

Rapport TP2 - Classe UDEV 2 - Zakaria NYA

1-

J'ai commencé par l'installation de dot.net sur ma machine MacOS et l'utilisation de l'outil de

```
using System;
using Xunit;

namespace dark_place_game.tests
{
    /// Cette classe contient tout un ensemble de tests unitaires pour la classe CurrencyHolder
    public class CurrencyHolderTest
    {
        public static readonly int EXEMPLE_CAPACITE_VALIDE = 2748;
        public static readonly int EXEMPLE_CONTENANCE_INITIALE_VALIDE = 978;
        public static readonly string EXEMPLE_NOM_MONNAIE_VALIDE = "Brouzouf";

        [Fact]
        public void VraiShouldBeTrue()
        {
            var vrai = true;
            Assert.True(vrai, "Erreur, vrai vaut false. Le test est volontairement mal écrit, corrigez le.");
        }

        [Fact]
        public void CurrencyHolderCreatedWithInitialCurrentAmountOf10ShouldContain10Currency()
        {
            var ch = new CurrencyHolder(EXEMPLE_NOM_MONNAIE_VALIDE, EXEMPLE_CAPACITE_VALIDE, 10);
            var result = ch.CurrentAmount == 10;
            Assert.True(result);
        }

        [Fact]
        public void CurrencyHolderCreatedWithInitialCurrentAmountOf25ShouldContain25Currency()
        {
            var ch = new CurrencyHolder(EXEMPLE_NOM_MONNAIE_VALIDE, EXEMPLE_CAPACITE_VALIDE, 25);
            var result = ch.CurrentAmount == 25;
            Assert.True(result);
        }

        [Fact]
        public void CurrencyHolderCreatedWithInitialCurrentAmountOf0ShouldContain0Currency()
        {
            var ch = new CurrencyHolder(EXEMPLE_NOM_MONNAIE_VALIDE, EXEMPLE_CAPACITE_VALIDE, 0);
            var result = ch.CurrentAmount == 0;
            Assert.True(result);
        }
    }
}
```

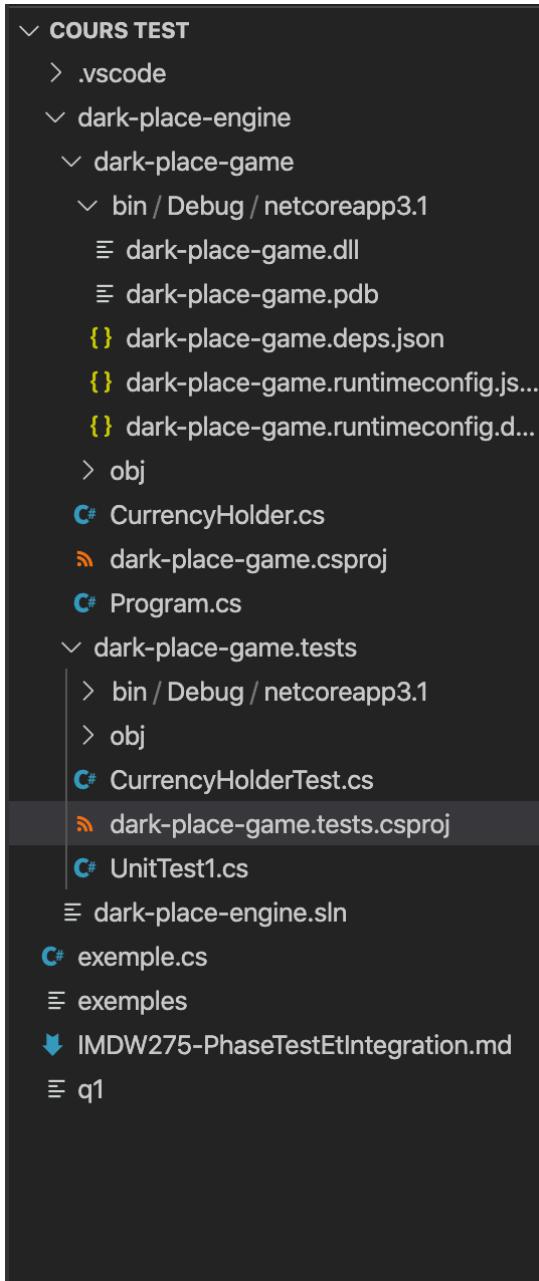
développement VScode que j'avais déjà installé.

Pour dot.net la version est netcoreapp3.1

Et pour les tests unitaires la version de XUnit est : 2.4.0

2- j'ai commencé par lancer les commandes suivante pour créer la solution et aussi une suite d'instructions pour aboutir à un projet sur lequel je pouvais faire mes tests unitaires.

- dotnet new sln -o dark-place-engine : pour créer une solution
- dotnet new console -o dark-place-game : cree le projet qui contient les sources du client du jeu
- dotnet sln add ./dark-place-game/dark-place-game.csproj : pour ajouter le projet à la solution
- dotnet new xunit -o dark-place-game.tests : pour creep un projet de tests unitaire
- dotnet add ./dark-place-game.tests/dark-place-game.tests.csproj reference ./dark-place-game/dark-place-game.csproj : On référence les sources du jeu dans le projet de tests unitaires
- dotnet sln add ./dark-place-game.tests/dark-place-game.tests.csproj : permet d'ajouter le projet de tests unitaire à la solution



Etape 1 : On commence par vérifier que tout fonctionne :
Il y a bien tous les fichiers dans le dossier que nous avons installé.

Les Etapes 1 étant faite on passe à l'étape 2 , étape 3 et étape 4 on lance les tests et on les fait :

CurrencyHolderTest.cs — cours test

```
暗黙の場所 > dark-place-game.tests > CurrencyHolderTest.cs > () dark-place-game.tests > dark-place-game.tests.CurrencyHolderTest > VraiShouldBeTrue()
```

```
1 using System;
2 using Xunit;
3
4 namespace dark_place_game.tests
5 {
6
7     /// Cette classe contient tout un ensemble de tests unitaires pour la classe CurrencyHolder
8     0 references
9     public class CurrencyHolderTest
10    {
11        13 references
12        public static readonly int EXEMPLE_CAPACITE_VALIDE = 2748;
13        5 references
14        public static readonly int EXEMPLE_CONTENANCE_INITIALE_VALIDE = 978;
15        7 references
16        public static readonly string EXEMPLE_NOM_MONNAIE_VALIDE = "Brouzouf";
17
18        [Fact]
19        0 references
20        public void VraiShouldBeTrue()
21        {
22            1 var vrai = true;
23            Assert.True(vrai, "Erreur, vrai vaut false. Le test est volontairement mal écrit, corrigez le.");
24        }
25
26        [Fact]
27        0 references
28        public void CurrencyHolderCreatedWithInitialCurrentAmountOf10ShouldContain10Currency()
29        {
30            1 var ch = new CurrencyHolder(EXEMPLE_NOM_MONNAIE_VALIDE, EXEMPLE_CAPACITE_VALIDE, 10);
31            var result = ch.CurrentAmount == 10;
32            Assert.True(result);
33        }
34
35        [Fact]
36        0 references
37        public void CurrencyHolderCreatedWithInitialCurrentAmountOf25ShouldContain25Currency()
38        {
39            1 var ch = new CurrencyHolder(EXEMPLE_NOM_MONNAIE_VALIDE, EXEMPLE_CAPACITE_VALIDE, 25);
40            var result = ch.CurrentAmount == 25;
41            Assert.True(result);
42        }
43    }

```

Ln 15, Col 39 Spaces: 4 UTF-8 LF C# ⌂

CurrencyHolderTest.cs — cours test

```
暗黙の場所 > dark-place-game.tests > CurrencyHolderTest.cs > () dark-place-game.tests > dark-place-game.tests.CurrencyHolderTest > VraiShouldBeTrue()
```

```
44
45
46        [Fact]
47        0 references
48        public void CreatingCurrencyHolderWithNegativeContentThrowException()
49        {
50            // Petite subtilité : pour tester les levées d'exception en c# on est obligé d'utiliser un concept un peu exotique : les
51            // sans entrer dans le détail pour déclarer une lambda respecte la syntaxe ci dessous, pour rédiger des tests unitaires
52            Action mauvaisAppel = () => {new CurrencyHolder(EXEMPLE_NOM_MONNAIE_VALIDE, EXEMPLE_CAPACITE_VALIDE, -10)};
53            Assert.Throws<ArgumentException>(mauvaisAppel);
54        }
55
56        [Fact]
57        0 references
58        public void CreatingCurrencyHolderWithNullNameThrowException()
59        {
60            // Petite subtilité : pour tester les levées d'exception en c# on est obligé d'utiliser un concept un peu exotique : les
61            // sans entrer dans le détail pour déclarer une lambda respecte la syntaxe ci dessous, pour rédiger des tests unitaires
62            Action mauvaisAppel = () => {new CurrencyHolder(null, EXEMPLE_CAPACITE_VALIDE, EXEMPLE_CONTENANCE_INITIALE_VALIDE)};
63            //Assert.NotNull(mauvaisAppel);
64            Assert.Throws<ArgumentException>(mauvaisAppel);
65        }
66
67        [Fact]
68        0 references
69        public void CreatingCurrencyHolderWithEmptyNameThrowException()
70        {
71            // Petite subtilité : pour tester les levées d'exception en c# on est obligé d'utiliser un concept un peu exotique : les
72            // sans entrer dans le détail pour déclarer une lambda respecte la syntaxe ci dessous, pour rédiger des tests unitaires
73            Action mauvaisAppel = () => {new CurrencyHolder("", EXEMPLE_CAPACITE_VALIDE, EXEMPLE_CONTENANCE_INITIALE_VALIDE)};
74            //Assert.NotEmpty(mauvaisAppel);
75            Assert.Throws<ArgumentException>(mauvaisAppel);
76        }
77
78        [Fact]
79        0 references
80        public void BrouzoufIsAValidCurrencyName() -
81
82
83        [Fact]
84        0 references
85        public void DollarIsAValidCurrencyName()
86        {
87            // A vous d'écrire un test qui vérifie qu'on peut créer un CurrencyHolder contenant une monnaie dont le nom est Dollar
88            var ch = new CurrencyHolder("Dollar", EXEMPLE_CAPACITE_VALIDE, 25);
89            var result = ch.CurrencyName == "Dollar";
90            Assert.True(result);
91        }
92
93        [Fact]
94        0 references
95        public void TestPut10CurrencyInNonFullCurrencyHolder() -
96
97
98
99
100
101
102
103    /**

```

Ln 15, Col 39 Spaces: 4 UTF-8 LF C# ⌂

```
dark-place-engine > dark-place-game.tests > CurrencyHolderTest.cs > {} dark_place_game.tests > dark_place_game.tests.CurrencyHolderTest > VraiShouldBeTrue()
```

```
103  /**
104   * [Fact]
105   * public void TestPut10CurrencyInNearlyFullCurrencyHolder()
106   {
107     // A vous d'écrire un test qui vérifie que si on ajoute via la méthode put 10
108     // currency à un sac quasiment plein, une exception NotEnoughSpaceInCurrencyHolderException est levée,
109
110     Action sacPlein = () => new CurrencyHolder(EXEMPLE_NOM_MONNAIE_VALIDE, 2748, 2750); 
111     Assert.Throws<ArgumentException>(sacPlein);
112   }
113
114  */
115  /**
116   * [Fact]
117   * 0 references
118   * public void CreatingCurrencyHolderWithNameShorterThan4CharacterThrowException()
119   {
120     // A vous d'écrire un test qui doit échouer s'il es possible de créer un
121     // CurrencyHolder dont Le Nom De monnaie est inférieur a 4 lettres
122     var ch = new CurrencyHolder("AZE",EXEMPLE_CAPACITE_VALIDE, 2748);
123     var nomCourt = ch.CurrencyName.Length < 4;
124     Assert.True(nomCourt, "nom trop court");
125   }
126
127  /**
128   * 0 references
129   * public void WithdrawMoreThanCurrentAmountInCurrencyHolderThrowException()
130   {
131     // A vous d'écrire un test qui vérifie que retirer (méthode withdraw)
132     // une quantité négative de currency lève une exception Can'tWithdrawNegativeCurrencyAmountException.
133     // Ainsi : dans ce cas prévu avant même de pouvoir compiler le test,
134     // vous allez être obligé de créer
135     // la classe Can'tWithdrawMoreThanCurrentAmountException
136     // (vous pouvez la mettre dans le même fichier que CurrencyHolder)
137     var ch = new CurrencyHolder(EXEMPLE_NOM_MONNAIE_VALIDE,EXEMPLE_CAPACITE_VALIDE , EXEMPLE_CONTENANCE_INITIALE_VALIDE);
138     Action mauvaisAppel = () => {ch.Withdraw(10)};
139     Assert.Throws<ArgumentException>(mauvaisAppel);
140   }
141
142  /**
143   * 0 references
144   * public void CreatingCurrencyHolderWithNameEqualTo12CharacterThrowException()
145   {
146     // A vous d'écrire un test qui doit échouer s'il es possible de créer un
147     // CurrencyHolder dont Le Nom De monnaie est inférieur a 4 lettres
148     var ch = new CurrencyHolder("AZE",EXEMPLE_CAPACITE_VALIDE, 2748);
149     var currencyName12 = ch.CurrencyName.Length == 12;
150     Assert.False(currencyName12, "le nom de la monnaie est égale à 12 caractères");
151   }

```

The screenshot shows a dark-themed instance of Visual Studio Code with the following details:

- Title Bar:** CurrencyHolderTest.cs — cours test
- Left Sidebar:** Shows file navigation with CurrencyHolderTest.cs as the active tab, and a list of files including CurrencyHolder.cs, dark-place-engine, dark-place-game.tests, and dark_place_game.tests.
- Code Editor:** Displays the following C# code for a unit test:

```
151 [Fact]
152 0 references
153 public void CreatingCurrencyHolderWithSimpleNameInRange4To10CharacterThrowException()
154 {
155     // A vous d'écrire un test qui doit échouer s'il est possible de créer un
156     // Currencyholder dont Le Nom De monnaie est inférieur à 4 lettres
157     var ch = new CurrencyHolder("AZERTY", EXEMPLE_CAPACITE_VALIDE, 2748);
158     var CurrencyName = ch.CurrencyName.Length;
159     Assert.InRange(CurrencyName, 4, 10);
160 }
161 }
162 }
163 }
```

- Right Sidebar:** Shows a detailed code analysis report with sections like "Issues", "CodeLens", "References", and "Metrics".
- Bottom Status Bar:** Shows the current file (Line 15, Column 39), character count (Spaces: 4), encoding (UTF-8), language (C#), and other status indicators.

On passe à l'étape 5, étape 6 et etape 7 on corrige le code de la classe CurrencyHolder

```
using System;
namespace dark_place_game
{
    [System.Serializable]
    /// Une Exception Custom
    public class NotEnoughSpaceInCurrencyHolderException : System.Exception {}

    public class CantWithDrawNegativeCurrencyAmountException : System.Exception {}

    public class CurrencyHolder
    {
        public static readonly string CURRENCY_DEFAULT_NAME = "Unnamed";

        // Le nom de la monnaie
        public string CurrencyName {
            get { return currencyName; }
            private set {
                currencyName = value;
            }
        }

        // Le champs interne de la property
        private string currencyName = CURRENCY_DEFAULT_NAME;

        // Le montant actuel
        public int CurrentAmount {
            get { return currentAmount; }
            private set {
                currentAmount = value;
            }
        }

        // Le champs interne de la property
        private int currentAmount = 0;

        // La contenance maximum du conteneur
        public int Capacity {
            get { return capacity; }
            private set {
                capacity = value;
            }
        }

        // Le champs interne de la property
        private int capacity = 0;
    }
}
```

```
dark-place-engine > dark-place-game > CurrencyHolder.cs > {} dark_place_game > dark_place_game.NotEnoughSpaceInCurrencyHolderException
}
// Le champs interne de la property
private int capacity = 0;

13 references
public CurrencyHolder(string name,int capacity, int amount) {
    Capacity = capacity;
    CurrencyName = name;
    CurrentAmount = amount;

    if(CurrencyName == null || CurrencyName == ""){
        throw new System.ArgumentException("Le nom ne peut pas être null", CurrencyName);
    }

    if(amount < 0){
        throw new ArgumentException("le porte monnaie ne peut pas être initialisé négativement");
    }

    if(CurrentAmount > Capacity){
        throw new NotEnoughSpaceInCurrencyHolderException();
    }
}

0 references
public bool IsEmpty() {
    return true;
}

0 references
public bool IsFull() {
    if(CurrentAmount == Capacity){
        return true;
    } else {
        return false;
    }
}

1 reference
public int Store(int amount) {
    return CurrentAmount + amount;
}

1 reference
public int Withdraw(int amount) {
    if(amount < 0 ){
        throw new CantWithDrawNegativeCurrencyAmountException();
    }
}
```

The screenshot shows the Microsoft Visual Studio IDE interface with the following details:

- Title Bar:** CurrencyHolder.cs — cours test
- Toolbars:** Standard Windows-style toolbar.
- Left Sidebar:** Shows icons for Solution Explorer, Task List, Properties, and Help.
- Central Area:** Code editor displaying `CurrencyHolder.cs`. The code implements a currency holder with properties for name, capacity, and current amount, and methods for storing and withdrawing currency.

```
dark-place-engine > dark-place-game > CurrencyHolder.cs <
```

```
41     }
42     // Le champs interne de la property
43     2 references
44     private int capacity = 0;
45
46     13 references
47     public CurrencyHolder(string name,int capacity, int amount) {
48         Capacity = capacity;
49         CurrencyName = name;
50         CurrentAmount = amount;
51
52         if(CurrencyName == null || CurrencyName == ""){
53             throw new System.ArgumentException("Le nom ne peut pas être null", CurrencyName);
54         }
55
56         if(amount < 0){
57             throw new ArgumentException("le porte monnaie ne peut pas être initialisé négativement");
58         }
59
60         if(CurrentAmount > Capacity){
61             throw new NotEnoughSpaceInCurrencyHolderException();
62         }
63
64     }
65
66     0 references
67     public bool IsEmpty() {
68         return true;
69     }
70
71     0 references
72     public bool IsFull() {
73         if(CurrentAmount == Capacity){
74             return true;
75         } else {
76             return false;
77         }
78     }
79
80     1 reference
81     public int Store(int amount) {
82         return CurrentAmount + amount;
83     }
84
85     1 reference
86     public int Withdraw(int amount) {
87         if(amount < 0 ){
88             throw new CantWithdrawNegativeCurrencyAmountException();
89         }

```

- Bottom Status Bar:** Shows the current line (Ln 7, Col 28), character count (Spaces: 4), and file type (UTF-8 LF C#).

En résumé, nous avons effectué une démarche de va et vient entre le code et les tests pour essayer de couvrir le plus de cas possible dans nos tests.
Nous avons voulu que ces tests jetent des exceptions si il faut, mais aussi qu'il nous retourne les possibilités d'erreur de code, ou d'erreur utilisateur.

Merci de votre lecture