TP DarkPlace Engine

Au cours de ce TP vous allez programmer le petit RPG textuel en c#.

Prérequis logiciels

Le TP se fait en c# en utilisant le framework open source .NET Core 3.1.100

J'ai arbitrairement choisie la bibliothèque de tests xUnit afin de gagner du temps par rapport à l'écriture de tests manuels uniquements et parcequ'elle est nativement incluse dans .net core.

Techniquement vous pouvez utiliser l'IDE de votre choix voire travailler sous notepad si ca vous amuse. Néanmoins si vous souhaitez un support je vous encourage à utiliser le même environnement que moi : VisualStudio Code qui est un editeur open source proposé par Microsoft pour le language C#.

Compte tenu de la lourdeur des modifications de code, il est vivement recommandé d'utiliser git. Téléchargez le avant de faire la suite des installation pour que VS Code l'intègre automatiquement à son interface.

Commencez par télécharger le SDK pour framework .NET Core 3.1 et installez le.

Pour ceux qui ne sont pas sous windows64.

Ensuite téléchargez et installez VS Code.

Initialisation du projet

A la main

Pour initialiser le projet de zero voici les instructions à suivre, je suis parti du tutorial officiel de Microsoft;

- Ouvrir avec VScode le répertoire dans lequel on veut travailler.
- Allez dans le terminal (en bas de l'éditeur par défaut)
- Tapez les commandes suivantes :
- Création de la solution :

dotnet new sln -o dark-place-engine

- On rentre dans le répertoire de la solution

cd dark-place-engine

- On crée le projet qui vas contenir les sources du client du jeu (en utilisant le template projet console)

dotnet new console -o dark-place-game

- On ajoute ce projet à la solution

dotnet sln add ./dark-place-game/dark-place-game.csproj

- On crée un projet de tests unitaire

dotnet new xunit -o dark-place-game.tests

- On référence les sources du jeu dans le projet de tests unitaires

dotnet add ./dark-place-game.tests/dark-place-game.tests.csproj reference ./dark-place-game/dark-place-game.csproj

- On ajoute le projet de tests unitaires à la solution

dotnet sln add ./dark-place-game.tests/dark-place-game.tests.csproj

- Créez dans le sous-répertoire dark-game-place un fichier CurrencyHolder.cs. Mettez y le contenu suivant :

```
using System;

namespace dark_place_game
{

    [System.Serializable]
    /// Une Exeption Custom
    public class NotEnoughtSpaceInCurrencyHolderExeption : 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;

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

        public bool IsEmpty() {
            return true;
```

```
        }

        public bool IsFull() {
            return true;
        }

        public void Store(int amount) {

        }

        public void Withdraw(int amount) {

        }
    }
}
```

- Dans le même répertoire vous devriez trouver le fichier Program.cs qui contient la méthode Main, point d'entrée du projet. Remplacer son contenu par le suivant :

```
using System;

namespace dark_place_game
{
    class Program
    {
        private GameWorld gameWorld;

        static void Main(string[] args)
        {
            Console.WriteLine("Lancement du jeu Dark Place ...");
            throw new Exception("Le jeu ne semble pas encore codé !");
        }
    }

}
```

- Créez dans le sous répertoire dark-place-game.tests le fichier CurrencyHolderTest.cs. Mettez y le contenu suivant :

```
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 = false;
            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);
        }

        [Fact]
        public void CreatingCurrencyHolderWithNegativeContentThrowExeption()
        {
            // Petite subtilité : pour tester les levées d'exeption en c# on est obligé d'utiliser un concept un
            // sans entrer dans le détail pour déclarer une lambda respectez la syntaxe ci dessous, pour rédiger
            Action mauvaisAppel = () => new CurrencyHolder(EXEMPLE_NOM_MONNAIE_VALIDE,EXEMPLE_CAPACITE_VALIDE , -
            Assert.Throws<ArgumentException>(mauvaisAppel);
```

```
}

[Fact]
public void CreatingCurrencyHolderWithNullNameThrowExeption()
{
    // Petite subtilité : pour tester les levées d'exeption en c# on est obligé d'utiliser un concept un
    // sans entrer dans le détail pour déclarer une lambda respectez la syntaxe ci dessous, pour rédiger
    Action mauvaisAppel = () => new CurrencyHolder(null,EXEMPLE_CAPACITE_VALIDE , EXEMPLE_CONTENANCE_INIT
    Assert.Throws<ArgumentException>(mauvaisAppel);
}

[Fact]
public void CreatingCurrencyHolderWithEmptyNameThrowExeption()
{
    // Petite subtilité : pour tester les levées d'exeption en c# on est obligé d'utiliser un concept un
    // sans entrer dans le détail pour déclarer une lambda respectez la syntaxe ci dessous, pour rédiger
    Action mauvaisAppel = () => new CurrencyHolder("",EXEMPLE_CAPACITE_VALIDE , EXEMPLE_CONTENANCE_INITIA
    Assert.Throws<ArgumentException>(mauvaisAppel);
}

/** #TODO_ETAPE_4
[Fact]
public void BrouzoufIsAValidCurrencyName ()
{
    // A vous d'écrire un test qui vérife qu'on peut créer un CurrencyHolder contenant une monnaie dont I
}

[Fact]
public void DollardIsAValidCurrencyName ()
{
    // A vous d'écrire un test qui vérife qu'on peut créer un CurrencyHolder contenant une monnaie dont I
}

[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 moi
}

[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 quasi
}

[Fact]
public void CreatingCurrencyHolderWithNameShorterThan4CharacterThrowExeption()
{
    // A vous d'écrire un test qui doit échouer s'il es possible de créer un CurrencyHolder dont Le Nom D
}
```

```
        [Fact]
        public void WithdrawMoreThanCurrentAmountInCurrencyHolderThrowExeption()
        {
            // A vous d'écrire un test qui vérifie que retirer (methode withdraw) une quantité negative de currer
            // Asruce : dans ce cas prévu avant même de pouvoir compiler le test, vous allez être obligé de créer
        }
        #TODO_ETAPE_4 */

    }
}
```

- Il y a potentielement un fichier exemple de tests qui traine, supprimez le.

Conditions de rendu

Il faudra envoyer par mail le répertoire de la solution et un compte rendu. Envoyez par mail à gael.fenetgarde@campus-igs-toulouse.fr un dossier avec un compte rendu de tp et le dossier avec votre solution finale ce soir avant minuit.

Partie 1

Etape 1

On commence par vérifier que tout fonctionne :

- Choisissez un répertoire et faites l'initialisation à la main.
- Ouvrez VS Code, enregistrez en tant que Workspace le répertoire contenant la solition puis dans la barre du haut de l'éditeur cliquez sur l'onglet Terminal puis choisissez l'option New Terminal.
- Un Terminal s'est ouvert en bas de la fenêtre de code, n'hésitez pas à l'agrandir un peu si necessaire pour y voir.
- Vous devriez voir un résultat de ce style :

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS F:\IPI_TP\test\dark-place-engine>
```

- tapez ensuite la commande 'ls' pour vérifier que votre répertoire contient bien la solution :

```
PS F:\IPI_TP\test\dark-place-engine> ls


    Répertoire : F:\IPI_TP\test\dark-place-engine


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----       10/12/2019     20:06                .vscode
d-----       10/12/2019     20:15                dark-place-game
d-----       10/12/2019     20:26                dark-place-game.tests
-a----       10/12/2019     18:54           1752 dark-place-engine.sln
```

- Si vous ne voyez pas ce résultat vous n'êtes pas dans le bon répertoire ou avez mal récupéré le projet.
- Ensuite nous allons tester l'execution du programme et l'execution des tests. Tapez d'abord la commande "dotnet run --project .\dark-place-game" pour essayer d'executer le projet. Vous devriez voir quelque chose comme ca :

```
PS F:\IPI_TP\test\dark-place-engine> dotnet run --project .\dark-place-game\
Lancement du jeu Dark Place ...

Unhandled Exception: System.Exception: Le jeu ne semble pas encore codé !
   at dark_place_game.Program.Main(String[] args) in F:\IPI_TP\test\dark-place-engine\dark-place-game\Program.cs:
PS F:\IPI_TP\test\dark-place-engine>
```

- Ensuite executez les tests de l'application en tapant dans le terminal la commande "dotnet test .\dark-place-game.tests". Vous devriez voir un résultat de ce style :

```
PS F:\IPI_TP\test\dark-place-engine> dotnet test  .\dark-place-game.tests\
Série de tests pour F:\IPI_TP\test\dark-place-engine\dark-place-game.tests\bin\Debug\netcoreapp2.2\dark-place-gam
Outil en ligne de commande d'exécution de tests Microsoft (R), version 16.3.0
Copyright (c) Microsoft Corporation. Tous droits réservés.

Démarrage de l'exécution de tests, patientez...

Au total, 1 fichiers de test ont correspondu au modèle spécifié.

[xUnit.net 00:00:00.42]     dark_place_game.tests.CurrencyHolderTest.Test1 [FAIL]

[xUnit.net 00:00:00.43]     dark_place_game.tests.CurrencyHolderTest.Test2 [FAIL]

[xUnit.net 00:00:00.43]     dark_place_game.tests.CurrencyHolderTest.Test4 [FAIL]

  X dark_place_game.tests.CurrencyHolderTest.Test1 [9ms]
  Message d'erreur :
   Un porte monnaie crée avec un contenu initial de 10 doit contenir 10
Expected: True
Actual:   False
  Arborescence des appels de procédure :
     at dark_place_game.tests.CurrencyHolderTest.Test1() in F:\IPI_TP\test\dark-place-engine\dark-place-game.test

  X dark_place_game.tests.CurrencyHolderTest.Test2 [1ms]
  Message d'erreur :
   Un porte monnaie crée avec un contenu initial de 25 doit contenir 25
Expected: True
Actual:   False
  Arborescence des appels de procédure :
     at dark_place_game.tests.CurrencyHolderTest.Test2() in F:\IPI_TP\test\dark-place-engine\dark-place-game.test

  X dark_place_game.tests.CurrencyHolderTest.Test4 [6ms]
  Message d'erreur :
   Assert.Throws() Failure
Expected: typeof(System.ArgumentException)
Actual:   (No exception was thrown)
  Arborescence des appels de procédure :
     at dark_place_game.tests.CurrencyHolderTest.Test4() in F:\IPI_TP\test\dark-place-engine\dark-place-game.test


Échec de la série de tests.
Nombre total de tests : 4
     Réussi(s) : 1
     Non réussi(s) : 3
 Durée totale : 0,9356 Secondes
```

- Il est normal que la plupart des tests échouent à ce stade. Anecdotiquement certains tests réussissent par accident.

Etape 2

- Afin de vous préparer à la rédactions de vos propres test plus tard, le test0 est volontairement mal codé. Corrigez le. A part si le contraire est indiqué dans l'exercice, les tests que j'ai rédigés pour la suite du TP sont toujours valides.
- Lisez la trace de l'echec du test dans le terminal, trouver la ligne à modifier, et corrigez l'erreur évidente. Relancez le test,

Etape 3

- Vous allez pouvoir profiter des tests pour coder CurrencyHolder. Modifiez le code de CurrencyHolder jusqu'à ce que tous les tests passent.

Etape 4

- Trouvez le bloc de code commenté avec le tag #TODO_ETAPE_4.
- Décommentez le
- Remplissez les tests vides. Executez les.

Etape 5

- Corrigez le code de CurrencyHolder jusqu'a faire passer vos tests. (en cas de doute sur vos tests demandez à vérifier qu'ils soient valides)

Etape 6

- Voici quelques spécifications et ou/description de tests. Implantez ces tests dans le code.
- Un nom de currency doit faire entre 4 et 10 characteres :
- Ecrivez un test pour un nom de douze caracteres
- On ne peux pas mettre (methode) put une quantité negative de currency dans un CurrencyHolder
- On ne peux pas mettre des currency dans un CurrencyHolder si ca le fait depasser sa capacité
- On ne peux pas ajouter ou retirer 0 currency (lever expetion) (2 tests)
- Un nom de currency ne doit pas commencer par la lettre a majuscule ou minuscule (2 tests)
- Un CurrencyHolder ne peux avoir une capacité inférieure à 1 (2 tests)
- Faire 2 tests unitaires pertinents pour la methode IsEmpty
- Un CurrencyHolder est plein (IsFull) si son contenu est égal à sa capacité (4 test, y a pas de piege)

Etape 7

- Corrigez le code de CurrencyHolder jusqu'a faire passer vos tests. (en cas de doute sur vos tests demandez à vérifier qu'ils soient valides)

Etape 8

- Félicitation Vous avez terminé l'exercice sur les tests unitaires. Demandez l'accès à la partie 2.

Faites une copie archivée de votre travail.