# COLECCIONES - CONTINUACIÓN

# LISTAS

# LA CLASE ARRAYLIST

- El arraylist es una colección genérica. EN TEORÍA puede alamcenar elementos de distinto tipo.
- Es una colección dinámica. La cantidad de elementos puede modificarse en tiempo de ejecución.

# DECLARACIÓN E INSTANCIACIÓN

```
static void Main(string[] args)
{
    ArrayList miLista;
    miLista = new ArrayList(5); //podemos pasarle o no la capacidad del array
}
```

# PROPIEDADES DE UN ARRAYLIST

- Capacity: Nos permite conocer la cantidad de nodos que presenta la lista.
- Count: Nos permite conocer la cantidad de objetos efectivamente agregadas a una lista.
- Para nuestro caso, cuál sería count y capacity? Son propiedades de instancia?

# AGREGANDO UN ELEMENTO

- Para agregar un objeto a la lista usamos el método Add (método de instancia).
- El agregado será de forma secuencial como en las colecciones o arrays.

```
miLista.Add(5555);
miLista.Add("Holaaa");
Cuenta cuenta = new Cuenta();
miLista.Add(cuenta);
```

# CAPACIDAD DE UN ARRAYLIST

- Que pasa con la capacidad de un array? Es fija? Veamos un ejemplo:

```
ArrayList miLista;
miLista = new ArrayList(3); //podemos pasarle o no la capacidad del array
miLista.Add("Holaaa");
miLista.Add(24U);
miLista.Add(234);
Console.WriteLine("Capacidad: {0}", miLista.Capacity);
miLista.Add(5555);
Console.WriteLine("Capacidad: {0}", miLista.Capacity);
Cuenta cuenta = new Cuenta();
miLista.Add(cuenta);
Console.ReadKey();
```

# AGREGADO EN POSICIÓN

- Para agregar un elemento en una posición determinada podemos usar el método de instancia Insert(Índice,Objeto).
- Nos permite agregar un item en una posición específica corriendo el resto de los elementos automáticamente

```
miLista.Insert(3, 200.233F);
```

# DESAGREGADO DE OBJETOS

- Para sacar un elemento de la lista podemos usar el método de instancia Remove(objeto).
- Acá podemos eliminar el objeto que se pasemos por parámetro (o valor siempre y cuando sea un tipo primitivo).

```
miLista.Remove(200.233F);
miLista.Remove(cuenta);
```

# DESAGREGADO EN POSICIÓN

- Como desagregamos elementos cuando no sabemos el valor exacto?
- Para desagregar un elemento en una posición usamos el método RemoveAt(índice)

```
miLista.RemoveAt(2);
```

# REAJUSTE DE CAPACIDAD

- Si queremos recortar la capacidad de una lista a la cantidad de elementos agregados usamos el método trimToSize()

```
Console.WriteLine("Capacidad antes de recortar: {0}", miLista.Capacity);
miLista.TrimToSize();
Console.WriteLine("Capacidad despues de recortar: {0}", miLista.Capacity);
```

# DESAGREGADO GENERAL

- Para limpiar una colección y que quede vacía usamos el método de instancia clear()

```
miLista.Clear()
```

# BÚSQUEDA SIMPLE

- Para saber si un objeto está en un elemento podemos usar el método contains pero este devuelve un booleano y NO una  posición

# ACLARACIÓN

- En ArrayList podemos usar cualquiera de los métodos que usamos para arrays:
  - IndexOf()
  - LastIndexOf()
  - Sort()
  - Etc…

# LIST

# DIFERENCIA ENTRE LIST Y ARRAYLIST

- Cuando declaramos una ArrayList vamos a trabajar con tipos de datos genéricos ya que almacena objetos.
- En cambio, cuando trabajamos con List<T> vamos a trabajar con lista de un tipo de elementos específicos.
  - Por ejemplo List<Alumno> almacenará elementos de la clase alumno y nada más
- ArrayList se encuentra deprecado en la actualidad y se prefiere el uso de List para trabajar

# DECLARACIÓN EN C#

```
List<Cuenta> cuentas = new List<Cuenta>(15);
cuentas.Add(cuenta);
cuentas.Remove(cuenta);
```

# HASHTABLE

# HASHTABLES

- Representa una colección pero que no está enumerada.
- No nos basamos en la posición y no necesariamente en si esté ordenada.
- Los elementos se organizan en pares key-value/values. La key debe ser única para cada elemento.

**Array**

| Value |
|---|
| New York |
| Boston |
| Mexico |
| Kansas |
| Detroit |
| California |

**Hash Table**

| Key | Value |
|---|---|
| 1 | New York |
| 2 | Boston |
| 3 | Mexico |
| 4 | Kansas |
| 5 | Detroit |
| 6 | California |

# DECLARANDO UNA HASHTABLE

```
Hashtable capitales = new Hashtable();


capitales.Add("Argentina", "Buenos Aires");
capitales.Add("Uruguay", "Montevideo");
capitales.Add("Paraguay", "Asuncion");
capitales.Add("Brasil", "Brasilia");
```

# ACCEDIENDO A VALORES DE UNA HASHTABLE

```
Console.WriteLine("La capital de Brasil es: ", capitales["Brasil"]);
```

# EL MÉTODO CONTAINSKEY

```
if (!capitales.ContainsKey("Chile"))
{
    capitales.Add("Chile", "Santiago");
}
else
{
    Console.WriteLine("Ya existe un pais con ese nombre, sorry");
}
```