



Génie Logiciel (DSI310)

Examen Final

Date : 02/02/2023

Durée : 1h30

Filière : DSI

***** Tous les documents sont interdits *****

Exercice 1 : Test Junit (7 points)

On définit une classe **Argent** permettant de représenter un certain montant (montant dans une devise donnée (devise). Le montant doit être positif, et la devise s'exprime comme une chaîne de trois caractères, par exemple, EUR (euro), USD (dollar), MRU (Ouguiya), etc... Quand on additionne deux montants de même devise, le résultat a pour montant la somme des deux autres montants.

```
public class Argent {
    private int fMontant;
    private String fDevise;

    public Argent(int montant, String devise) {
        fMontant = montant;
        fDevise = devise;
    }

    public int montant() { return fMontant ; }

    public String devise() { return fDevise ; }

    public int somArgent(int a, int b) {

        if(b>0) {
            for(int i=0;i<=b;i++) {
                a++;
            }
        }else if(b<0){
            for(int i=0;i<=-b;i++) {
                a--;
            }
        }

        return a;
    }

    public Argent add(Argent a) {
        return new Argent(somArgent(montant(), a.montant()),devise());
    }
}
```



Questions :

1. On se donne la classe *ArgentTest* ci-dessous qui permet de tester si la classe *Argent* est correcte. Cette classe contient entre autres les méthodes *testBpositif* et *testBnegatif* qui permettent de tester la classe *Argent* selon que $b > 0$ ou $b < 0$ respectivement.

```
Public class ArgentTest{  
  
    @Test  
    public void testBpositif(){  
  
        Argent money1=new Argent(5,"MRU");  
        Argent money2=new Argent(2,"MRU");  
        int attendu=7;  
        int resultat=money1.somAdd(money2).amount();  
  
        assertTrue(attendu==resultat);  
  
    }  
  
    @Test  
    public void testBnegatif(){  
  
        Argent money1=new Argent(5,"MRU");  
        Argent money2=new Argent(-2,"MRU");  
        int attendu=3;  
        int resultat=money1.somAdd(money2).amount();  
  
        assertEquals(attendu, resultat);  
  
    }  
}
```

- a. Est-ce que la classe *Argent* passe les tests *testBpositif* et *testBnegatif*? Justifiez votre réponse.
 - b. On constate que dans chacune des méthodes de test ci-dessus, des objets *Argent* sont créés, des variables *attendu* et *resultat* sont déclarées. Proposez une solution pour éviter cette redondance en utilisant les tags *@Beforexxx* et/ou *@Afterxxx*. Donnez la nouvelle version de la classe.
 - c. On veut également afficher à la fin de chaque méthode de test le message «*Finish*». Donnez la nouvelle version de la classe.
2. Quel est l'intérêt des tags *@Test*, *@BeforeEach* et *@AfterAll* ?

Exercice 2 : Reverse Engineering (6 points)

En utilisant les techniques de reverse engineering vues en cours, propose une squelette de diagramme de classes correspondant aux classes Java A, B, C et D ci-dessous.

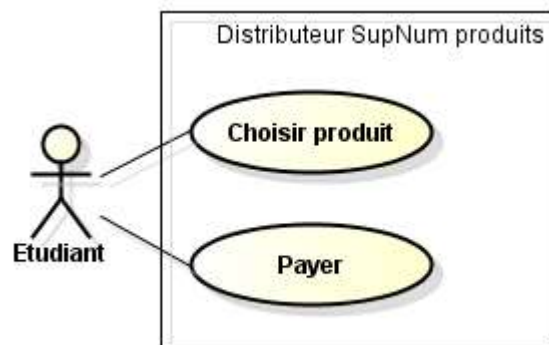
```
package Exam;

import java.util.ArrayList;
public class A {
    int a1, a2;
    ArrayList <C> ListC= new ArrayList<C>();
    B [] tabB= new B[10];
}

public class B { C c1; }
public class C extends D{ B b1; }
public class D { int d1, d2; }
```

Exercice 3 : Cas d'utilisation (7 points)

Les étudiants de DSI ont créé pour SupNum un distributeur de produits. Les étudiants peuvent ainsi choisir des produits alimentaires (pains, sardines, boissons,...) ou des produits de nettoyage (javel, savons, omo, ...). Une fois qu'un étudiant choisit les produits qu'il désire acheter, il doit ensuite payer ses achats. La figure suivante illustre les cas d'utilisation *payer* et *choisir produit*.



1. Il existe deux façons de payer les produits via le distributeur SupNum : par carte de crédit ou par espèces. Lors du paiement, l'étudiant peut récupérer son reçu de paiement s'il le veut.
2. Dans le cas du choix des produits alimentaires, l'étudiant doit vérifier la date limite de consommation du produit.
3. Seuls les étudiants de DSI peuvent, en plus, choisir les produits de type beauté.

Modifiez le diagramme pour prendre en considération le scénario décrit ci-dessus. Toutes les opérations sur le distributeur de SupNum nécessitent la saisie du matricule de l'étudiant.