Rapport du TP :

Tâche 1 : Créer une classe abstraite Transaction.

Attributs :

protected String transactionID : L'ID de la transaction.

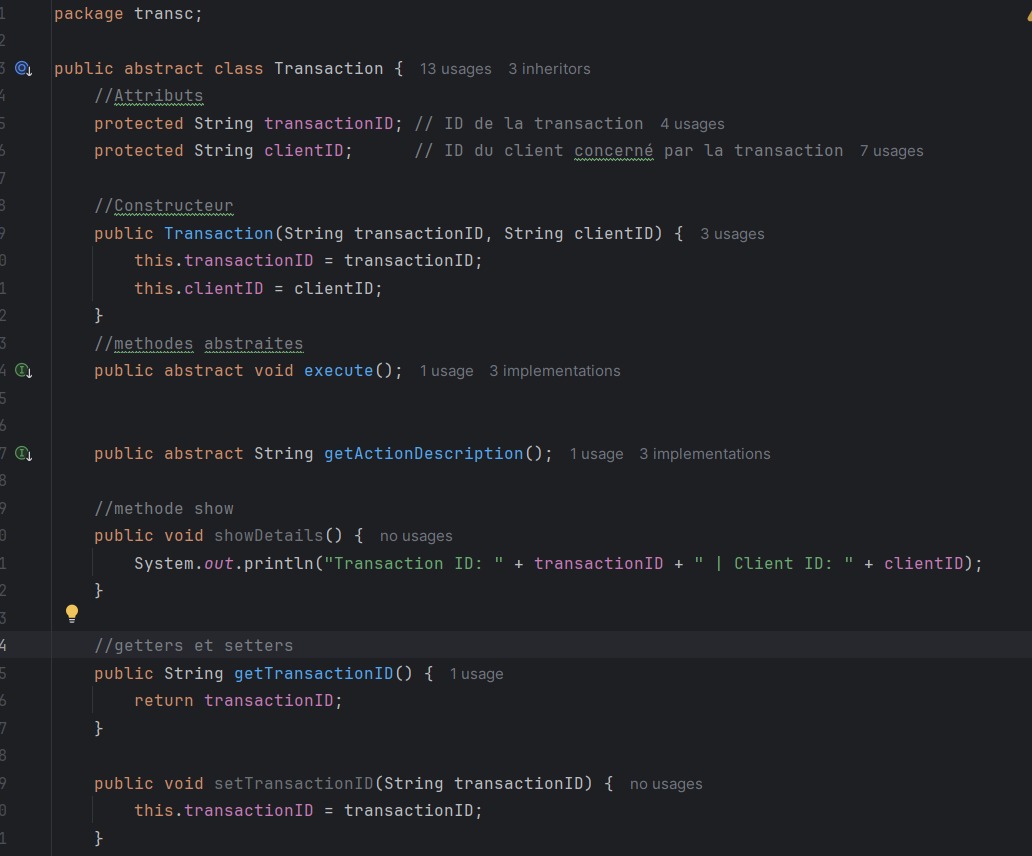
protected String clientID : L'ID du client concerné par la transaction.

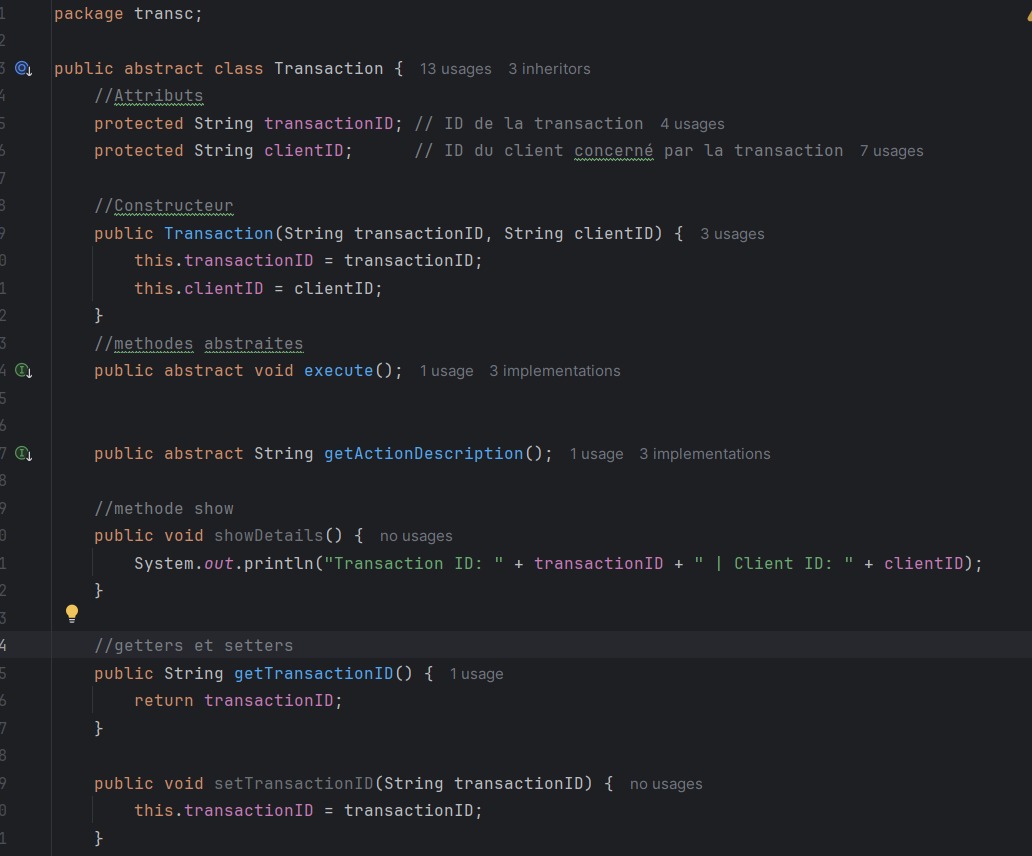
Méthodes :

Méthode abstraite public abstract void execute() : à implémenter dans les sous-classes pour exécuter la transaction.

Méthode non abstraite public void showDetails() : Affiche les informations de base sur la transaction. Exemple de sortie :

Transaction ID: T001 | Client ID: C001





Cette tâche consiste à créer une classe abstraite qui servira de base à différents types de transactions. Elle contient des attributs communs à toutes les transactions (ID de la transaction et du client) et une méthode abstraite execute()que les sous-classes seront implémentées. Cela permet de définir un cadre pour toutes les transactions.

Tâche 2 : Créer les sous-classes de Transaction pour différents types de transactions.

Sous-classe InsertTransaction :

Implementez la méthode execute() pour afficher un message indiquant l'insertion d'un nouveau client.

Exemple de sortie :

Insertion d'un nouveau client avec ID: C001

Sous-classe UpdateTransaction :

Implementez la méthode execute() pour afficher un message indiquant la mise à jour des informations d’un client.

Exemple de sortie :

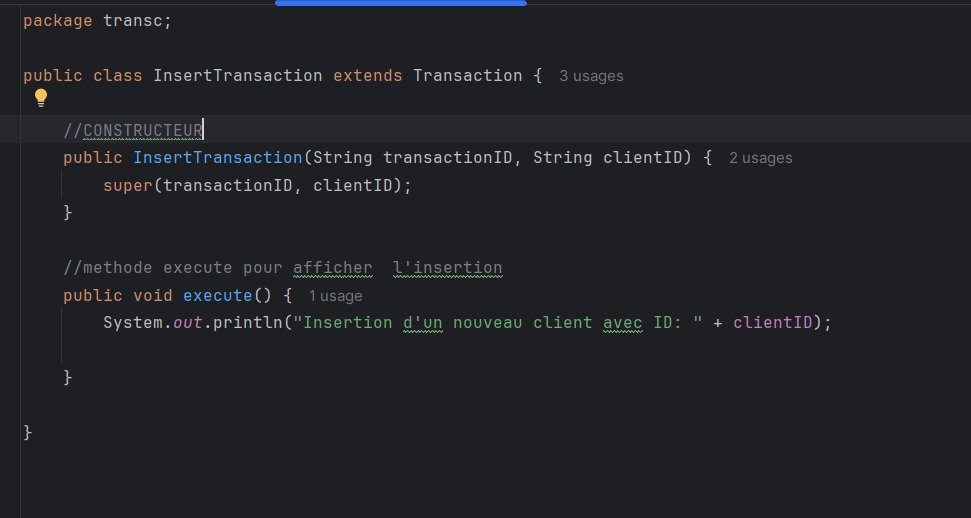
Mise à jour des informations du client avec ID: C002

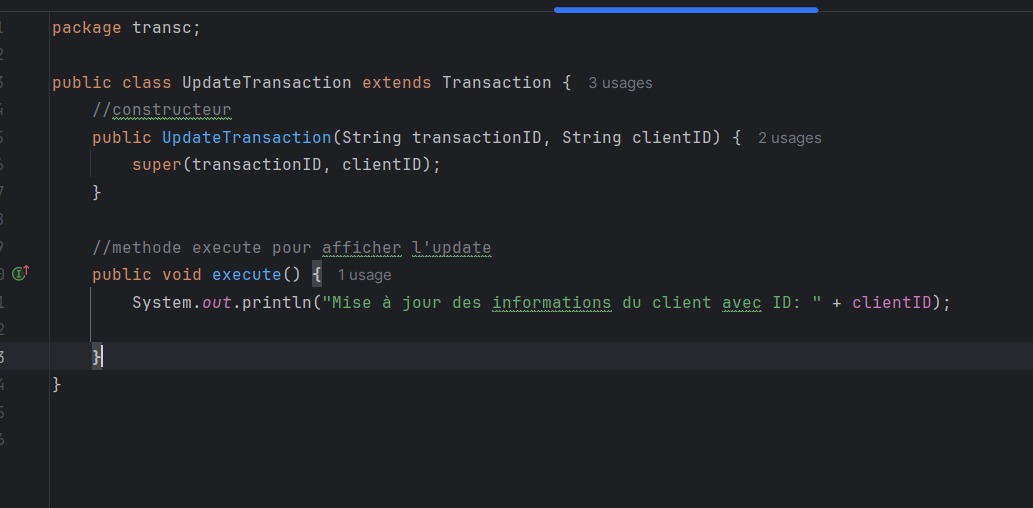
Sous-classe DeleteTransaction :

Implementez la méthode execute() pour afficher un message indiquant la suppression d’un client.

Exemple de sortie :

Suppression du client avec ID: C003





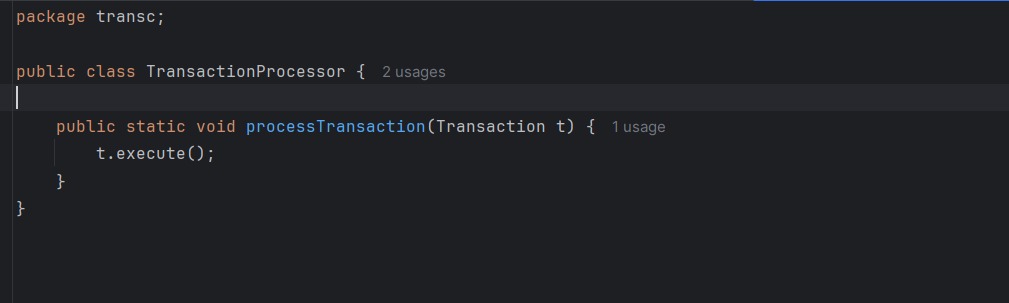


Ici, on crée trois sous-classes (InsertTransaction, UpdateTransaction, DeleteTransaction), chacune représentant un type de transaction différent. Ces cours doivent implémenter la méthode execute()pour exécuter des actions spécifiques comme l'insertion, la mise à jour ou la suppression d'un client.

Tâche 3 : Créer une méthode statique processTransaction(Transaction t) dans une classe TransactionProcessor.

Cette méthode doit prendre en paramètre un objet de type Transaction et appeler sa méthode execute().

Cela permet d'utiliser le polymorphisme pour traiter de manière uniforme différents types de transactions.



Dans cette tâche, il s'agit de créer une méthode qui prend n'importe quelle transaction en paramètre et exécute la méthode execute()correspondante. Cela permet de gérer différents types de transactions de manière uniforme, en utilisant le concept de polymorphisme.

Tâche 4 : Créer une méthode pour tester le polymorphisme.

Créez une liste d'objets Transaction contenant différents types de transactions (InsertTransaction, UpdateTransaction, DeleteTransaction).

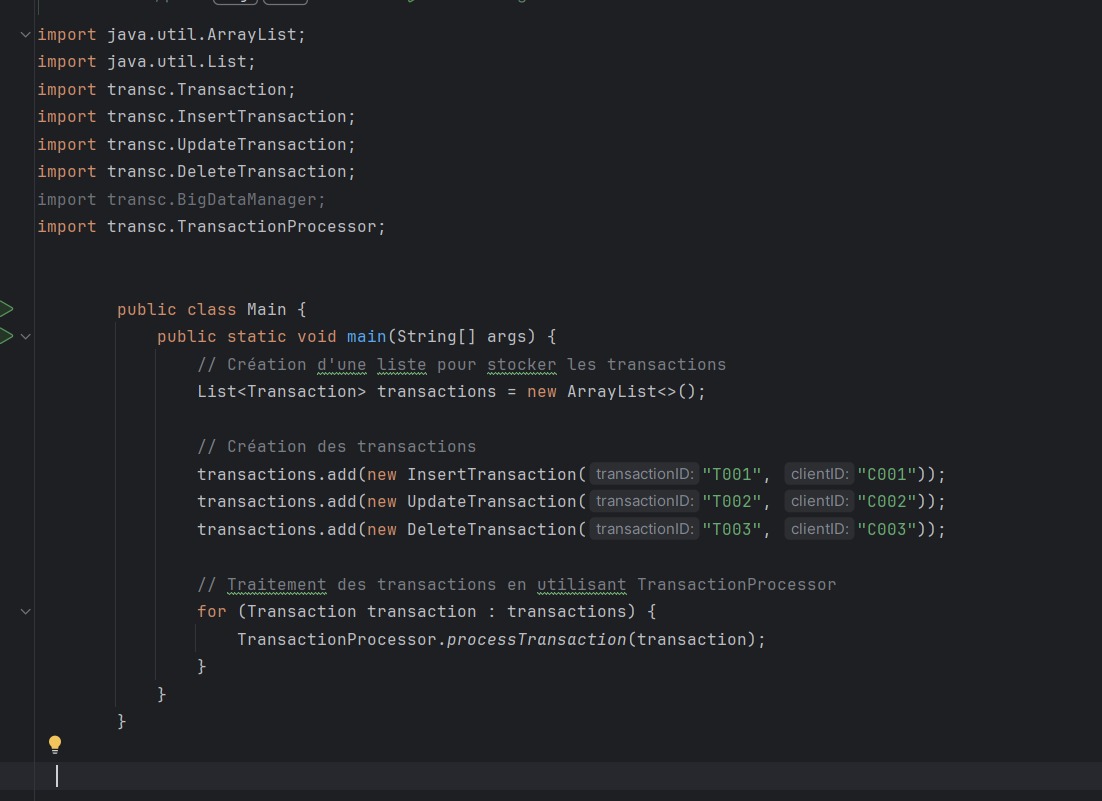
Utilisez une boucle pour parcourir la liste et appelez processTransaction() pour chaque transaction.

Exemple de sortie attendue :

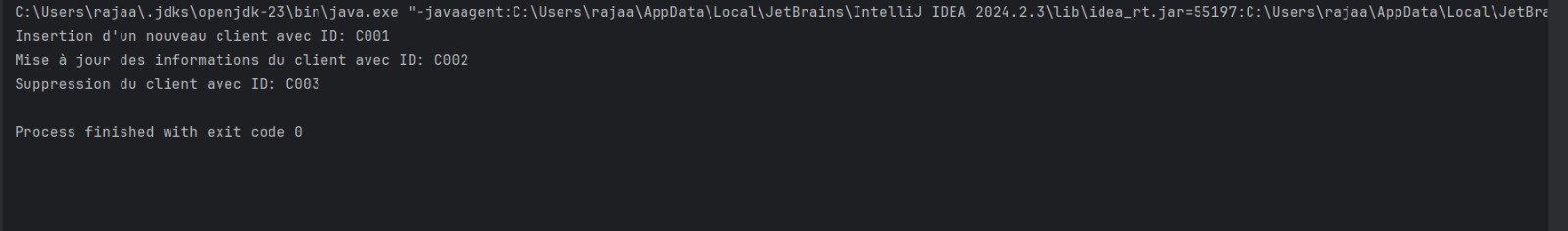
Transaction ID: T001 | Client ID: C001 | Insertion d'un nouveau client.

Transaction ID: T002 | Client ID: C002 | Mise à jour des informations du client.

Transaction ID: T003 | Client ID: C003 | Suppression du client.



Exécution:



Pour cette tâche, on crée une liste de transactions (de types différents) et on les traite toutes de manière identique avec la méthode processTransaction(). Cela permet de tester le polymorphisme et de vérifier que toutes les transactions sont exécutées correctement, peu importe leur type.

Tâche 5 : Créer une classe BigDataManager.

Cette classe doit contenir une liste de transactions et offrir des méthodes pour :

Ajouter des transactions à la liste.

Traiter toutes les transactions de la liste en utilisant TransactionProcessor.processTransaction().

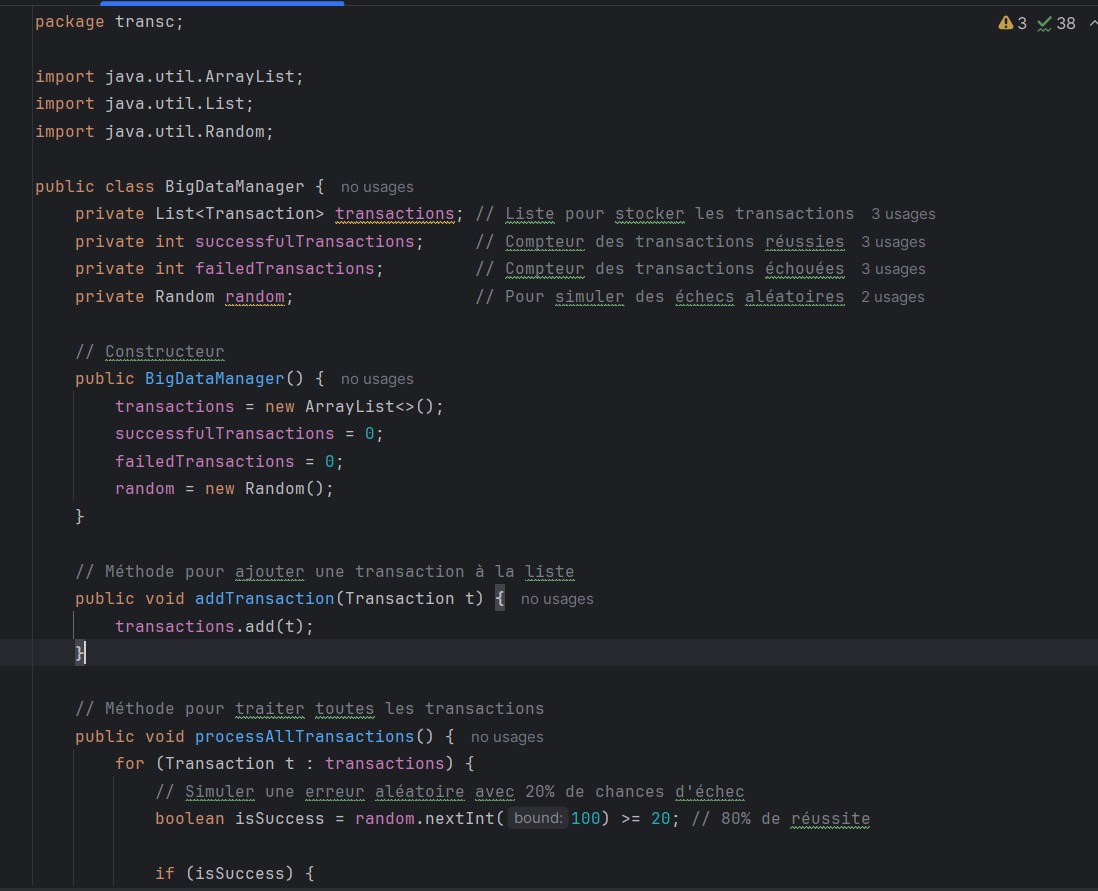
Compter le nombre de transactions réussies et échouées (simulez des erreurs aléatoires pour certaines transactions).

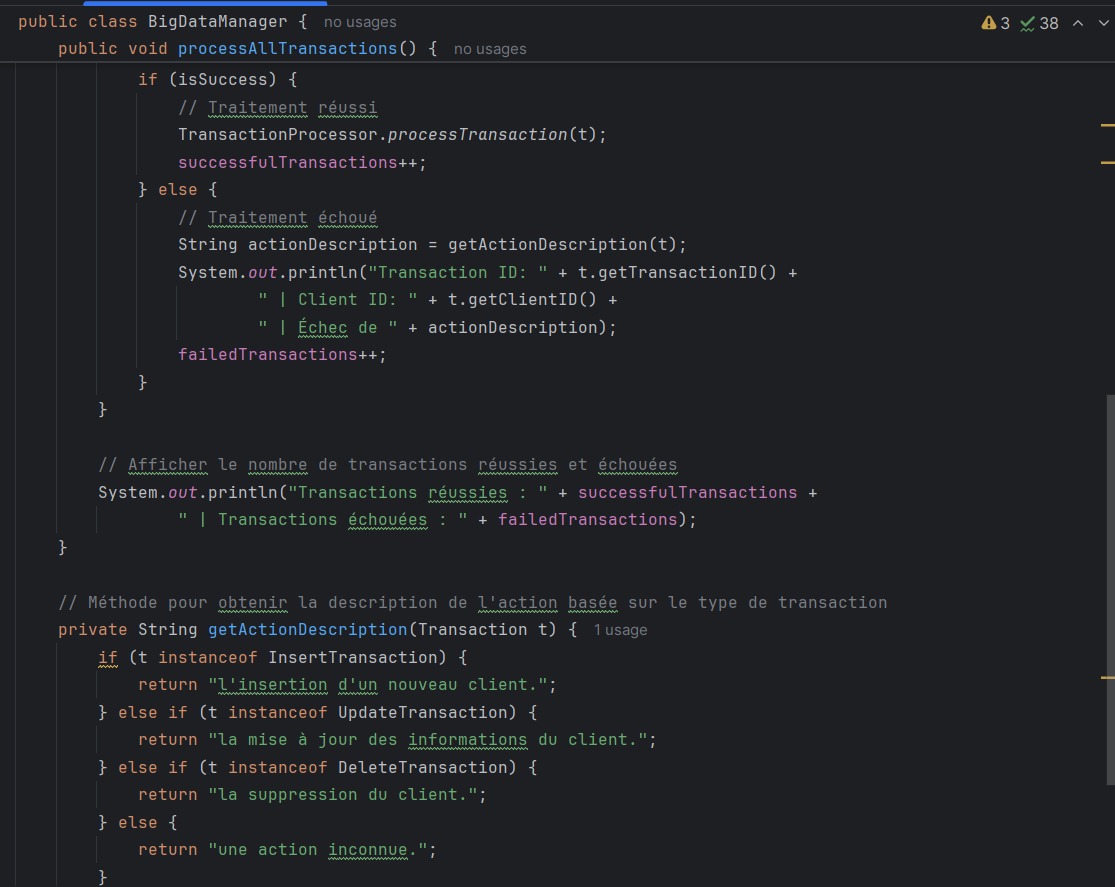
Exemple de sortie pour cette étape :

Transaction ID: T001 | Client ID: C001 | Insertion d'un nouveau client.

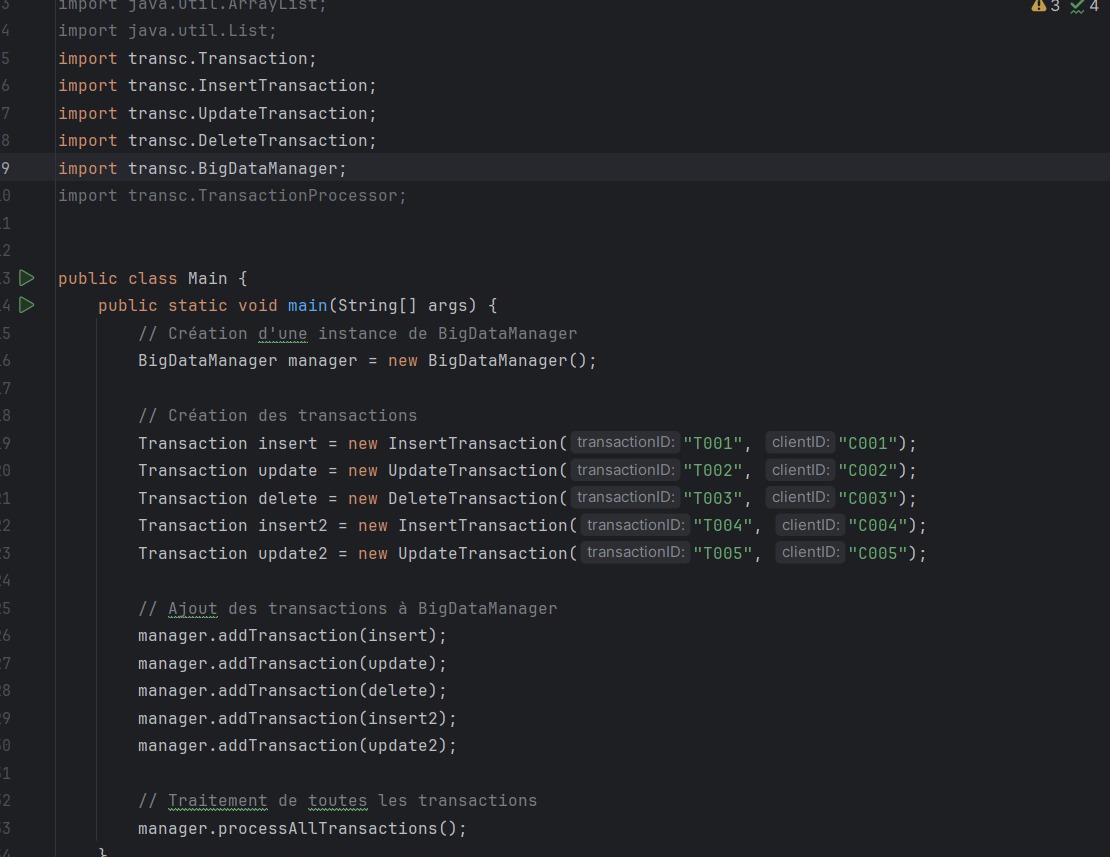
Transaction ID: T002 | Client ID: C002 | Mise à jour des informations du client.

Transaction ID: T003 | Client ID: C003 | Échec de la suppression du client.

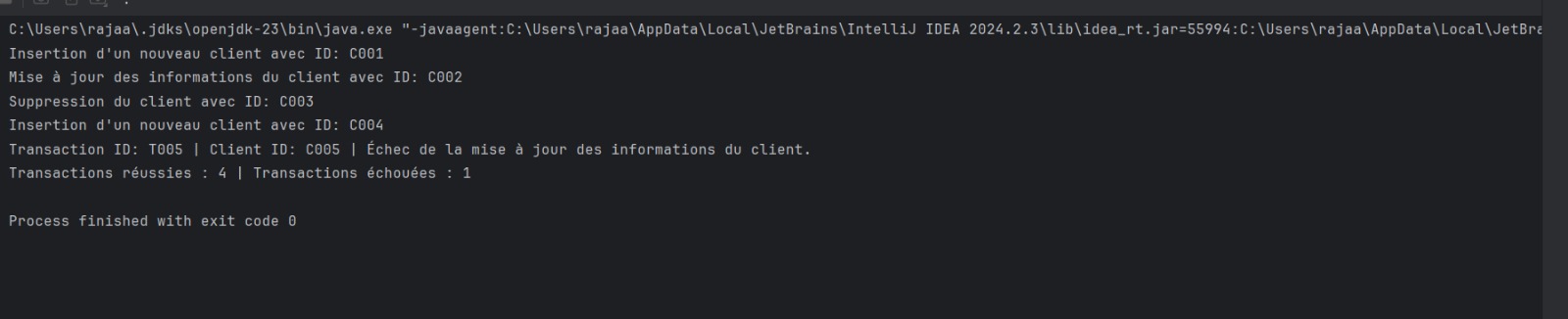




New Main:



Exécution:



La dernière tâche consiste à gérer un grand nombre de transactions. On crée une classe BigDataManager pour stocker les transactions, les exécuter, et compter les réussites ou échecs simulés. Cela permet de traiter efficacement un grand volume de transactions tout en surveillant leur succès ou leur échec.

Fait par Raja Fdili et Khadija Nachid Idrissi