جامعة الحسن الأول
UNIVERSITÉ HASSAN 1ER

Département Mathématique et Informatique

ENSA KHOURIBGA ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES

Ingénierie avancé objet
GI3

Énoncés
Feuille n° 1

Principes SOLID

NB : Avant de démmarrer le TD, veuillez effecttuer les opérations suivantes :
- Connectez vous à mon ddépot et recuperer les packages du TD1
- Partager votre dépot avec moi

N'oubliez pas de mettre a ? jour votre de ?po ?t local à chaque fin d'exercice

1 Refacetoring du code

Dans la premie ?re partie de ce TD, nous allons corriger le code afin de respecter les principe du SOLID vue en cours.

Exercice 1
Le programme permet de réaliser les fonctionnlités d'une factture :
- Création facture
- Ajout des produits
- Calcul prix total sans et avec taxe
- Sauvegarde de facture
- Imprression de facture

1 Identifier le principe du SOLIID non respecté
2 Donnez un nouveau diagramme de classe et justifier votre choix
3 Implémnter votre solution (Assurez-vous qu ?en effectuant les changements qu aucun code n'as été ajouté mais seulement refactorisé)
4 Réaliser un main qui effectue les tests

Exercice 2
Le programme permet de réaliser les fonctionnlités d'un compte :
- Créattion de compte
- Gestion des opérations bancaires

1 Identifier le principe du SOLIID non respecté
2 Donnez un nouveau diagramme de classe et justifier votre choix
3 Implémnter votre solution
4 Réaliser un main qui effectue les tests

Exercice 3
Un tableau de Young (ou tableau de Young-Ferrers) est une disposition d'entiers dans un tableau de cases, où les lignes et les colonnes sont ordonnées de manière croissante. Plus formellement, un tableau de Young remplit ces deux conditions :

1 Les nombres dans chaque ligne sont disposés dans un ordre croissant, c'est-à-dire que chaque élément dans une ligne est inférieur ou égal à celui qui le suit.
2 Les nombres dans chaque colonne sont également disposés dans un ordre croissant, c'est-à-dire que chaque élément dans une colonne est inférieur ou égal à celui qui se trouve en dessous.

Le tableau de Young doit implementer deux opérations : Extraire Min et inserer en gardant les deux conditions valables après chaque opération.
Nous proposon une imlémentation du tableau du Young :

1 Identifier le principe du SOLIID non respecté
2 Donnez un nouveau diagramme de classe et justifier votre choix
3 Implémnter votre solution
4 Réaliser un main qui effectue les tests

Exercice 4
Le programme permet de réaliser les fonctionnlités de gestions des comptes universitaires des professeurs et des étudiants

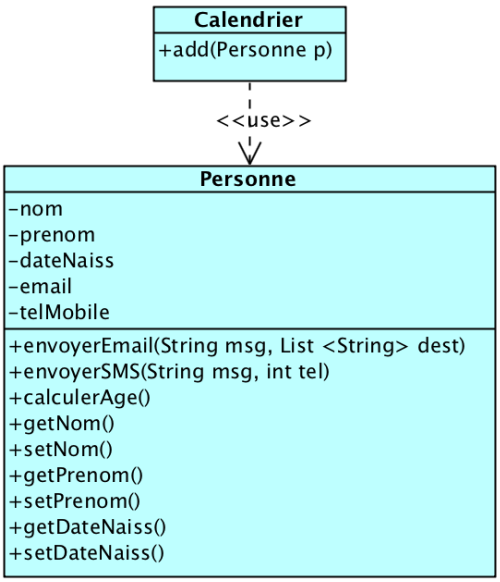
1 Identifier le principe du SOLIID non respecté
2 Donnez un nouveau diagramme de classe et justifier votre choix
3 Implémnter votre solution
4 Réaliser un main qui effectue les tests

2 Conception avancée

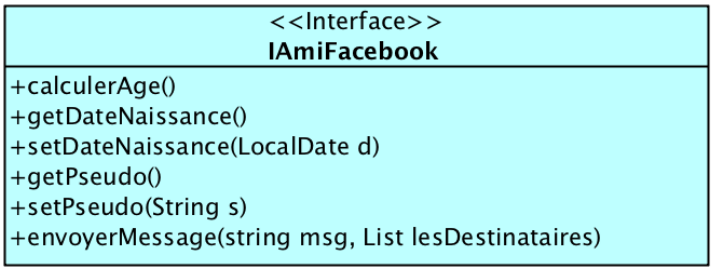
Exercice 5
Imaginons devoir développer une application qui gère les anniversaires : elle stocke les dates d'anniversaires de nos amis et leur envoie un message d' anniversaire automatique.
Nous avons donc une classe qui s'occupe de charger et d'enregistrer les dates d'anniversaires dans un Calendrier partagé. La classe **Calendrier** dispose ainsi d' une méthode add() qui s'écrit comme ceci :
publicclass**Calendrier** {
publicvoidadd(Personne personne) { /* TODO */
}
}
où Personne est une classe réutilisée de l'équipe de développement :

1.1) On constate donc que **Calendrier** est fortement couplée à **Personne**. Est-ce bien nécessaire ? Quelle première solution simple vue en cours permettrait de diminuer le couplage ?

1.2) On décide de construire une interface **IPersonne** avec juste les méthodes de **Personne**(extract interfacesous l'IDE).



Décrire l'interface **IPersonne**ainsi que la nouvelle signature de add() dans **Calendrier**.
Lors de la Revue de Sprint, vous apprenez du client que **Calendrier** ne va finalement jamais envoyer de SMS et d'emails à une Personne, mais un message pour leur anniversaire. **Calendrier** a donc besoin de connaître le nom de la personne et son âge, càd. qu' elle peut affecter une date de naissance, la consulter et envoyer un message.
IPersonneest-elle donc adaptée à notre situation ? Si non, décrire ce que serait l' interface souhaitée.
1.3) Votre chef vous apprend finalement que **Calendrier** devra aussi gérer les anniversaires des membres de Facebook, avec le contrat suivant :



Trouver les e ?le ?ments communs entre IPersonne et IAmi Facebook. Proposer ainsi une ame ?lioration de Calendrier visant a ? diminuer le couplage entre les 3 classes. 1.4)uel principe SOLID n' était pas vérifié dans l' exercice quand on voulait utiliser **IPersonne** dans Calendrier ?

Exercice 6
Imaginons qu'on dispose d'une classe qui applique des validations en fonction de l'âge d'un utilisateur :

```
public class ValidationAge {
public boolean peutBoireAlcool(int age) {
return age >= 18 ;
}
public boolean peutUtiliserFesseBouk(int age) { return age >= 13 ;
}
public boolean peutEtreEluMaire(int age) { return age >= 21 ;
}
}
```

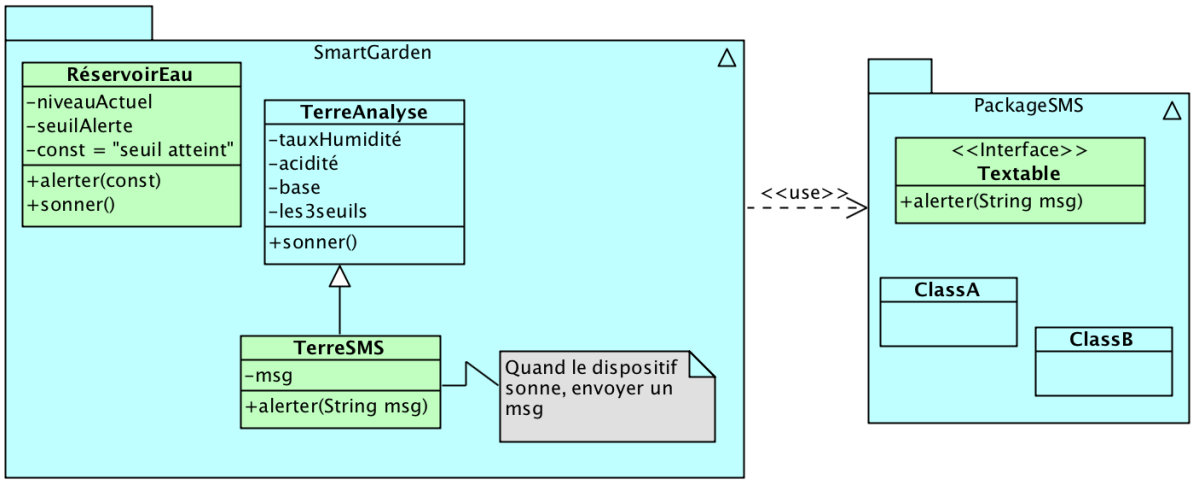
Cette classe est utilisée par tous les programmes de l'agence qui exploitent les utilisateurs. Six

mois plus tard, on apprend que notre application s' élargit à d' autres régiions du monde où les limites d' âge ne sont pas les mêmes...
4.1- Coder la solution proposée où on modifie la classe ValidationAge en envoyant la région en paramètre aux fonctions, pour tester si ça concerne quuel région (limites d'âge : 14, 15 et 19 pour les conditions ci-dessus). Testez si ça fonctionne.
4.2 - Pensez-vous que la classe obtenue respecte les principes SOLID ? Proposez une solution plus élégante.

Exercice 7
Dans un jardin connecté, différents dispositifs sont installés, certains fabriqués par nous, d' autres sont des composants propriétaire. Un suivi à distance est possible par notifications de SMS. Ainsi :
- Un **réservoir** fait maison (en vert) sonne quand il atteint un seuil d'alerte : on a ajouté une puce qui envoie alors un SMS au jardinier : "Seuil atteint" ;

- Un dispositif propriétaire d'analyse de la terre se met à sonner quand il y a une anomalie sur les mesures (humidité, acidité, base).
- On souhaiterait que le jardinier soit aussi informé à distance : on crée alors, en extension de ce capteur (en vert), un composant qui va **transmettre l' information au jardinier via un SMS.**

On a donc le S.I. suivant :



Mais on aimerait que le déclenchement d'une alerte puisse aussi se faire par une personne : on installe donc un nouveau composant, un **carillon**, qu'un employé du jardin actionne quand il constate un problème.
Et comme pour l'analyse de la terre, on programme le carillon pour qu'il puisse transmettre un SMS aux jardiniers.
7.1. Modifier le **DCL** en conséquence.
7.2. Quel **inconvénient** possède cette architecture ? Imaginez qu'on ajoute un autre composant sonore, puis un autre, etc.
7.3. Notre PKG SmartGarden dépend donc d' une interface **alerter()** par SMS (développée par nos soins). On aimerait que ce soit l' inverse, que le package technique d'alerte dépende du package Métier. Comment procéder ?

I. HAFIDI