1 TESTER UN PALINDROME

La mise en pratique suivante permet de comprendre le fonctionnement des tests unitaires.

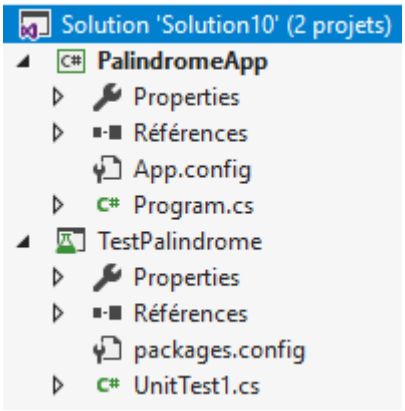
1.1 CRÉATION D'UNE NOUVELLE SOLUTION

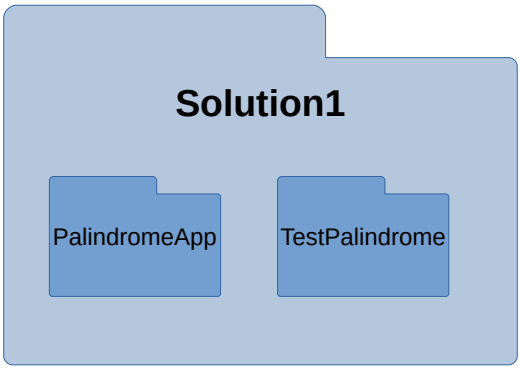
Vous allez tester une fonction de vérification de palindrome (phrase inversée qui contient les mêmes lettres dans un sens comme dans l'autre) et la corriger.

Avant de commencer, voici la structure du projet. La solution implique 3 phases :

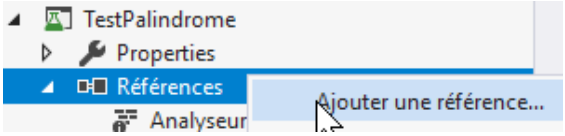
1. Création d'un nouveau projet Microsoft : **Autres types de projets** > **Solutions Visual Studio**
2. Création d'un sous projet **Visual C#** > **Application console (.NET framework)** appelé *PalindromeApp*
3. Création d'un sous projet **Visual C#** > Test > **Projet de test unitaire (.NET Framework)** nommé *TestPalindrome*, avec une référence au projet *PalindromeApp*.

Votre projet aura alors la structure suivante :

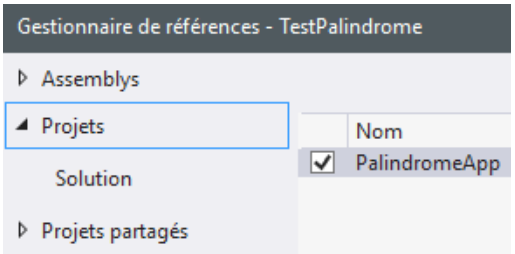




Pour ajouter une référence à un projet, il suffit de faire un clic droit sur le mot "Références" du projet et choisir "Ajouter une référence".



Il suffit alors de cocher le projet *PalindromeApp* dans la partie **Projets** (et pas dans assemblys)



Dans le projet **PalindromeApp**, vous modifierez le fichier **program.cs** comme suit :

```
namespace PalindromeApp {
    public class Palindrome {
        private String chaineNormale;
        private String chaineOriginale;
        private String chaineInversee;

        public Palindrome(String Z)
        {
            this.chaineOriginale = Z;
        }
        public Boolean verifierPalindrome(String chaine) {
            // écrivez votre code ici et ajouter les "return" corrects
            return true;
        }
    }

    static class Program {
        // Exécution du Programme
        static void Main(string[] args)
        {
            String laChaine = "";
            Console.WriteLine("Saisissez une phrase :");
            laChaine = Console.ReadLine();
            Palindrome monPal = new Palindrome();
            if (monPal.verifierPalindrome(laChaine)) {
                Console.WriteLine("c'est un palindrome");
            } else {
                Console.WriteLine("ce n'est pas un palindrome");
            }
            Console.ReadKey();
        }
    }
}
```

Dans le projet **TestPalindrome** il faut ajouter l'importation de la classe Palindrome :

```
using PalindromeApp;
```

et modifier le code comme suit :

```
namespace TestPalindrome {
    [TestClass]
    public class UnitTest1 {
        [TestMethod]
        public void TestPalindromeSimple()
        {
```

```
            Palindrome lepalindrome = new Palindrome();
        }
    }
}
```

S'il n'y a pas d'erreur, vous pouvez continuer, sinon, relisez les instructions ci-dessus.

1.2 CRÉATION DES TESTS

Nous voulons vérifier les princpaux cas de fonctionnement de la fonction palindrome :

1. Cas simple : tout est en minuscules et uniquement des caractères alphabétique
2. Cas évolué : il peut y avoir des minuscules et majuscules
3. Cas complexe n°1 : il y a des caractères accentués
4. Cas complexe n°2 : il y a des caractères de ponctuation à supprimer
5. Cas en erreur : si la phrase n'est pas un palindrome mais que la fonction affirme l'inverse

Pour chaque cas, nous devons envisager toutes les solutions que l'utilisateur pourrait saisir.

Au final, notre fonction devra pouvoir accepter les palindromes suivants :

	VRAI	FAUX
Cas simple	bob anna	
Cas évolué	Bob AnNa	
Cas complexe n°1	RéussiràParissuer	
Cas complexe n°2	Réussir à Paris : suer !	
Cas faux		B0B

Voici également quelques exemples tirés du web :

A Cuba, Anna a bu ça

Bon sport, trop snob

Engage le jeu, que je le gagne

Et si l'arôme des bott es révèle ma déviante et naïve dame, le verset t'obsède, moraliste

Réussir à Paris : suer

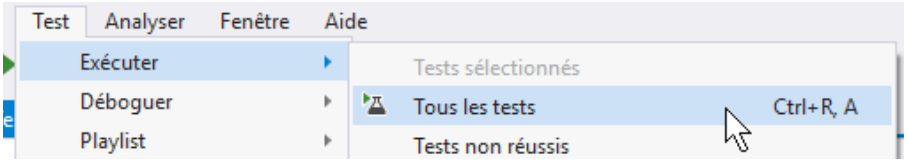
En créant plusieurs méthodes de tests, nous créons une matrice de tests que notre fonction devra réussir.

Voici maintenant le code pour le projet **TestPalindrome** (fichier **UnitTest1.cs**)

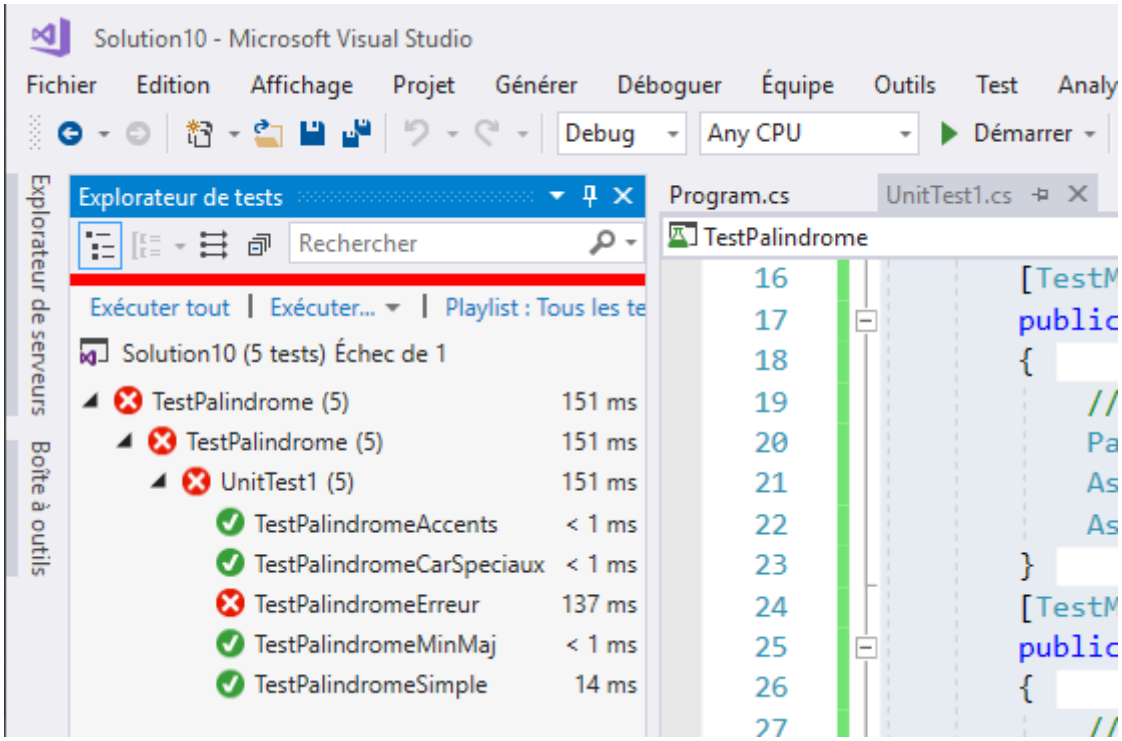
```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using PalindromeApp;

namespace TestPalindrome {
    [TestClass]
    public class UnitTest1 {
        [TestMethod]
        public void TestPalindromeSimple()
        {
            // vérification que la fonction passe le test simple
            Palindrome lepalindrome = new Palindrome();
            Assert.IsTrue(lepalindrome.verifierPalindrome("bob"));
            Assert.IsTrue(lepalindrome.verifierPalindrome("BOB"));
        }
        [TestMethod]
        public void TestPalindromeMinMaj()
        {
            // vérification que la fonction passe le test simple
            Palindrome lepalindrome = new Palindrome();
            Assert.IsTrue(lepalindrome.verifierPalindrome("Bob"));
            Assert.IsTrue(lepalindrome.verifierPalindrome("AnNa"));
        }
        [TestMethod]
        public void TestPalindromeAccents()
        {
            // vérification que la fonction passe le test des accents
            Palindrome lepalindrome = new Palindrome();
            Assert.IsTrue(lepalindrome.verifierPalindrome("RéussiràParissuer"));
        }
        [TestMethod]
        public void TestPalindromeCarSpeciaux()
        {
            // vérification que la fonction passe le test des caractères à supprimer
            Palindrome lepalindrome = new Palindrome();
            Assert.IsTrue(lepalindrome.verifierPalindrome("Reussir a Paris : suer"));
        }
        [TestMethod]
        public void TestPalindromeErreur()
        {
            Palindrome lepalindrome = new Palindrome();
            Assert.IsFalse(lepalindrome.verifierPalindrome("annna"));
            Assert.IsFalse(lepalindrome.verifierPalindrome("B0B"));
            Assert.IsFalse(lepalindrome.verifierPalindrome("Pas un palindrome"));
        }
    }
}
```

Actuellement, la fonction verifierPalindrome(chaine) renvoie systématiquement true.

Voici le résultat des tests (menu Test > Exécuter > Tous les tests)



On constate que même si 4 tests semblent réussir, la méthode TestPalindromeErreur est fausse.



Votre travail consiste maintenant à corriger la fonction verifierPalindrome() afin qu'elle réussisse tous les tests.

Vous savez désormais utiliser les tests unitaires en C#.

Pour d'autres langages il faudra parfois intégrer des librairies particulières, comme c'est le cas en Java, avec la bibliothèque JUnit.