste de notes dans un ArrayList<Integer>

Moyenne de type double. Un participant sans aucune note aura une moyenne nulle.

1. Implémentez le constructeur de la classe Participant pour permettre uniquement l'initialisation du nom du participant.
2. Ajoutez la méthode ajouterNote(int note) pour permettre l'ajout d'une note au participant.
3. Ajoutez la méthode calculerMoyenne() pour calculer la moyenne du participant.
4. Overridez la méthode toString() pour afficher le nom, les notes et la moyenne du participant.

**Partie 2 : Modélisation d'un Groupe de Participants**

1. Créez une classe GroupeParticipants dans le package gestionParticipants avec l'attribut listeParticipants de type ArrayList<Participant>.
2. Ajoutez la méthode nombre() pour renvoyer le nombre de participants dans le groupe.
3. Ajoutez la méthode getListe() pour renvoyer la liste des participants.
4. Ajoutez la méthode ajouterParticipant(Participant participant) pour ajouter un participant au groupe.
5. Ajoutez la méthode chercher(String nom) pour rechercher un participant par son nom dans le groupe.
6. Ajoutez la méthode lister() pour afficher la liste des participants en utilisant la méthode toString de la classe Participant.

**Partie 3 : Implémentation de l'Interface Comparable**

1. Modifiez la classe Participant pour qu'elle implémente l'interface Comparable<Participant>.
2. Implémentez la méthode compareTo(Participant autreParticipant) pour comparer les participants par leur moyenne.
3. Modifiez la méthode main de la classe de test pour tester la méthode compareTo.

**Partie 4: Ajout de Fonctionnalités au Groupe de Participants**

1. Ajoutez la méthode meilleurParticipant() dans la classe GroupeParticipants pour retourner le participant ayant la meilleure moyenne.
2. Ajoutez la méthode trierParticipants() dans la classe GroupeParticipants pour trier la liste des participants par ordre croissant de moyennes.
3. Modifiez la méthode main de la classe de test pour tester ces nouvelles fonctionnalités.

**Projet : Messagerie**

Bien sûr, voici l'énoncé de l'exercice :

**Projet de Messagerie avec JDBC, DAO, SWING, et Socket**

**Description :**

Vous êtes chargé(e) de développer une application de messagerie client-serveur en utilisant Java. L'application devra permettre à plusieurs utilisateurs de se connecter au serveur, d'envoyer des messages textuels, et de voir les messages reçus.

**Composants du Projet :**

**Client :**

Interface graphique SWING pour la saisie des messages et l'affichage des conversations.

Gestion des connexions Socket pour communiquer avec le serveur.

Utilisation de JDBC pour enregistrer les messages dans la base de données.

**Serveur :**

Gestion des connexions Socket avec les clients.

Utilisation de JDBC pour la persistance des messages dans une base de données.

Implémentation d'un DAO pour gérer les opérations liées à la base de données.

**Base de Données :**

Utilisation de JDBC pour créer une base de données permettant de stocker les messages.

**Fonctionnalités Attendues :**

**Connexion :**

Les utilisateurs doivent pouvoir se connecter au serveur avec un nom d'utilisateur.

**Envoi de Messages :**

Les utilisateurs connectés peuvent envoyer des messages textuels aux autres utilisateurs.

**Affichage des Conversations :**

Les messages échangés doivent être affichés dans l'interface graphique du client.

**Persistance des Messages :**

Les messages doivent être enregistrés dans une base de données à chaque envoi.

**Gestion des Erreurs :**

Gérer les déconnexions et les erreurs de communication entre le client et le serveur de manière robuste.

**Contraintes Techniques :**

Utiliser JDBC pour la manipulation de la base de données.

Mettre en œuvre un modèle DAO pour encapsuler les opérations liées à la base de données.

Utiliser SWING pour l'interface graphique du client.

Implémenter la communication client-serveur via des Sockets.

Bonus (Optionnel):

Ajouter la possibilité d'envoyer des fichiers.

Implémenter une fonctionnalité de chat de groupe.

**Conseils :**

Divisez votre code en modules logiques (client, serveur, DAO) pour une meilleure organisation.

Testez votre application avec plusieurs clients pour assurer une communication fluide.

Documentez votre code de manière claire pour faciliter la compréhension.

**Note :** N'hésitez pas à personnaliser cet énoncé selon vos préférences et compétences. Bonne chance dans votre projet !

**Exercice : (Thread)**

**Objectif de l'exercice** : Créer un programme Java simulant une course entre plusieurs coureurs à l'aide de threads.

**Description :**

1. Créez une classe Coureur implémentant l'interface Runnable. Chaque coureur aura un nom et parcourra une distance pendant la course.
2. Dans la classe principale (CourseMain), créez une course avec plusieurs coureurs. Utilisez la classe Thread pour démarrer chaque coureur dans un thread séparé.
3. Dans la méthode run de la classe Coureur, simulez la course en faisant avancer le coureur sur une distance donnée à des intervalles aléatoires. Affichez le progrès du coureur à chaque étape.
4. Affichez le résultat de la course une fois que tous les coureurs ont terminé.
5. Exécutez le programme et observez la simulation de la course avec les différents threads.

