BTS SIO
Bloc 2 – TP1 - suite
Programmation C#

Prestations soins

Vous allez devoir apporter quelques nouvelles fonctionnalités au projet Soins2021

Evolution 1

En tant que responsable du projet, je souhaite organiser mes tests.
Vous considèrerez que la partie codée dans la méthode Main est le premier test à effectuer.

En tant que développeur junior, vous décidez
- ✓ de créer une classe statique Traitement qui contiendra les méthodes de tests de vos classes
- ✓ de rajouter dans cette classe la méthode *TesteDossier* qui va afficher les éléments d'un dossier (celui que vous avez créé dans la première partie de ce projet)

```
static class Traitement
{
    internal static void TesteDossier()
    {
        Dossier dossier = new Dossier("Robert", "Jean", new DateTime(1980, 12, 3));
        dossier.AjoutePrestation("Libelle P3", new DateTime(2015, 9, 10, 12, 0, 0), new Intervenant("Dupont", "Jean"));
        dossier.AjoutePrestation("Libelle P1", new DateTime(2015, 9, 1, 12, 0, 0), new IntervenantExterne("Durand", "Annie", "Cardiologue", "Marseille", "0202020202"));
        dossier.AjoutePrestation("Libelle P2", new DateTime(2015, 9, 8, 12, 0, 0), new IntervenantExterne("Sainz", "Olivier", "Radiologue", "Toulon", "0303030303"));
        dossier.AjoutePrestation("Libelle P4", new DateTime(2015, 9, 20, 12, 0, 0), new Intervenant("Maurin", "Joëlle"));
        dossier.AjoutePrestation("Libelle P6", new DateTime(2015, 9, 8, 9, 0, 0), new Intervenant("Blanchard", "Michel"));
        dossier.AjoutePrestation("Libelle P5", new DateTime(2015, 9, 10, 6, 0, 0), new Intervenant("Tournier", "Hélène"));

        Console.WriteLine(dossier);
        Console.WriteLine("Nombre de Jours de soins V1 : " + dossier.GetNbJoursSoins());
        Console.WriteLine("Nombre de Jours de soins V2 : " + dossier.GetNbJoursSoinsV2());
        Console.WriteLine("Nombre de Jours de soins V3 : " + dossier.GetNbJoursSoinsV3());
        Console.WriteLine("Nombre de soins externes : " + dossier.GetNbPrestationsExternes());
        Console.WriteLine("-------------------------\n\n");
    }
```

Evolution 2

En tant que responsable du projet, je veux que mon application puisse afficher le nombre de prestations des intervenants.

En tant que développeur junior, vous décidez :
- ✓ De créer la méthode *GetNbPrestations* dans la classe *Intervenant*

```
,
/// <summary>
/// Retourne le nombre de prestations effectuées par l'intervenant
/// </summary>
/// <returns>nombre d'interventions de l'intervenant</returns>
public int GetNbPrestations()
{
    return this.prestations.Count;
}
```

- ✓ De créer la méthode TesteGetNbPrestationsI qui testera le nombre d'interventions effectuées par un intervenant

```
internal static int TesteGetNbPrestationsI()
{
    // intervenant 1 (2 prestations)
    Intervenant intervenant = new Intervenant("Dupont", "Pierre");

    intervenant.AjoutePrestation(new Prestation("Presta 10", new DateTime(2019, 6, 12), intervenant));
    intervenant.AjoutePrestation(new Prestation("Presta 11", new DateTime(2019, 6, 15), intervenant));
    return intervenant.GetNbPrestations();
}
```

- ✓ De créer la méthode *TesteGetNbPrestationsIE* qui testera le nombre d'interventions effectuées par un intervenant externe.

```
/// <summary>
/// Ce test montre que si l'intervenant est un intervenant externe, c'est la méthode publique GetNbPrestations
/// de la class Intervenant (classe mère) qui est exécutée.
/// </summary>
/// <returns></returns>
internal static int TesteGetNbPrestationsIE()
{
    // intervenant 2 type externe (3 prestations)
    IntervenantExterne intervenantExterne = new IntervenantExterne("Terrature", "Julie", "Cardiologue", "Toulon", "0000112233");
    intervenantExterne.AjoutePrestation(new Prestation("Presta 20", new DateTime(2019, 6, 12), intervenantExterne));
    intervenantExterne.AjoutePrestation(new Prestation("Presta 21", new DateTime(2019, 6, 14), intervenantExterne));
    intervenantExterne.AjoutePrestation(new Prestation("Presta 22", new DateTime(2019, 7, 18), intervenantExterne));
    return intervenantExterne.GetNbPrestations();

}
```

Vous devriez obtenir ceci :

```
 -----Fin dossier--------------
Nb Prestations Intervenant : 2
Nb Prestations Intervenant Externe: 3
```

Avec les appels suivants dans la méthode Main() :

```
Console.WriteLine("Nb Prestations Intervenant : " + Traitement.TesteGetNbPrestationsI());
Console.WriteLine("Nb Prestations Intervenant Externe: " + Traitement.TesteGetNbPrestationsIE());
Console.ReadKey();
```

Evolution 3

En tant que responsable de projet, je veux conserver la date de création d'un dossier afin de pouvoir effectuer des statistiques

En tant que développeur junior, vous décidez :

- ✓ De créer un attribut dateCreation dans la classe Dossier, cet attribut ne sera pas modifiable depuis l'application.
- ✓ De valoriser automatiquement cet attribut dans le constructeur avec la date système.

Revue de code : Il faudrait surcharger le constructeur :

- ✓ *Sans la date de création de dossier : la date de création du dossier est initialisée avec la date duu jour*
- ✓ *Avec la date de création de dossier si on veut pouvoir créer un dossier par exemple avec des dossiers que l'on récupérerait d'une base de données*

Vous aurez à modifier et surcharger également les constructeurs qui permettent de créer un dossier avec une ou des prestations

Evolution 4

En tant que responsable de projet, je veux effectuer des contrôles sur les dates afin de pouvoir gérer les erreurs de saisie.

En tant que développeur junior, vous décidez :
De tester :

- ✓ La date de prestation qui doit être postérieure à la date de création du dossier auquel elle est affectée

Le contrôle doit se faire dans la méthode AjoutePrestation pour pouvoir comparer la date de prestation avec le date de création du dossier.

- ✓ La date de prestation qui est inférieure à la date courante car votre application ne gère que les prestations réellement effectuées.

Prestation

- ✓ La date de naissance du dossier qui ne peut pas être supérieure à la date du jour.

Le contrôle se fera dans le constructeur de lacclasse Dossier

- ✓ De créer une classe SoinsException qui hérite da la classe Exception, dont l'espace de noms est Soins2021.Exceptions

Pour l'instant cette classe n'aura que le constructeur :

```
public Soins2021Exception(string message) : base(message)
{
}
```

- ✓ De déclencher une exception de type SoinsException dans le cas d'une date non conforme.

Exemple :

```
throw new Soins2021Exception("La date de création du dossier nepeut être postèrieure à la date du jour");
```

- ✓ D'intercepter ces exceptions dans les méthodes de test (exemple : *TesteDossier*)

L'idée ici est d'intercepter, dans chaque méthode de test, les exceptions de type Soins2021Exception. Les autres exceptions vont se propager jusqu'à la méthode Main.
Par exemple :

```
internal static void TesteDossier()
{
    try
    {

    }
    catch (Soins2021Exception ex)
    {

    }
}
```

Cela permet au programme et notamment aux tests de se poursuivre après une erreur applicative.

- ✓ D'intercepter toutes les autres exceptions dans la méthode Main.

Toutes les autres exceptions sont inhérentes au système et sont attrapées et gérées dans la méthode Main :

```
static void Main(string[] args)
{
    try
    {

    }
    catch(Exception ex)
    {

    }
    Console.ReadKey();
}
```

Evolution 5

En tant que responsable du projet je veux pouvoir journaliser les exceptions SoinsException dans un fichier au format texte.

En tant que développeur junior, vous devez proposer une solution et la mettre en œuvre.

Revue de code :
Solution 1 : fichier texte non formatté : facile à mettre en œuvre, peu lisible
Solution 2 : fichier texte au format JSON : pas de grande difficulté, données structurées et formattées. Il existe de nombreuses bibliothèques pour utiliser les fichiers JSON
Solution 3 : fichier texte au format XML : Un peu plus compliqué à mettre en œuvre, très verbeux

Solution retenue : fichier Json.
Données à récupérer pour chaque exception :

- ✓ *Le nom de l'application,*
- ✓ *La classe d'exception (ici , Soins2021Exception)*
- ✓ *La date de l'exception,*
- ✓ *Le message de l'exception,*
- ✓ *Le nom de l'utilisateur qui a déclenché l'exception*
- ✓ *Le nom de la machine sur laquelle l'exception s'est déclenchée.(exemple : CP-SIO-W10-PG10)*

L'idée est de placer les exceptions dans un tableau JSON.

Début de tableau

```
[
  {
    "Application": "Soins2021",
    "ClasseException": "Soins2021.Exceptions.Soins2021Exception",
    "DateException": "2021-09-20T23:00:40.5929877+02:00",
    "MessageException": "Teste l'exceptionV2",
    "UserException": "roche",
    "UserMachine": "CP-SIO-W10-PG10"
  },
  {
    "Application": "Soins2021",
    "ClasseException": "Soins2021.Exceptions.Soins2021Exception",
    "DateException": "2021-09-20T23:01:19.6843571+02:00",
    "MessageException": "Teste l'exceptionV2",
    "UserException": "roche",
    "UserMachine": "CP-SIO-W10-PG10"
  },
  {
    "ClasseException": "Soins2021.Exceptions.Soins2021Exception",
    "DateException": "2021-09-22T23:31:17.2193342+02:00",
    "MessageException": "La date de création du dossier nepeut être postèrieure à la date du jour",
    "UserException": "roche",
    "UserMachine": "CP-SIO-W10-PG10"
  }
]
```

Une exception

Fin de tableau

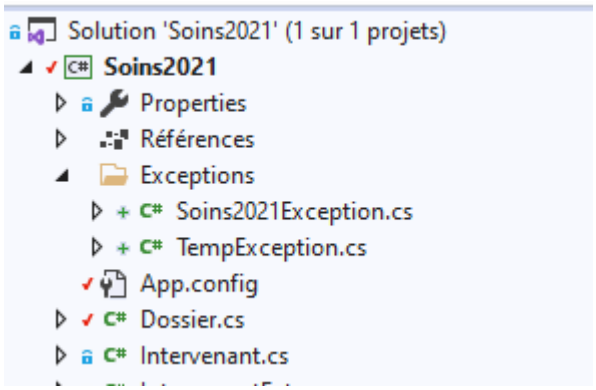
La démarche à suivre :

On peut facilement créer un objet JSON à partir d'un objet d'une classe. C'est vrai en C# mais aussi en PHP, Java,… l'action de convertir un objet d'une classe en objet Json s'appelle la ***sérialisation****. L'action inverse s'appelle la* ***désérialisation****.*
Vous aurez donc besoin :

- ✓ *D'une classe dont les attributs publics seront ceux que vous voulez voir apparaitre dans votre fichier Json,*

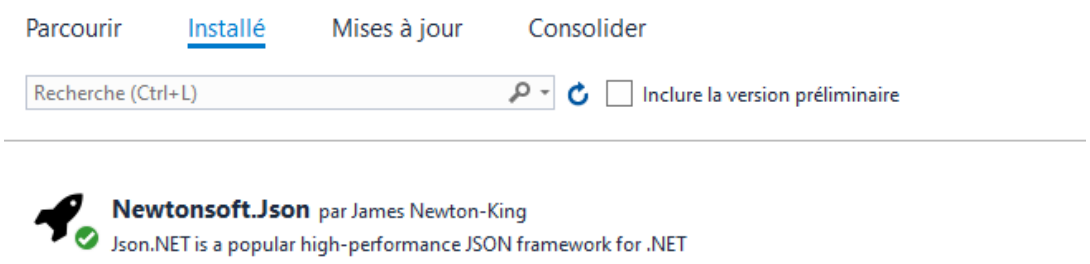
- ✓ *D'une bibliothèque qui vous expose les méthodes nécessaires pour interagir avec les fichiers Json.*

La classe : dans le dossier Exceptions qui contient la classe Soins2021Exception :

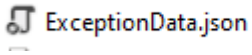



```
namespace Soins2021.Exceptions
{
    4 références | 0 modification | 0 auteur, 0 modification
    internal class TempException
    {
        1 référence | 0 modification | 0 auteur, 0 modification
        public String Application { get; set; }
        1 référence | 0 modification | 0 auteur, 0 modification
        public String ClasseException { get; set; }
        1 référence | 0 modification | 0 auteur, 0 modification
        public DateTime DateException { get; set; }
        1 référence | 0 modification | 0 auteur, 0 modification
        public String MessageException { get; set; }
        1 référence | 0 modification | 0 auteur, 0 modification
        public String UserException { get; set; }
        1 référence | 0 modification | 0 auteur, 0 modification
        public String UserMachine { get; set; }
    }
}
```

La bibliothèque : Vous utiliserez la bibliothèque Newtonsoft.Json que vous installerez depuis NuGet.



Vous allez créer un fichier ExceptionData.json vide à l'extérieur du dossier de votre application.



Vous allez stocker son chemin dans le fichier de configuration App.config :

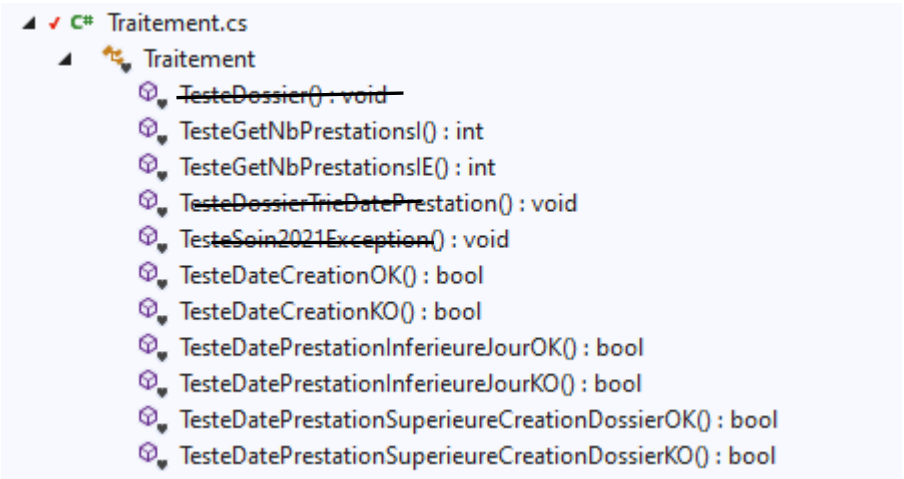
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
    <appSettings>
        <add key="pathToLogFile" value="T:'                                                    :\ExceptionData.json"/>
    </appSettings>
</configuration>
```

Vous devrez écrire dans le fichier ExceptionData.json chaque fois que vous déclencherez un exception du type Soins2021Exception … donc chaque fois que vous instancierez un objet de la classe Soins2021Exception (throw new Soins2021Exception(…))

Vous pourrez donc, dans le constructeur de cette classe, effectuer les opérations liées au journal de logs de l'application à savoir :

- ✓ *Instancier un objet de la classe TempException en valorisant ses attributs : Application,….*
- ✓ *Récupérer le chemin du fichier ExceptionData.json,*
- ✓ *Récupérer le contenu du fichier json dans une variable de type chaine de caractères.*
- ✓ *Désérialiser les objets contenus dans la chaine de caractères dans une liste d'objets de la classe TempException (voir bibliothèque Newtonsoft.Json)*
- ✓ *Ajouter dans cette liste l'objet TempException créé plus haut*
- ✓ *Sérialiser la liste d'objets dans une variable*
- ✓ *Ecrire dans le fichier (voir bibliothèque Newtonsoft.Json)*

Les méthodes de tests à écrire dans la classe Traitement :



Exemple 1 :

```
internal static bool TesteDateCreationOK()
{
    try {
        Dossier dossier = new Dossier("Dupont", "Jean", new DateTime(1990,11,12) , new DateTime(2019, 3, 31));
        return true;
    }catch (Soins2021Exception )
    {
       return false;
    }

}
```

Cette méthode doit retourner vrai car la date de création du dossier est inférieure à la date du jour. Si elle retourne faux, c'est qu'il y a un bug dans le code de la méthode.

Exemple 2 :

```
internal static bool TesteDateCreationKO()
{
    try
    {
        Dossier dossier = new Dossier("Dupont", "Jean", new DateTime(1990, 11, 12), new DateTime(2030, 3, 31));
        return false;
    }
    catch (Soins2021Exception ex)
    {
        return true;
    }

}
```

Cette méthode doit retourner vrai car la date de création du dossier est postérieure à la date du jour, elle a donc déclenché une exception de type Soins2021Exception. Si elle retourne faux, c'est qu'il y a un bug dans le code de la méthode.

Vous utiliserez cette logique pour les autres méthodes.

Il ne vous reste plus qu'à lancer l'exécution des tests

- ✓ *depuis la méthode Main*

ou

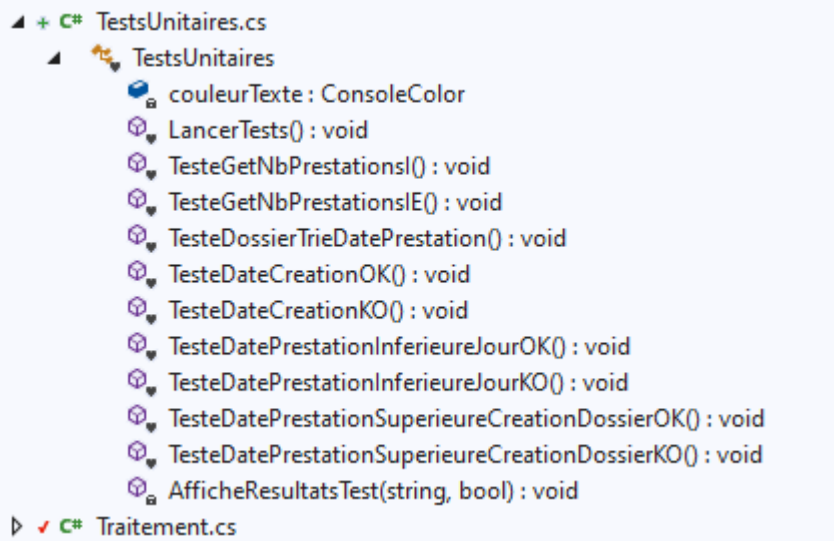
- ✓ *depuis la méthode Main qui appelle une méthode de la classe Traitement*

Résultat :

```
Nombre de jours de prestations (prestations triées sur date) : 4
Nb Prestations Intervenant : 2
Nb Prestations Intervenant Externe: 3
TesteDateCreationOK : True
TesteDateCreationKO : True
TesteDatePrestationInferieureJourOK : True
La date de la prestation ne peut être postérieure à la date courante
TesteDatePrestationInferieureJourKO : True
TesteDatePrestationSuperieureCreationDossierOK : True
La date des soins ne peut être antérieure à la date de création du dossier.
TesteDatePrestationSuperieureCreationDossierKO : True
-------------------------------------------------
```

Amélioration possible

Vous pouvez aussi créer une classe TestsUnitaires qui contient la méthode LancerTests qui lance l'exécution des tests :




```
1 référence | 0 modification | 0 auteur, 0 modification
internal static void LancerTests()
{
    Console.WriteLine("------------------------------------------------");
    Console.WriteLine(" --------  Début des Tests  --------------------");
    Console.WriteLine("------------------------------------------------");
    try
    {
        TestsUnitaires.TesteGetNbPrestationsI();
        TestsUnitaires.TesteGetNbPrestationsIE();
        TestsUnitaires.AfficheResultatsTest("Essai de méthode ", false);
        TestsUnitaires.TesteDateCreationOK();
        TestsUnitaires.TesteDateCreationKO();
        TestsUnitaires.TesteDatePrestationInferieureJourOK();
        TestsUnitaires.TesteDatePrestationSuperieureCreationDossierOK();
        TestsUnitaires.TesteDatePrestationSuperieureCreationDossierKO();
    }
    catch (Exception)
    {
        Console.WriteLine("Erreur à l'exécution des tests");
    }
    finally
    {
        Console.WriteLine("------------------------------------------------");
        Console.WriteLine(" --------  Fin des Tests  --------------------");
        Console.WriteLine("------------------------------------------------");
    }
}
```

Résultat de l'exécution :

```
------------------------------------------------
 --------  Début des Tests  --------------------
------------------------------------------------
TesteGetNbPrestationsI : True
TesteGetNbPrestationsIE : True
Essai de méthode  : False
TesteDateCreationOK : True
TesteDateCreationKO : True
TesteDatePrestationInferieureJourOK : True
TesteDatePrestationSuperieureCreationDossierOK : True
TesteDatePrestationSuperieureCreationDossierKO : True
------------------------------------------------
 --------  Fin des Tests  --------------------
------------------------------------------------
```

Vous remarquerez que

- ✓ Vous affichez le nom des méthodes testées (dynamiquement),
- ✓ Le résultat du test : true ou false
- ✓ La couleur du texte est verte quand le test est réussi,
- ✓ La couleur est rouge quand le test est échoué

… prenez du plaisir à développer !

Bon courage