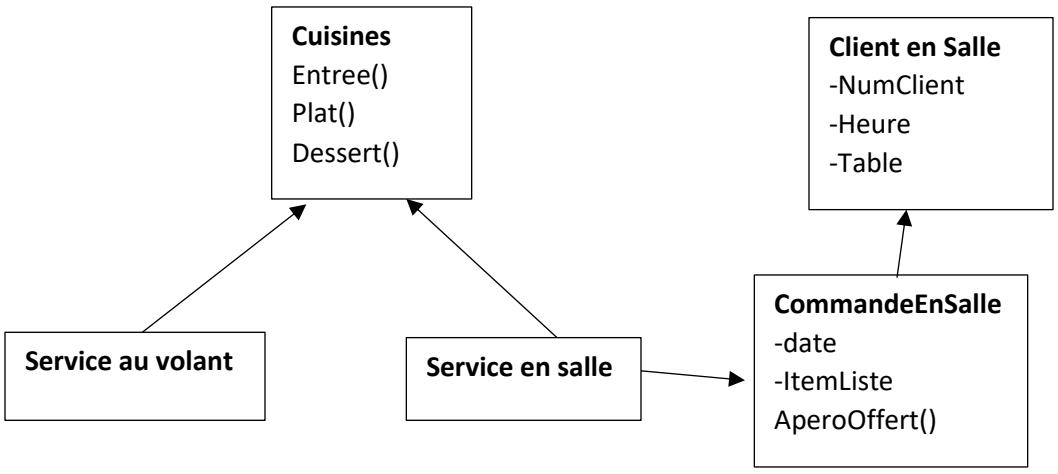
Principe LCOM Question 2 TP2 Programmation objet avancé

A) **Cohésion entre méthodes** : un lien de cohésion temporelle est un exemple de cohésion entre méthode. Un programme devant s'exécuter par un chaînage de méthodes (les classes appelant la bibliothèque SocketIO par exemple) suivent une procédure bien précise pour fonctionner. Ces méthodes sont considérées comme ayant une bonne cohésion entre méthode.
Cohésion entre classe : La cohésion fonctionnelle est un exemple de de cohésion entre classe. La modélisation d'une entreprise par différents services (production, comptabilité, direction et communication) est un exemple de cohésion entre chaque classe. Chacune de ces classes ont une méthode qui leur est spécifique. Chaque classe remplis la tâche qui lui est dédiée.
Cohésion de l'héritage : L'exemple du thread permet de démontrer la cohésion de l'héritage. Ainsi une classe fille permet d'ajouter des particularité à une classe mère tout en gardant les propriétés de son ascendante.

Ci-dessous un exemple de modèle ayant une forte cohésion entre classe, méthodes et héritage.



Assil ESSIMOU
Nyemo KOUMADI
Rama SUNTHARASARMA

B) La métrique LCOM (Lack of COhesion Method) permet de quantifier la cohésion d'une classe à l'aide de l'algorithme suivant :

$P = 0$; $Q = 0$;

Parcourir toutes les paires de méthodes

Si les méthodes ne partagent pas de données alors P++ ;

Si les méthodes partagent au moins une donnée Q++

Résultat = $P - Q$.

Si le résultat est > 0 alors la classe doit être coupé.

En appliquant cette métrique a la classe CashRegister, nous obtenons les valeurs $P = 1; Q = 2$. Selon la métrique, la classe n'as pas besoin d'être coupé, car $P - Q = -1 (< 0)$.

C) Selon la métrique, la classe CashRegister est cohésive et n'as pas besoin d'être coupé. Néanmoins, la méthode RecordPurchase et ReceivePayment ne partagent aucun attribut en commun. Il reste donc un moyen de décomposer la classe CashRegister en sous-ensembles plus indépendants, par héritage.

D) En appliquant le principe SRP et ISP décris dans l'extrait en intitulé, nous pouvons restructurer la classe CashRegister en deux responsabilités :

- Gestion du capital (méthodes ReceivePaiement et RecordPurchase)
- Gestion de transaction (méthode giveChange)

Nous pourrions séparer chacune de ces tâches en deux classes. Ainsi, la classe CashRegister n'est pas cohésive selon la définition exposée dans l'intitulé.

La classe reste néanmoins simple et intuitive de prise en main. En effet, les deux responsabilités sont implémentées par trois méthodes synthétiques et simples.

Ces trois méthodes ont pour finalité de mettre en place un registraire de transactions financières.

C'est pourquoi il apparaît que la modélisation SRP et ISP sont des paradigmes pouvant apporter une importante amélioration dans des projets d'envergure importante. Nous pensons donc que cet outil est un moyen d'optimiser un code et non pas un outil indispensable de modélisation et conception.

E) La nouvelle structure de la classe est proposée dans le fichier CashRegister_modif.java ci-joint. Pour augmenter la cohésion, nous mettons en place un lien d'héritage entre une classe CashRegister et la classe ChangeGiver. La première classe est une borne de paiement qui ne rend pas la monnaie (comme dans les bus de la STS). La classe ChangeGiver rend la monnaie. Elle pourra évoluer en mettant en place un système de réduction ou fidélisation du client sans avoir à influer sur sa classe mère, CashRegister.