# Colecciones y Genéricos

# Vectores y sus limitaciones

- Un programa debe declarar el tamaño que tendrá el vector al momento de ser creado. Java no permite cambiar su tamaño luego de definido

- Añadir o quitar elementos del medio de un vector es muy ineficiente

# Colecciones

- Las colecciones comparten ciertas características con los vectores, pero le agregan más flexibilidad
  - Sólo pueden contener un tipo de datos y tiene que ser especificado al momento de ser declarada. Una vez declarada no puede ser modificado el tipo de datos
  - Se puede acceder a la mayoría de ellas al igual que los vectores, utilizando su índice
  - Se añade el atributo ‘Count’ que nos informará de la cantidad de elementos que posea la colección

# Colecciones

- Tipos de Colecciones

| Colección | Descripción |
|---|---|
| **ArrayList<T>** | Vector dinámico que contiene valores de tipo ‘T’ |
| LinkedList<T> | Lista enlazada que contiene valores de tipo ‘T’ |
| Queue<T> | Cola de valores tipo ‘T’ – Paradigma FIFO |
| Stack<T> | Pila de valores tipo ‘T’ – Paradigma FILO |
| **Dictionary<Tkey, TValue>** | Diccionario con llaves tipo ‘Tkey’ y valores tipo ‘Tvalue’ |
| HashSet<T> | Un set sin valores duplicados sobre el cual se pueden aplicar operaciones de conjuntos |

# Colecciones

- Notamos en la anterior tabla la notación ‘<T>’. Dicha notación indica que las colecciones pueden ser declaradas de cualquier tipo definido en nuestro programa. A esto se lo denomina colecciones *genéricas*, ya que pueden albergar distintos tipos de datos en una mismo tipo de estructura
- Ejemplo
  - ArrayList<Decimal> listaPrecios
  - ArrayList<String> listadoAlumnos
  - Dictionary<Integer,String> diccionarioDnis
  - Dictionary<String,String[]> diccionarioGrupos

# Listas Genéricas

- Las listas genéricas poseen distintas operaciones posibles para su uso, algunos ejemplos son
  - listadoAlumnos.Add(“Pedro”); // Añade un nuevo elemento a la lista
  - listadoAlumnos.Add(3, “Juan”); //Añade el elemento en la posición 3
  - listadoAlumnos.Remove(3);//Remueve el elemento en la posición 3
  - listadoAlumnos.Contains(“Pedro”);//Retorno booleano si existe o no
  - listadoAlumnos.IndexOf(“Juan”);//Devuelve la posición en la que se encuentra lo buscado o -1 en caso de no encontrarlo
  - Collections.Sort(listadoAlumnos); //Ordena el listado de ser posible

# Iteraciones – Foreach

- La instrucción 'foreach'(por cada) se utiliza para recorrer una colección de elementos
- No se debe utilizar para cambiar el contenido de la colección que se esta iterando ya que se pueden producir efectos secundarios no previsibles

# Iteraciones - Foreach

- Sintaxis

```
for (tipo variable : coleccion<tipo>)
{
   //Bloque de codigo
}
```

CAPTTON
IT CONSULTING