TP – GENIE LOGICIEL AVANCEE SUR LE Test Unitaire et le Refactoring

Numéro d'inscription : 1153 H-F

Nom : RAMANDIMBINIRINA

Prénoms : Tahinjanahary Marie Angela

Mention : Informatique

Parcours : Informatique General

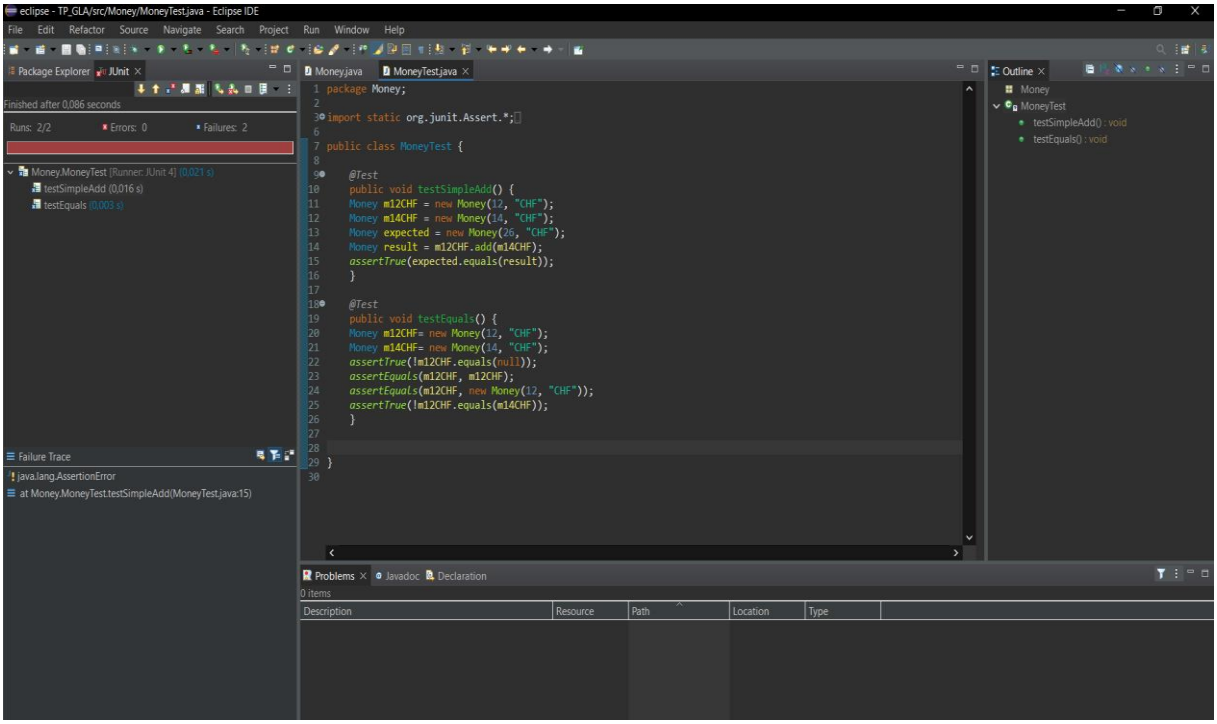
Niveau : M1

Réponse aux questions de chaque sur le Test Unitaire et Le Refactoring :

1- TEST UNITAIRE

Classe Money

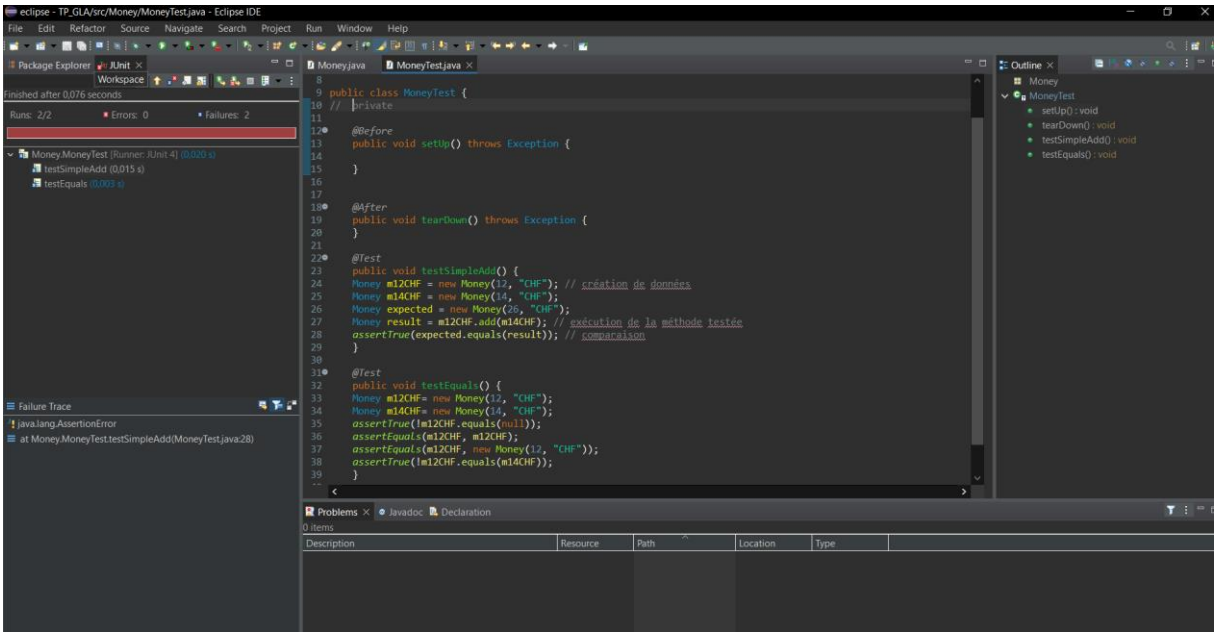
3) Après avoir créé les classe Money et MoneyTest y ajoutant la méthode **add** avec le cas précédemment défini **:** « Run > Run As > JUnit test », on a un résultat de Test Négative (c'est-à-dire que le test échoue) après l'exécution. Le schéma ci-dessous nous montre cela

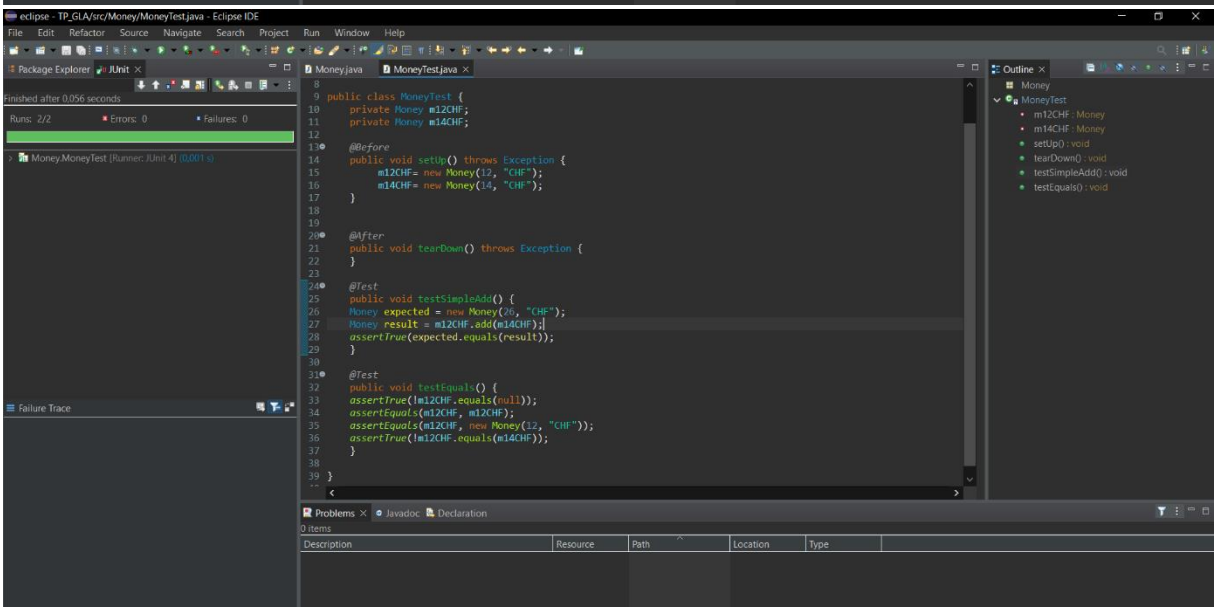


Cela nous explique que l'implémentation de la méthode « add » est incorrecte et cela entraine une addition incorrecte des montants d'argent.

5) En relançant la classe de test MoneyTest après avoir ajouté une méthode de test d'égalité, on obtient encore résultat de Test Négative parce qu'on n'a pas encore surcharger la méthode **equals** de la classe Money pour que le test sera avec succès et en utilisant aussi l'annotation @Before pour résoudre la

duplication de code dans les méthodes **testSimpleAdd** et **testEquals**. Le schéma suivant nous illustre cela.





Simplification

13- Lorsque le résultat de l'addition est un MoneyBag avec une seule valeur, cela signifie que le MoneyBag ne contient qu'un seul montant d'argent dans une seule devise. Dans ce cas, pour des raisons de simplicité et de cohérence, il est souhaitable que ce MoneyBag soit simplifier pour représenter directement ce montant d'argent unique au lien de rester sous forme de MoneyBag

2- LE REFACTORING

9) Si on veut déplacer la méthode toString() vers les sous classe « Homme et Femme », cela permet de personnaliser l'affichage des détails spécifiques a chaque type d'humain, comme le genre. Cela améliorerait la lisibilité du code et permettrait une représentation plus précise des objet « Homme et Femme ».