DELPHES JABRI CDEV SOPRA





**TP 2 : Les tests**

**Module IMDW275**



**Sommaire :**

Table des matières

[I) INTRODUCTION 3](#_Toc27659448)

[***II)*** ***Développement*** 4](#_Toc27659449)

[Etape 1 : Prise en main du code 4](#_Toc27659450)

[Etape 2 : 4](#_Toc27659451)

[Etape 3 : 5](#_Toc27659452)

[Etape 4 : 5](#_Toc27659453)

[Etape 5 : 7](#_Toc27659454)

[Etape 6 : 8](#_Toc27659455)

[Un nom de currency doit faire entre 4 et 10 characteres : 8](#_Toc27659456)

[Ecrivez un test pour un nom de douze caractères : 8](#_Toc27659457)

[On ne peux pas mettre (methode) put une quantité negative de currency dans un CurrencyHolder 8](#_Toc27659458)

[On ne peux pas ajouter ou retirer 0 currency 9](#_Toc27659459)

[Un nom de currency ne doit pas commencer par la lettre a majuscule ou minuscule (2 tests) 10](#_Toc27659460)

[Un CurrencyHolder ne peux avoir une capacité inférieure à 1 (2 tests) 10](#_Toc27659461)

[Faire 2 tests unitaires pertinents pour la methode IsEmpty 11](#_Toc27659462)

[Un CurrencyHolder est plein (IsFull) si son contenu est égal à sa capacité (4 test, y a pas de piege) 12](#_Toc27659463)

[III) CONCLUSION 13](#_Toc27659464)

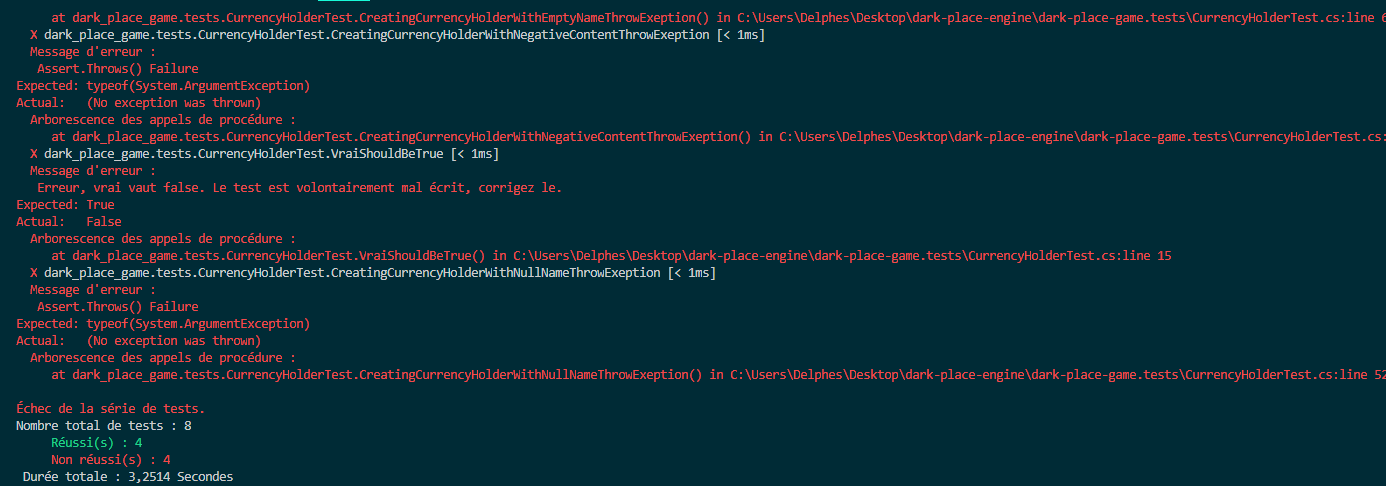
# INTRODUCTION

Durant ce TP nous allons partir d’un code existant et créer des tests unitaires et tester des levés d’exception.

# ***Développement***

## Etape 1 : Prise en main du code

**Réponse :**



## Etape 2 :

**Réponse :**

  public void VraiShouldBeTrue()

        {

            var vrai = true;

            Assert.True(vrai, "Erreur, vrai vaut false. Le test est volontairement mal écrit, corrigez le.");

        }

Effectivement, la variable « vrai » était initialisé à false.

## Etape 3 :

**Réponse :**

        public CurrencyHolder(string name, int capacity, int amount)

        {

            if(name == null || name.Length < 4 ){

                throw new System.ArgumentException("Parameter cannot be null", "name");

            }else if(name == ""){

                throw new System.ArgumentException("Parameter cannot be empty", "name");

            }else if(amount < 0){

                throw new System.ArgumentException("Parameter cannot be negative", "amount");

            }

J’ai ajouté les exceptions manquantes dans le constructeur de « CurrencyHolder ».

## Etape 4 :

**Réponse :**

        [Fact]

        public void TestPut10CurrencyInNonFullCurrencyHolder()

        {

*// A vous d'écrire un test qui vérifie que si on ajoute via la methode put 10 currency à un sac a moité plein, il contient maintenant la bonne quantité de currency*

            CurrencyHolder currency = new CurrencyHolder(EXEMPLE\_NOM\_MONNAIE\_VALIDE, EXEMPLE\_CAPACITE\_VALIDE/2, EXEMPLE\_CONTENANCE\_INITIALE\_VALIDE);

            currency.Put(10);

            Assert.True(currency.CurrentAmount == EXEMPLE\_CONTENANCE\_INITIALE\_VALIDE + 10);

        }

        [Fact]

        public void TestPut10CurrencyInNearlyFullCurrencyHolder()

        {

*// A vous d'écrire un test qui vérifie que si on ajoute via la methode put 10 currency à un sac quasiement plein, une exeption NotEnoughtSpaceInCurrencyHolderExeption est levée.*

            Action mauvaisAppel = () => {

                CurrencyHolder currency = new CurrencyHolder(EXEMPLE\_NOM\_MONNAIE\_VALIDE,10,9);

                currency.Put(10);

            };

            Assert.Throws<NotEnoughtSpaceInCurrencyHolderExeption>(mauvaisAppel);

        }

        [Fact]

        public void CreatingCurrencyHolderWithNameShorterThan4CharacterThrowExeption()

        {

*// A vous d'écrire un test qui doit échouer s'il est possible de créer un CurrencyHolder dont Le Nom De monnaie est inférieur a 4 lettres*

            Action mauvaisAppel = () => new CurrencyHolder("del", EXEMPLE\_CAPACITE\_VALIDE, EXEMPLE\_CONTENANCE\_INITIALE\_VALIDE);

            Assert.Throws<ArgumentException>(mauvaisAppel);

        }

        [Fact]

        public void WithdrawMoreThanCurrentAmountInCurrencyHolderThrowExeption()

        {

*// A vous d'écrire un test qui vérifie que retirer (methode withdraw) une quantité negative de currency leve une exeption CantWithDrawNegativeCurrencyAmountExeption.*

*// Asruce : dans ce cas prévu avant même de pouvoir compiler le test, vous allez être obligé de créer la classe CantWithDrawMoreThanCurrentAmountExeption (vous pouvez la mettre dans le meme fichier que CurrencyHolder)*

            Action mauvaisAppel = () => {

                CurrencyHolder currency = new CurrencyHolder(EXEMPLE\_NOM\_MONNAIE\_VALIDE,10,9);

                currency.Withdraw(-5);

            };

            Assert.Throws<CantWithDrawNegativeCurrencyAmountExeption>(mauvaisAppel);

        }

        [Fact]

        public void WithdrawMoreThanCurrentAmountInCurrencyHolderThrowExeption()

        {

*// A vous d'écrire un test qui vérifie que retirer (methode withdraw) une quantité negative de currency leve une exeption CantWithDrawNegativeCurrencyAmountExeption.*

*// Asruce : dans ce cas prévu avant même de pouvoir compiler le test, vous allez être obligé de créer la classe CantWithDrawMoreThanCurrentAmountExeption (vous pouvez la mettre dans le meme fichier que CurrencyHolder)*

            Action mauvaisAppel = () => {

                CurrencyHolder currency = new CurrencyHolder(EXEMPLE\_NOM\_MONNAIE\_VALIDE,10,9);

                currency.Withdraw(-5);

            };

            Assert.Throws<CantWithDrawNegativeCurrencyAmountExeption>(mauvaisAppel);

        }

J’ai rempli toutes les méthodes de tests. Pour que cela marche il faut maintenant modifier le code dans « CurrencyHolder ».

## Etape 5 :

**Réponse :**

        public void Store(int amount)

        {

            if(currentAmount + amount > Capacity){

                throw new NotEnoughtSpaceInCurrencyHolderExeption();

            }else{

                currentAmount = currentAmount + amount;

            }

        }

        public void Put (int amount)

        {

            if(currentAmount + amount > Capacity){

                throw new NotEnoughtSpaceInCurrencyHolderExeption();

            }else{

                currentAmount = currentAmount + amount;

            }

        }

        public void Withdraw(int amount)

        {

            if(amount < 0){

                throw new CantWithDrawNegativeCurrencyAmountExeption();

            }else{

                currentAmount = currentAmount - amount;

            }

        }

## Etape 6 :

### Un nom de currency doit faire entre 4 et 10 characteres :

else if(name.Length > 10){

                throw new System.ArgumentException("Parameter cannot be more than 10 characters", "name");

            }else if(name.Length < 4){

                throw new System.ArgumentException("Parameter cannot be less than 4 characters", "name");

            }

### Ecrivez un test pour un nom de douze caractères :

        public void CreatingCurrencyHolderWithBigNameThrowExeption()

        {

*// Test si un nom est supérieur à 10 charactères*

            Action mauvaisAppel = () => new CurrencyHolder("jesuisunlongname", EXEMPLE\_CAPACITE\_VALIDE, EXEMPLE\_CONTENANCE\_INITIALE\_VALIDE);

            Assert.Throws<ArgumentException>(mauvaisAppel);

        }

### On ne peux pas mettre (methode) put une quantité negative de currency dans un CurrencyHolder

       public void Put (int amount)

        {   if(amount < 0){

                throw new CantPutNegativeCurrencyAmountExeption();

            }else{

                if(currentAmount + amount > Capacity){

                    throw new NotEnoughtSpaceInCurrencyHolderExeption();

                }else{

                    currentAmount = currentAmount + amount;

                }

            }

        }

        public void PutNegativeCurrentAmountInCurrencyHolderThrowExeption(){

            Action mauvaisAppel = () => {

                CurrencyHolder currency = new CurrencyHolder(EXEMPLE\_NOM\_MONNAIE\_VALIDE,10,9);

                currency.Put(-5);

            };

            Assert.Throws<CantPutNegativeCurrencyAmountExeption>(mauvaisAppel);

        }

### On ne peux pas ajouter ou retirer 0 currency

        [Fact]

        public void PutNullCurrencyAmountExeption(){

            Action mauvaisAppel = () => {

                CurrencyHolder currency = new CurrencyHolder(EXEMPLE\_NOM\_MONNAIE\_VALIDE,10,9);

                currency.Put(0);

            };

            Assert.Throws<CantPutNullCurrencyAmountExeption>(mauvaisAppel);

        }

        [Fact]

        public void WithDrawNullCurrencyAmountExeption(){

            Action mauvaisAppel = () => {

                CurrencyHolder currency = new CurrencyHolder(EXEMPLE\_NOM\_MONNAIE\_VALIDE,10,9);

                currency.Withdraw(0);

            };

            Assert.Throws<CantWithDrawNullCurrencyAmountExeption>(mauvaisAppel);

        }

        public void Put (int amount)

        {   if(amount < 0){

                throw new CantPutNegativeCurrencyAmountExeption();

            }else if(amount == 0){

                    throw new CantPutNullCurrencyAmountExeption();

                }else{

                if(currentAmount + amount > Capacity){

                    throw new NotEnoughtSpaceInCurrencyHolderExeption();

                }else{

                    currentAmount = currentAmount + amount;

                }

            }

        }

        public void Withdraw(int amount)

        {

            if(amount < 0){

                throw new CantWithDrawNegativeCurrencyAmountExeption();

            }else if(amount == 0){

                throw new CantWithDrawNullCurrencyAmountExeption();

                }else{

                currentAmount = currentAmount - amount;

            }

        }

### Un nom de currency ne doit pas commencer par la lettre a majuscule ou minuscule (2 tests)

else if(name.Substring(0, 1).Equals("a") || name.Substring(0, 1).Equals("A")){

                throw new System.ArgumentException("Parameter cannot be a or A");

            }

        [Fact]

        public void IfStartACurrencyAmountException(){

            Action mauvaisAppel = () => new CurrencyHolder("acolyte", EXEMPLE\_CAPACITE\_VALIDE, EXEMPLE\_CONTENANCE\_INITIALE\_VALIDE);

            Assert.Throws<ArgumentException>(mauvaisAppel);

        }

        [Fact]

        public void IfStartaCurrencyAmountException(){

            Action mauvaisAppel = () => new CurrencyHolder("Acolyte", EXEMPLE\_CAPACITE\_VALIDE, EXEMPLE\_CONTENANCE\_INITIALE\_VALIDE);

            Assert.Throws<ArgumentException>(mauvaisAppel);

        }

### Un CurrencyHolder ne peux avoir une capacité inférieure à 1 (2 tests)

else if(capacity < 1){

                throw new System.ArgumentException("Parameter cannot be less than 1", "capacity");

            }

        [Fact]

        public void CreatingCurrencyHolderWithLessThanOneCapacityThrowExeption(){

            Action mauvaisAppel = () => new CurrencyHolder("Exemple", 0, EXEMPLE\_CONTENANCE\_INITIALE\_VALIDE);

            Assert.Throws<ArgumentException>(mauvaisAppel);

        }

        [Fact]

        public void CreatingCurrencyHolderWithMinimumOneCapacityThrowExeption(){

            CurrencyHolder currency = new CurrencyHolder("Exemple",1,EXEMPLE\_CONTENANCE\_INITIALE\_VALIDE);

        }

### Faire 2 tests unitaires pertinents pour la methode IsEmpty

        public bool IsEmpty()

        {

            if(CurrentAmount == 0){

            return true;

            }else{

                return false;

            }

        }

        public bool IsFull()

        {

            if(CurrentAmount == Capacity){

                return true;

            }else{

                return false;

            }

        }

        [Fact]

        public void CheckisNotEmpty(){

            CurrencyHolder currency = new CurrencyHolder("Exemple",1,0);

            currency.Put(1);

            var result = currency.IsEmpty();

            Assert.False(result);

        }

        [Fact]

        public void CheckisEmpty(){

            CurrencyHolder currency = new CurrencyHolder("Exemple",1,0);

            var result = currency.IsEmpty();

            Assert.True(result);

        }

### Un CurrencyHolder est plein (IsFull) si son contenu est égal à sa capacité (4 test, y a pas de piege)

        [Fact]

        public void CheckisFull(){

            CurrencyHolder currency = new CurrencyHolder("Exemple",1,0);

            currency.Put(1);

            var result = currency.IsFull();

            Assert.True(result);

        }

        [Fact]

        public void CheckisNotFull(){

            CurrencyHolder currency = new CurrencyHolder("Exemple",2,0);

            currency.Put(1);

            var result = currency.IsFull();

            Assert.False(result);

        }

        [Fact]

        public void CheckisFullInitial(){

            CurrencyHolder currency = new CurrencyHolder("Exemple",2,2);

            var result = currency.IsFull();

            Assert.True(result);

        }

        [Fact]

        public void CheckisNotFullInitial(){

            CurrencyHolder currency = new CurrencyHolder("Exemple",2,0);

            var result = currency.IsFull();

            Assert.False(result);

        }

# CONCLUSION

Ce TP m’a permis de mettre réellement en pratique la création de test à partir d’un code existant.

A présent je peux tester les exceptions.