DELPHES JABRI CDEV SOPRA



**TP 1 : Les tests**

**Module IMDW275**

**Sommaire :**

Table des matières

[I) INTRODUCTION 3](#_Toc27578119)

[***II)*** ***Partie 1 : Test Unitaire (pseudoCode)*** 4](#_Toc27578120)

[Question 1 : INT SOUSTRAIRE (INT A, INT B) 4](#_Toc27578121)

[Question 2 : Est-il possible de couvrir tous les cas de figure ? Si oui comment, si non pourquoi ? 4](#_Toc27578122)

[Question 3 : additionner (A,B) 5](#_Toc27578123)

[Question 4 : diviser (A,B) 5](#_Toc27578124)

[Question 4a : 5](#_Toc27578125)

[Question 4b : Quels est la particularité de cette méthode par rapport aux précédentes ? 5](#_Toc27578126)

[Question 4c : Cela la rend elle moins testable ? Si oui pourquoi ? Sinon comment la tester ? 5](#_Toc27578127)

[Question 5 : Multiplier(A,B) 6](#_Toc27578128)

[Question 6 : 6](#_Toc27578129)

[Question 6b : 6](#_Toc27578130)

[Question 6 c : 7](#_Toc27578131)

[Question 6d : 7](#_Toc27578132)

[Question 7 : Un site Web Comporte 3 pages, HOME, PAGE 1, PAGE 2. Lorsque une mauvaise url sur le bon domaine est entrée, le navigateur est redirigé vers la page HOME. 8](#_Toc27578133)

[Question 7a : 8](#_Toc27578134)

[Question 8 : 9](#_Toc27578135)

[Question 9 : 10](#_Toc27578136)

[Question 10 : 13](#_Toc27578137)

[Question 11 : 13](#_Toc27578138)

[Question 12 : 15](#_Toc27578139)

[Question 13 : 16](#_Toc27578140)

[Question 14 : 18](#_Toc27578141)

[III) CONCLUSION 19](#_Toc27578142)

# INTRODUCTION

Durant ce TP nous allons voir comment créer des tests unitaires.

Premièrement, en créant des pseudos code et puis en second nous allons créer des tests unitaires en C# avec l’éditeur de code VSCode.

# ***Partie 1 : Test Unitaire (pseudoCode)***

## Question 1 : INT SOUSTRAIRE (INT A, INT B)

**Réponse :**

1-Vérifier (Soustraire(10,5) == 5)

2-Vérifier(Soustraire(0,5)== -5)

3-Vérifier(Soustraire(0,0) == 0)

4-Vérifier(Soustraire(vide,0) == null)

5-Vérifier(Soustraire(0,vide) == null)

6-Vérifier(Soustraire(vide,vide) == null)

7-Vérifier(Soustraire(-1,5) == 4)

8-Vérifier(Soustraire(-1,-2) == -3)

9-Vérifier(Soustraire(2,-20) == -18)

10-Vérifier(Soustraire(4200,4200) == 0)

11-Vérifier(Soustraire(4500,8000) == -3500)

12-Vérifier(Soustraire(-10000,-10000) == 0)

13-Vérifier(Soustraire(1,-1) == 2)

14-Vérifier(Soustraire(-1,1) == -2)

15-Vérifier(Soustraire(-1,5 , -1,5) == null)

16-Vérifier(Soustraire(-1,5 , 1,5) == null)

17-Vérifier(Soustraire(a<b) == négatif)

18-Vérifier(Soustraire(a>b) == positif)

19-Vérifier(Soustraire(100000 , 200000) == -100000)

20-Vérifier(Soustraire(200000 , 100000) == 100000)

21-Vérifier(Soustraire(-200000 , -200000) == 0)

22-Vérifier(Soustraire(Négatif<Négatif) == positif)

23-Vérifier(Soustraire(Positif<Négatif) == positif)

24-Vérifier(Soustraire(Négatif<Positif) == Négatif)

25-Vérifier(Soustraire(355136516351361, 1) == 355136516351360)

## Question 2 : Est-il possible de couvrir tous les cas de figure ? Si oui comment, si non pourquoi ?

**Réponse :**

Non il n’est pas possible de couvrir tous les cas de figures parce qu’il y a un nombre de cas possible proche de l’infini.

## Question 3 : additionner (A,B)

**Réponse :**

1-Vérifier(Additioner(10,10) == 20)

2-Vérifier(Additioner(0,0) == 0)

3-Vérifier(Additioner(-5,6) == 1)

4-Vérifier(Additioner(-5,-5) == -10)

5-Vérifier(Additioner(5, -5) == 10)

## Question 4 : diviser (A,B)

### Question 4a :

**Réponse :**

1-Vérifier(Diviser(10,10) == 1)

2-Vérifier(Diviser(10,15) == 0)

3-Vérifier(Diviser(15,10) == 1)

4-Vérifier(Diviser(-10,10) == -1)

### Question 4b : Quels est la particularité de cette méthode par rapport aux précédentes ?

**Réponse :**

La particularité de cette méthode par rapport aux précédentes est qu’elle s’appuie sur les principes de la division. Il est impossible de diviser par 0.

### Question 4c : Cela la rend elle moins testable ? Si oui pourquoi ? Sinon comment la tester ?

**Réponse :**

Non elle est toujours testable. Il faut juste garantir que les paramètres sont différents de 0.

## Question 5 : Multiplier(A,B)

**Réponse :**

1-Verifier(Multiplier(5, 5) == 25)

2-Verifier(Multiplier(-5, 5) == -25)

3-Verifier(Multiplier(5, -5) == -25)

4-Verifier(Multiplier(-5, -5) == 25)

5-Verifier(Multiplier(5, 0) == 0)

6-Verifier(Multiplier(0, 5) == 0)

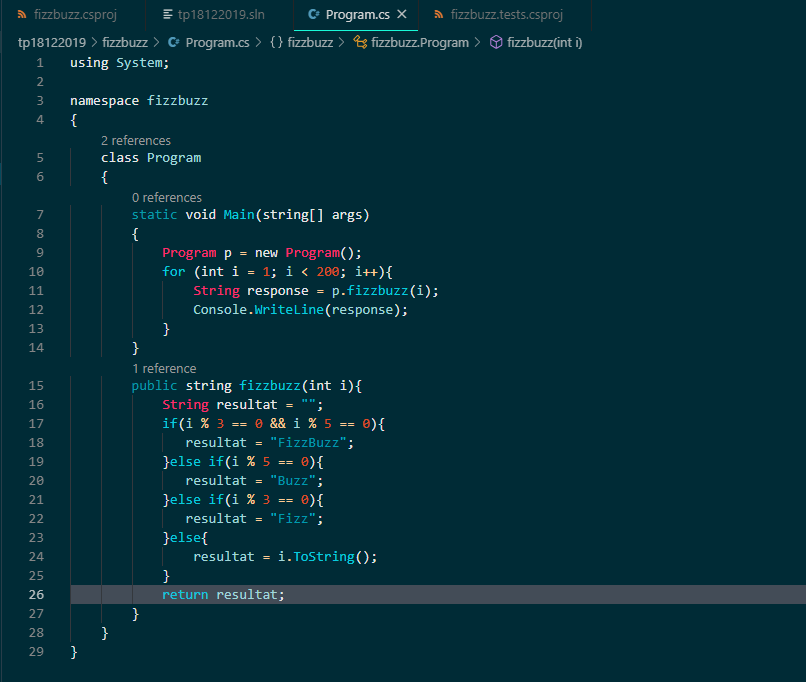
7-Verifier(Multiplier(5, 1) == 5)

8-Verifier(Multiplier(1 ,5) == 5)

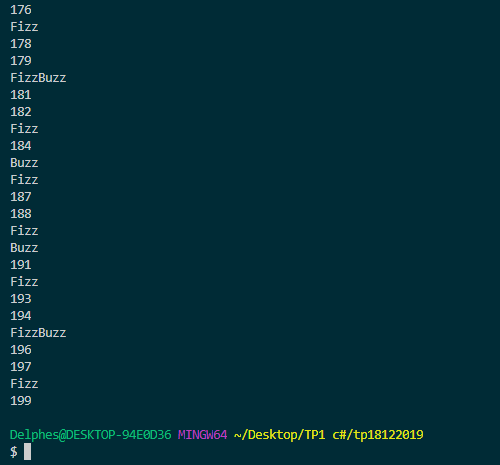
## Question 6 :

### Question 6b :

**Réponse :**



**Justification :**



### Question 6 c :

Vérifier(fizzbuzz(i) <200) : test pour le paramètre d’entré soit inférieur à 200.

Vérifier (fizzbuzz(i) > 0) : Test pour le paramètre d’entré soit supérieur à 0.

Vérifier(fizzbuzz(i % 3) == 0) : Test pour les nombres multiple de 3 doit afficher « Fizz »

Vérifier(fizzbuzz(i % 5) ==0) : Test pour les nombres multiple de 5 doit afficher « Buzz »

Vérifier(fizzbuzz(i % 5 == 0 && i % 3 == 0) : test pour les nombre multiple de 3 et 5 doit afficher « Fizzbuzz ».

### Question 6d :

Oui on peut tous les nombres entre 1 et 199.

## Question 7 : Un site Web Comporte 3 pages, HOME, PAGE 1, PAGE 2. Lorsque une mauvaise url sur le bon domaine est entrée, le navigateur est redirigé vers la page HOME.

### Question 7a :

Vérifier(lien home to page1 == page1)

Vérifier(lien home to page2 == page2)

Vérifier(lien page1 to page2 == page2)

Vérifier(lien page1 to home == home)

Vérifier(lien page2 to page1 == page1)

Vérifier(lien page2 to home == home)

Question 7b :

Vérifier(lien home to errorURL == home)

Vérifier(lien page1 to errorURL == home) renvoie vers home si errorURL

Vérifier(lien page2 to errorURL == home).

## Question 8 :

**Réponse :**

public class Person{

private int age;

    public Person(int pAge){

        this.age = pAge;

    }

    public int getAge(){

        return this.age;

    }

    public void setAge(int pAge){

        this.age = pAge;

    }

    public void vieillir (int pAge){

        this.age = this.age + pAge;

    }

}

**Justification :**

        [Fact]

        public void vieillirPersonnePos()

        {

            Person p = new Person(0);

            p.vieillir(5);

            Assert.True(p.getAge()  == 5);

        }

        [Fact]

        public void vieillirPersonneNull()

        {

            Person p = new Person(0);

            p.vieillir(0);

            Assert.True(p.getAge()  == 0);

        }

## Question 9 :

**Réponse :**

public class Elfe{

private int age;

private bool etreEnStase;

    public Elfe(){

        this.age = 0;

        this.etreEnStase = false;

    }

    public int getAge(){

        return this.age;

    }

    public void setAge(int pAge){

        this.age = pAge;

    }

    public bool getEtreEnStase(){

        return this.etreEnStase;

    }

    public void setEtreEnStase(bool pEtreEnStase){

        this.etreEnStase = pEtreEnStase;

    }

    public void vieillir (int pAge){

        if(!this.etreEnStase && this.age + pAge <= 999){

        this.age = this.age + pAge;

        }

        if(!this.etreEnStase && this.age == 999){

            this.etreEnStase = true;

        }

    }

    public void rajeunir (int pAge){

        if(!this.etreEnStase && this.age - pAge >= 0){

            this.age = this.age - pAge;

        }

    }

    public void EntrerEnStase(){

        this.etreEnStase = true;

    }

    public void SortirDeStase (){

        if(this.etreEnStase == true && this.age < 999){

            this.etreEnStase = false;

        }

    }

}

**Justification :**

        [Fact]

        public void AgeInitial (){

            Elfe elfe = new Elfe();

            Assert.True(elfe.getAge() == 0);

        }

        [Fact]

        public void EtreEnStaseInitial(){

            Elfe elfe = new Elfe();

            Assert.True(elfe.getEtreEnStase() == false);

        }

        [Fact]

        public void viellir(){

            Elfe elfe = new Elfe();

            elfe.vieillir(50);

            Assert.True(elfe.getAge() == 50);

        }

        [Fact]

        public void viellirMax(){

            Elfe elfe = new Elfe();

            elfe.vieillir(999);

            elfe.vieillir(1);

            Assert.True(elfe.getAge() == 999);

        }

        [Fact]

        public void RajeunirMin(){

            Elfe elfe = new Elfe();

            elfe.rajeunir(50);

            Assert.True(elfe.getAge() == 0);

        }

        [Fact]

        public void EtreEnStase(){

            Elfe elfe = new Elfe();

            elfe.vieillir(999);

            Assert.True(elfe.getEtreEnStase() == true);

        }

        [Fact]

        public void EtreEnStaseFalse(){

            Elfe elfe = new Elfe();

            elfe.vieillir(998);

            Assert.True(elfe.getEtreEnStase() == false);

        }

        [Fact]

        public void EntrerEnStase(){

            Elfe elfe = new Elfe();

            elfe.vieillir(2);

            elfe.EntrerEnStase();

            Assert.True(elfe.getEtreEnStase() == true);

        }

        [Fact]

        public void SortirDeStase(){

            Elfe elfe = new Elfe();

            elfe.vieillir(998);

            elfe.EntrerEnStase();

            elfe.SortirDeStase();

            Assert.True(elfe.getEtreEnStase() == false);

        }

        [Fact]

        public void SortirDeStaseError(){

            Elfe elfe = new Elfe();

            elfe.vieillir(999);

            elfe.SortirDeStase();

            Assert.True(elfe.getEtreEnStase() == true);

        }

    }

}

## Question 10 :

**Réponse :**

Le jeu choisit Counter Strike Global Offensive :

Si le joueur se fait tirer dessus ou se prend une grenade ou saute d’une grande hauteur, il perd de la vie.

Si le joueur n’a plus de balle dans le chargeur, il doit recharger et s’il n’a plus de balle dans le chargeur et n’a plus de balle sur lui, il ne peut plus tirer avec cette arme.

Si le joueur se fait tirer dessus, il est ralenti.

Si la vie du jour est égale à 0 il meurt.

## Question 11 :

**Réponse :**

        public int additionner (int i, int j){

            return i + j;

        }

        public int soustraire (int i, int j){

            return i - j;

        }

        public int diviser (int i, int j){

            return i / j;

        }

**Justification :**

        [Fact]

        public void AddtionnerPositif()

        {

            Program p = new Program();

            int resultat = p.additionner(1,1);

            Assert.True(resultat  == 2);

        }

        [Fact]

        public void AddtionnerDeuxZero()

        {

            Program p = new Program();

            int resultat = p.additionner(0,0);

            Assert.True(resultat  == 0);

        }

        [Fact]

        public void AddtionnerDeuxNegatif()

        {

            Program p = new Program();

            int resultat = p.additionner(-5,-5);

            Assert.True(resultat  == -10);

        }

        [Fact]

        public void AddtionnerUnNegUnPos()

        {

            Program p = new Program();

            int resultat = p.additionner(5,-5);

            Assert.True(resultat  == 0);

        }

        [Fact]

        public void SoustraireUnNegUnPos()

        {

            Program p = new Program();

            int resultat = p.soustraire(5,-5);

            Assert.True(resultat  == 10);

        }

        [Fact]

        public void SoustraireDeuxZero()

        {

            Program p = new Program();

            int resultat = p.soustraire(0,0);

            Assert.True(resultat  == 0);

        }

        [Fact]

        public void SoustraireDeuxPos()

        {

            Program p = new Program();

            int resultat = p.soustraire(10,15);

            Assert.True(resultat  == -5);

        }

        [Fact]

        public void SoustraireDeuxNeg()

        {

            Program p = new Program();

            int resultat = p.soustraire(-5,-10);

            Assert.True(resultat  == 5);

        }

        [Fact]

        public void DiviserDeuxPos()

        {

            Program p = new Program();

            int resultat = p.diviser(10,10);

            Assert.True(resultat  == 1);

        }

        [Fact]

        public void DiviserDeuxNeg()

        {

            Program p = new Program();

            int resultat = p.diviser(-5,-6);

            Assert.True(resultat  == 0);

        }

        [Fact]

        public void DiviserZeroParPos()

        {

            Program p = new Program();

            int resultat = p.diviser(0,5);

            Assert.True(resultat  == 0);

        }

## Question 12 :

**Réponse :**

public class Person{

private int age;

    public Person(int pAge){

        this.age = pAge;

    }

    public int getAge(){

        return this.age;

    }

    public void setAge(int pAge){

        this.age = pAge;

    }

    public void vieillir (int pAge){

        this.age = this.age + pAge;

    }

}

**Justification :**

        [Fact]

        public void vieillirPersonnePos()

        {

            Person p = new Person(0);

            p.vieillir(5);

            Assert.True(p.getAge()  == 5);

        }

        [Fact]

        public void vieillirPersonneNull()

        {

            Person p = new Person(0);

            p.vieillir(0);

            Assert.True(p.getAge()  == 0);

        }

## Question 13 :

**Réponse :**

public class Elfe{

private int age;

private bool etreEnStase;

    public Elfe(){

        this.age = 0;

        this.etreEnStase = false;

    }

    public int getAge(){

        return this.age;

    }

    public void setAge(int pAge){

        this.age = pAge;

    }

    public bool getEtreEnStase(){

        return this.etreEnStase;

    }

    public void setEtreEnStase(bool pEtreEnStase){

        this.etreEnStase = pEtreEnStase;

    }

    public void vieillir (int pAge){

        if(!this.etreEnStase && this.age + pAge <= 999){

        this.age = this.age + pAge;

        }

        if(!this.etreEnStase && this.age == 999){

            this.etreEnStase = true;

        }

    }

    public void rajeunir (int pAge){

        if(!this.etreEnStase && this.age - pAge >= 0){

            this.age = this.age - pAge;

        }

    }

    public void EntrerEnStase(){

        this.etreEnStase = true;

    }

    public void SortirDeStase (){

        if(this.etreEnStase == true && this.age < 999){

            this.etreEnStase = false;

        }

    }

}

**Justification :**

        [Fact]

        public void AgeInitial (){

            Elfe elfe = new Elfe();

            Assert.True(elfe.getAge() == 0);

        }

        [Fact]

        public void EtreEnStaseInitial(){

            Elfe elfe = new Elfe();

            Assert.True(elfe.getEtreEnStase() == false);

        }

        [Fact]

        public void viellir(){

            Elfe elfe = new Elfe();

            elfe.vieillir(50);

            Assert.True(elfe.getAge() == 50);

        }

        [Fact]

        public void viellirMax(){

            Elfe elfe = new Elfe();

            elfe.vieillir(999);

            elfe.vieillir(1);

            Assert.True(elfe.getAge() == 999);

        }

        [Fact]

        public void RajeunirMin(){

            Elfe elfe = new Elfe();

            elfe.rajeunir(50);

            Assert.True(elfe.getAge() == 0);

        }

        [Fact]

        public void EtreEnStase(){

            Elfe elfe = new Elfe();

            elfe.vieillir(999);

            Assert.True(elfe.getEtreEnStase() == true);

        }

        [Fact]

        public void EtreEnStaseFalse(){

            Elfe elfe = new Elfe();

            elfe.vieillir(998);

            Assert.True(elfe.getEtreEnStase() == false);

        }

        [Fact]

        public void EntrerEnStase(){

            Elfe elfe = new Elfe();

            elfe.vieillir(2);

            elfe.EntrerEnStase();

            Assert.True(elfe.getEtreEnStase() == true);

        }

        [Fact]

        public void SortirDeStase(){

            Elfe elfe = new Elfe();

            elfe.vieillir(998);

            elfe.EntrerEnStase();

            elfe.SortirDeStase();

            Assert.True(elfe.getEtreEnStase() == false);

        }

        [Fact]

        public void SortirDeStaseError(){

            Elfe elfe = new Elfe();

            elfe.vieillir(999);

            elfe.SortirDeStase();

            Assert.True(elfe.getEtreEnStase() == true);

        }

    }

}

## Question 14 :

L’oracle est une question à se poser par rapport aux attentes du test.

# CONCLUSION

Ce TP m’a permis de comprendre l’aspect théorique d’un test ainsi que comment réaliser des tests avec le langage C#. Le début est un peu flou mais finalement avec la pratique on comprend bien mieux.