Prérequis : connaissances de base du langage Java et de la programmation orientée objet.
Objectifs : découvrir les pratiques agiles de tests unitaires et refactoring.
Importance : ce TP fait partie de votre note de projet.

Le devoir consiste à créer un tutorial pour l'apprentissage par l'expérimentation des concepts orientés objet comme la classe et l'objet, à l'aide d'un exemple évolutif original, issu de votre imagination. Le thème est celui de votre choix, mais il se doit d'être cohérent et à visée pédagogique, voire ludique. Les évolutions que vous aurez à apporter progressivement à votre exemple devront rester cohérentes et parlantes. La première classe que vous aurez créée sera votre classe « fétiche » et donnera la tonalité du sujet choisi. Vous n'avez à donner aucune définition ni à élaborer une quelconque théorie. Il suffira de vous limiter à faire vivre à vos objets des expériences exaltantes, comme dans un conte de fées.

I. Première partie - BlueJ

- s'organiser en binôme et s'inscrire sur une liste de présence en indiquant la « classe fétiche », https://docs.google.com/spreadsheets/d/1zfeYk5gY8_4HIN9F3xaDaKUhAO859SCEaHR_4auGjLk/edit?usp=sharing
- en suivant le scénario suivant, rédiger un tutorial simple mais efficace permettant de créer et de tester des objets en prenant à chaque étape des « photos de vacances » (copies d'écran) et en les commentant brièvement
- générer le PDF et l'envoyer par email à zam@dauphine.fr en indiquant dans l'objet « [Agilite]-TPJU-<votre classe fétiche>-<vos noms> »
- durée : 1H30

Scénario à suivre (copie d'écran à chaque étape) :

1. Téléchargez BlueJ : http://www.bluej.org/
2. Installez-le sur votre machine
3. Créez un nouveau projet
4. Créez une première classe intitulée selon votre choix (assurez-vous que la classe est unique en l'inscrivant sur la liste de présence à côté des noms du binôme). Elle sera votre classe « fétiche »
5. Compilez la classe
6. Instanciez-là
7. Ajoutez 2 attributs, avec accesseurs et une méthode qui les manipulent
8. Instanciez-là de nouveau, en exécutant interactivement la méthode et en visualisant son effet sur l'état de l'objet créé.
9. Testez unitairement votre classe et montrez la barre verte
10. Ajoutez une seconde classe et associez-là (de façon unidirectionnelle) à la votre classe fétiche, avec une multiplicité 0..1 à 0..1.
11. Puis, ajoutez-y une méthode qui collabore avec la méthode de la classe fétiche pour obtenir un résultat basé sur la contribution des objets des deux classes
12. Instanciez les classes et reliez les objets, puis sauvegardez-les dans la fixture d'une classe de test (setup)
13. Créez interactivement une méthode de test qui utilise la fixture et montrez le résultat de son exécution et la couleur de la barre.

II. Seconde partie – Eclipse[1] et jUnit

14. Créez un projet éclipse
15. Importez les classes élaborées en BlueJ et organisez-les dans un package dédié
16. Implémentez une association bidirectionnelle « 0..1 à * » en l'encapsulant bien et testez unitairement sa robustesse.
17. Illustrez l'usage de deux techniques de refactoring (ex : rename et extractMethod)
18. Trouvez et parcourez le site officiel de jUnit. Lisez l'article « Test infected » et proposez une amélioration équivalente adaptée à votre exemple.
19. Exécutez les tests en ligne de commande
20. Citez une loi de Murphy associez-là à une situation rencontrée dans ce périple.

Bonne route
mz

[1] *Vous pouvez opter pour d'autres environnements de développement, après validation de votre enseignant*