**La classe Hashtable**

|  |  |
| --- | --- |
| **clé** | **valeur** |
| **clé1** | **valeur1** |

Représente une collection de paires clé/valeur qui sont organisées en fonction du code de hachage de la clé.

On peut voir une Hashtable comme un tableau à deux colonnes :

**Constructeurs publics**

|  |  |
| --- | --- |
| **Nom** | **Description** |
| [Hashtable()](http://msdn.microsoft.com/fr-fr/library/7b3y50ay.aspx) | Surchargé. Initialise une nouvelle instance vide de la classe Hashtable. |
| [Hashtable(int)](http://msdn.microsoft.com/fr-fr/library/7b3y50ay.aspx) | Initialise une nouvelle instance vide de la classe Hashtable à l’aide de la capacité initiale spécifiée. |

**Propriétés publiques**

|  |  |
| --- | --- |
| **Nom** | **Description** |
| [Count](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.count%28v%3DVS.80%29.aspx) | Obtient le nombre de paires clé/valeur contenues dans **Hashtable**. |
| Keys | Obtient [une](http://msdn.microsoft.com/fr-fr/library/system.collections.icollection.aspx) collection contenant les clés de **Hashtable**. |
| [Values](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.values.aspx) | Obtient une collection contenant les valeurs de **Hashtable**. |
| [Item](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.item.aspx) | Obtient ou définit la valeur associée à la clé spécifiée. |

**Méthodes publiques**

|  |  |
| --- | --- |
| **Nom** | **Description** |
| [Add](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.add.aspx)  | Ajoute un élément avec la clé et la valeur spécifiées dans **Hashtable**.  |
| [Clear](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.clear.aspx)  | Supprime tous les éléments de **Hashtable**.  |
| [Clone](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.clone.aspx)  | Crée une copie superficielle de **Hashtable**.  |
| [Contains](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.contains.aspx)  | Détermine si **Hashtable** contient une clé spécifique.  |
| [ContainsKey](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.containskey.aspx)  | Détermine si **Hashtable** contient une clé spécifique.  |
| [ContainsValue](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.containsvalue.aspx)  | Détermine si **Hashtable** contient une valeur spécifique.  |
| [CopyTo](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.copyto.aspx)  | Copie les éléments **Hashtable** dans une instance de [Array](http://msdn.microsoft.com/fr-fr/library/system.array.aspx) unidimensionnelle à l'index spécifié.  |
| [Equals(Object)](http://msdn.microsoft.com/fr-fr/library/bsc2ak47.aspx)  | Détermine si l'[Object](http://msdn.microsoft.com/fr-fr/library/system.object.aspx) spécifié est égal à l'[Object](http://msdn.microsoft.com/fr-fr/library/system.object.aspx) en cours. (Hérité de [Object](http://msdn.microsoft.com/fr-fr/library/system.object.aspx).) |
| [GetType](http://msdn.microsoft.com/fr-fr/library/system.object.gettype.aspx)  | Obtient le [Type](http://msdn.microsoft.com/fr-fr/library/system.type.aspx) de l'instance actuelle. (Hérité de [Object](http://msdn.microsoft.com/fr-fr/library/system.object.aspx).) |
| [KeyEquals](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.keyequals.aspx)  | Compare un [Object](http://msdn.microsoft.com/fr-fr/library/system.object.aspx) spécifique avec une clé spécifique dans **Hashtable**.  |
| [Remove](http://msdn.microsoft.com/fr-fr/library/system.collections.hashtable.remove.aspx)  | Supprime de **Hashtable** l'élément ayant la clé spécifiée.  |
| [ToString](http://msdn.microsoft.com/fr-fr/library/system.object.tostring.aspx)  | Retourne une chaîne qui représente l'objet actuel. (Hérité de [Object](http://msdn.microsoft.com/fr-fr/library/system.object.aspx).) |

**Exemple 1**

sing System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Collections;

namespace Test1Hashtable

{

 class Program

 {

 static void Main(string[] args)

 {

 Hashtable t = new Hashtable();

 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ajout\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 t.Add("A1", "ali");

 t.Add("A2", "yassine");

 t.Add("A3", "hamza");

 t.Add("A4", "salma");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Affichage de l'élément ayant comme clé A2\*\*\*\*\*\*\*\*/

 Console.WriteLine(t["A2"]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*Affichage des éléments du hashtable\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// DictionaryEntry: Définit une paire clé/valeur du hashtable

foreach (DictionaryEntry d in t)

 {

 Console.WriteLine(d.Key + "\t" + d.Value);

 }

/\*\*\*\*\*\*\*\*\*\*\*\*\* Gestion de l'exception si la clé existe déjà\*\*\*\*\*\*\*\*\*\*/

 try

 {

 t.Add("A2", "sofia"); // la clé A2 existe déjà

 }

 catch { Console.WriteLine("un élément existe déjà avec cette clé"); }

/\*\*\*\*\*\*\*\*\*\*\*Modifier un élément à partir de sa clé => si l'élément

existe il sera modifié sinon, il sera créé\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 t["A3"] = "hind";

 Console.WriteLine("la valeur associée à la clé A3 est:"+t["A3"]);

 t["A6"] = "hicham";

 Console.WriteLine("la valeur associée à la clé A6 est:" + t["A6"]);

/\*\*\*\*\*\*\*\*\*\*\*\*\* Vérifier si une clé existe déjà\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 if (t.Contains("A5") == false)

 {

 t.Add("A5", "anas");

 }

 else Console.WriteLine("clé déjà existante");

/\*\*\*\*\*\*\*\*\*\*\*\*\* Vérifier si une valeur existe déjà\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 if (t.ContainsValue("anas") == true) Console.WriteLine("valeur existante");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*Affichage des clés du hashtable\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 foreach (string s in t.Keys)

 {

 Console.WriteLine(s);

 }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*Affichage des valeurs du hashtable\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 foreach (string s in t.Values)

 {

 Console.WriteLine(s);

 }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Supprimer une valeur associée à une clé\*\*\*\*\*\*\*/

 t.Remove("A1");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*Affichage des éléments du hashtable\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 foreach (DictionaryEntry d in t)

 {

 Console.WriteLine(d.Key + "\t" + d.Value);

 }

 Console.Read();

 }

 }

}

**Exemple 2**

class Personne:IComparable

 {

 public string CIN { get; set; }

 public string Nom { get; set; }

 public string Prenom { get; set; }

 public Personne() { }

 public Personne(string c, string n, string pr)

 {

 CIN = c;

 Nom = n;

 Prenom = pr;

 }

 public override string ToString()

 {

 return CIN + "\t" + Nom + "\t" + Prenom;

 }

 public int CompareTo(Object obj)

 {

 if (obj is Personne)

 {

 Personne p = (Personne)obj;

 return this.Nom.CompareTo(p.Nom);

 }

 else throw new ArgumentException("object not a person");

 }

 }

class Program

 {

 static void Main(string[] args)

 {

 Hashtable t = new Hashtable();

 Personne p1 = new Personne("jc2131", "alami", "ahmed");

 Personne p2 = new Personne("ac1267", "alaoui", "yassine");

 Personne p3 = new Personne("bd2345", "chaaibi", "salma");

 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ajout\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 t.Add(p1.CIN, p1);

 t.Add(p2.CIN, p2);

 t.Add(p3.CIN, p3);

 foreach (DictionaryEntry d in t)

 {

 Console.WriteLine(d.Value);

 }

 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Suppression\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 t.Remove(p1.CIN);

 Console.WriteLine("\*\*\*\*\*\*\*\*\*\*Suppression\*\*\*\*\*\*");

 foreach (Personne p in t.Values)

 {

 Console.WriteLine(p);

 }

 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Rechercher une personne selon le nom\*\*\*/

 Console.WriteLine("donner le nom");

 string nom = Console.ReadLine();

 foreach (Personne p in t.Values)

 {

 if (p.Nom == nom) Console.WriteLine(p.ToString());

 }

 /\*\*\*\*\*\*\*\*\*\*\*\*\* Trier les valeurs d’une Hashtable\*\*\*\*\*\*\*\*\*/

 List<Personne> l = new List<Personne>();

 foreach (Personne p in t.Values) l.Add(p);

 l.Sort();

 foreach (Personne p in l) Console.WriteLine(p.ToString());

 Console.Read();

 }

 }