SynapInfo

Dictionnaire, Framework de test, classe générique

Matthieu PLANCHOT

Dans l’étude de cas SynapInfo, il y a la présence d’un dictionnaire qui est en fait un classe générique : public class Dictionary<TKey,TValue>, il en existe d’autre comme les listes. Dans le Framework .net, les classes génériques sont surtout présentes dans l’espace de nom System.Collections.Generic

# Classe Mission

class Mission

{

private string nom;

private string description;

private Intervenant executant;

private Dictionary<DateTime, double> relevéHoraire = new Dictionary<DateTime, double>();

public Mission() { }

public Mission(string nom, string description, Intervenant executant, DateTime date) {

}

public Dictionary<DateTime, double> getRelevéHoraire()

{

return relevéHoraire;

}

public Intervenant getExecutant()

{

return executant;

}

public void ajoutReleve(DateTime jour, double nbHeures)

{

if (relevéHoraire.ContainsKey(jour)) // la fonction CountainsKey(key) retourne un booléen la clé est présente dans le dictionnaire

nbHeures = nbHeures + relevéHoraire[jour]; // accès de la valeur grâce à la clé du dictionnaire

else if (nbHeures <= 8)

relevéHoraire.Add(jour, nbHeures); // méthode Add de la classe générique du dictionnaire

else

relevéHoraire.Add(jour, 8);

}

public int nbHeuresEffectuées()

{

double nbHeuresEffectuees = 0;

for (int i = 0; i < relevéHoraire.Count; i++) // propriété Count hérité de ICollection, compte le nombre d’élément de la collection

{

nbHeuresEffectuees += relevéHoraire.Values.ElementAt(i); // propriété qui retourne les valeurs dans une collection et la méthode ElementAt(index) permet de parcourir cette liste grâce à indice

}

return Convert.ToInt32(nbHeuresEffectuees);

}

}

# Classe Intervenant

class Intervenant

{

private string nom;

private double tauxHoraire;

public Intervenant(){ }

public Intervenant(string nom, double tauxHoraire) {

this.nom = nom;

this.tauxHoraire = tauxHoraire;

}

public double getTauxHoraire()

{

return tauxHoraire;

}

}

# Classe Projet

class Projet {

private string nom;

private DateTime debut;

private DateTime fin;

private List<Mission> missions = new List<Mission>();

private double prixFacturéMO;

public Projet() { }

public Projet(string nom, DateTime debut, DateTime fin) {

this.nom = nom;

this.debut = debut;

this.fin = fin;

}

public double cumulCoûtMO()

{

double cumul = 0;

Intervenant intervenant = new Intervenant();

for (int i = 0; i <= missions.Count; i++)

{

cumul += missions.ElementAt(i).nbHeuresEffectuées();

}

return cumul;

}

public double margeBruteCourante()

{

return prixFacturéMO - cumulCoûtMO();

}

}

# Debug.Assert

Vérifie une condition ; si la condition est false, affiche des messages et une boîte de message qui montre la pile des appels.

class Program

{

static void Main(string[] args)

{

Intervenant intervenant = new Intervenant("nicolas", 300);

Mission laMission = new Mission("Appolo","programmation", intervenant, new DateTime(2007, 4, 4));

DateTime laDate1, laDate2, laDate;

double nbHeures;

laDate1 = new DateTime(2007, 4, 4);

laMission.ajoutReleve(laDate1, 5);

nbHeures = laMission.getRelevéHoraire()[laDate1];

Debug.Assert(nbHeures == 5, "non conforme");

laDate2 = new DateTime(2007, 4, 6);

laMission.ajoutReleve(laDate2, 9);

nbHeures = laMission.getRelevéHoraire()[laDate2];

Debug.Assert(nbHeures == 8, "non conforme");

laDate = new DateTime(2007, 4, 5);

laMission.ajoutReleve(laDate, 6);

laMission.ajoutReleve(laDate, 5);

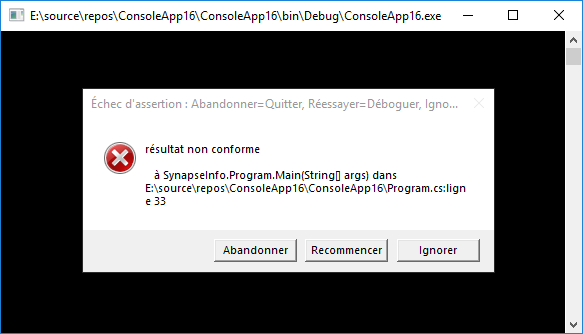
nbHeures = laMission.getRelevéHoraire()[laDate];

Debug.Assert(nbHeures == 8, "résultat non conforme");

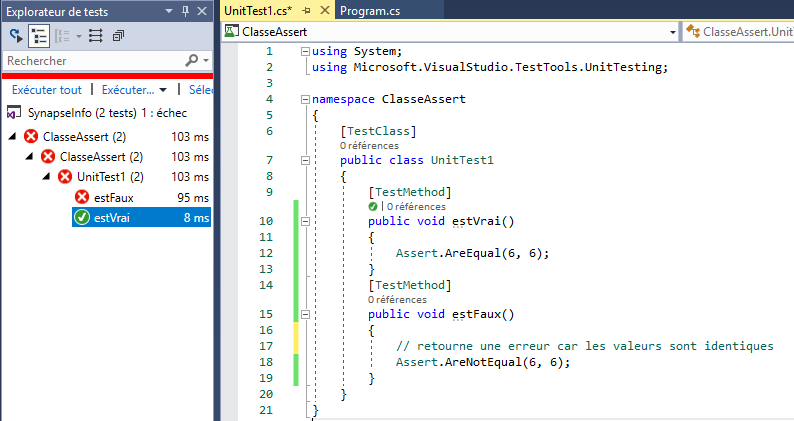
Console.Read();

}

}



# Framework de test unitaire



public class UnitTest1

{

[TestMethod]

public void estVrai()

{

// retourne une erreur quand les valeurs ne sont pas identiques

Assert.AreEqual(6, 6);

}

[TestMethod]

public void estFaux()

{

// retourne une erreur quand les valeurs sont identiques

Assert.AreNotEqual(5, 6);

}

[TestMethod]

public void estPareil()

{

// retourne une erreur quand les objets ne sont pas identiques

string p = "pareil";

Assert.AreSame(p, p);

}

[TestMethod]

public void estDifferent()

{

// retourne une erreur quand les objets sont identiques

string p = "pareil";

Assert.AreNotSame(p, p);

}

[TestMethod]

public void estNull()

{

// retourne une erreur quand l'objet n'est pas null

string d = null;

Assert.IsNull(d);

}

[TestMethod]

public void pasNull()

{

// retourne une erreur quand l'objet est null

string d = null;

Assert.IsNotNull(d);

}

[TestMethod]

public void estTrue()

{

// retourne une erreur quand la valeur est false

bool vrai = true;

Assert.IsTrue(vrai);

}

[TestMethod]

public void estFalse()

{

// retourne une erreur quand la valeur est true

bool vrai = true;

Assert.IsFalse(vrai);

}

}

# Classe générique

class MaClasseGenerique<T>

{

private int capacite;

private int nbElements;

private T[] tableau;

// T peut prendre n'importe quel type : int, double, string, classe définie comme Intervenant

public MaClasseGenerique()

{

capacite = 10;

nbElements = 0;

tableau = new T[capacite];

}

// procédure d'ajout, le type doit être le même pour l’instance créée

// mais on peut créer des instances différentes avec des types différents

public void Ajouter(T element)

{

if (nbElements >= capacite)

{

capacite \*= 2; // arbitraire on aurait pu ajouter 10 au lieu de doubler la capacité

T[] copieTableau = new T[capacite];

for (int i = 0; i < tableau.Length; i++)

{

copieTableau[i] = tableau[i];

}

tableau = copieTableau;

}

tableau[nbElements] = element;

nbElements++;

}

public T ObtenirElement(int indice)

{

if (indice < 0 || indice >= nbElements)

{

Console.WriteLine("L'indice n'est pas bon");

return default(T);

}

return tableau[indice];

}

public int CompterEl()

{

return nbElements;

}

}

# Programme de test

MaClasseGenerique<int> maListe = new MaClasseGenerique<int>();

for (int i = 0; i < 30; i++) {

maListe.Ajouter(i);

}

Console.WriteLine("élément à l'indice 2 : "+maListe.ObtenirElement(2));

Console.WriteLine("nombre d'éléments : "+maListe.CompterEl());

MaClasseGenerique<string> nbliste = new MaClasseGenerique<string>();

nbliste.Ajouter("zéro");

nbliste.Ajouter("un");

nbliste.Ajouter("deux");

nbliste.Ajouter("trois");

Console.WriteLine("élément à l'indice 2 : "+nbliste.ObtenirElement(2));

Console.WriteLine("nombre d'éléments : " + nbliste.CompterEl());

On obtient le résultat suivant : 