**IEnumerable<T>, ICollection<T> , IList<T>**

ICollection<T> inclou IEnumerable<T>

IList<T> inclou ICollection i també IEnumerable<T>

|  |  |
| --- | --- |
| TaulaLlista<T> amb un array | Main |
| public class TaulaLlista<T>:~~IEnumerable<T>~~, ~~ICollection<T>~~, IList<T>  private T[] dades;  private int nELem;  public TaulaLlista(){ dates = new T[5]; nElem = 0; }  public TaulaLlista(IEnumerable<T> items){  //afegir els elements de items a la nostra llista utilitzant el foreach i el this.Add  }  //public IEnumerator<T> GetEnumerator(){  // for(int i=0;i<this.Count;i++)  // yield return this[i];  //}  //public IEnumerator<T> GetEnumerator(){  // for(int i=0;i<this.Count;i++)  // yield return this.dades[i];  //}  public IEnumerator<T> GetEnumerator(){  return new ElMeuEnumerador(dades, nElem);  }  public override String ToString(){  //utilitzar el StringBuilder, el foreach doncs  //la llista és IEnumerable  }  Propietats Icollection  Count  IsReadOnly  Clear  Mètodes ICollection  Add(T item) → DoubleCapacity()  Remove(T item) → Eliminar el primer element de la llista que sigui “Equals” a item  Cal repassar com s’implementa el Equals d’una classe  Contains(T item) →  CopyTo    Propietats IList  this[int index] {get{…} set {…}}    Mètodes de IList  IndexOf(T item)  Insert(int index, T item)  RemoveAt(int index)  Altres mètodes  T[] ToArray()  DoubleCapacity()  Equals() → Comprovar si dos objectes de tipus TaulaLlista són iguals  Per tal que dos llistes siguin igual cal que tinguin la mateixa longitud  i que tots els elements de les llistes siguin iguals (Equals) | TaulaLlista<int> t1 = new TaulaLlista<int>();  t1.Add(2);  t1.AddRange([8,5,4,2]);  foreach(int num in t1) CW(num);  //2 8 5 4 2  Puc fer això?  TaulaLlista<int> t2 = new TaulaLlista<int>(t1);  Sí doncs és t1 és IEnumerable  Puc recorrer la taula t2 amb un iterador???  Ienumetador<int> iterador = t2.GetEnumerator();  while(iterador.MoveNext) CWL(iterador.Current);  //2 8 5 4 2  while(iterador.MoveNext) CWL(iterador.Current);  //no mostra res  iterador.Clear();  while(iterador.MoveNext) CWL(iterador.Current);  //2 8 5 4 2 |

|  |  |
| --- | --- |
| Pila<T> amb un array | Main |
| public Pila<T>:IEnumerable<T> {  private T[] dades;  private int top;  public Pila(){ top = ??; dades = new T[5]; }  public Pila(IEnumerable<T> items){ ??? }  Mètodes de Ienumerable  public Ienumerator<T> GetEnumarator();  Propietat que no implementen cap interfície  this[int index]{get{...}set{...}}  Count  IsReadOnly  Mètodes que no implementen cap interfície  Push(T item)  Pop()  Peek()  bool TryPeek(out T value)  ToStrindg()  Equals()  } | TaulaLlista<int> taula = new TaulaLlista<int>([2,3,4,5,6])  Pila<int> p1 = new Pila<int>(taula);  //dibuix com seria la pila... |

|  |  |
| --- | --- |
| IEnumerator<T> | Main |
| public class EnumeradorRevers<T>:IEnumetator<T>{  private T[] dades;  private int nElem;  private int index;  public EnumeradorRevers(T[] dades, int nElem){  this.nElem = nElem; this.dades = dades;  this.index = nElem;  }  public T Current { get{ return this.dades[index]; } }  public bool MoveNext(){ .. }  public void Reset(){ .. }  } |  |

|  |  |
| --- | --- |
| IComparable<T> | Main |
| public Pokemon:**Icomparable**<Pokemon> {    private string nom;  private int atac;  private int defensa;  private int velocitat;  public int **CompareTo(Pokemon other)**{  int result;  if(this == null && other==null) result = 0;  elseif(this == null) result = -1;  elseif(other==null) result = 1;  else{  result = this.nom.CompareTo(other.nom);  }  return result;  }  public class ComapardorAtac:**Icomparer**<Pokemon>{  public int **Compare(Pokemon p1, Pokemon p2)**{  int result;  if(p1 == null && p2==null) result = 0;  elseif(p1 == null) result = -1;  elseif(p2==null) result = 1;  else{  result = p1.Atac.CompareTo(p2.Atac);  if(resultat==0)  result = p1.Defensa.CompareTo(p2.Defensa);  if(resultat==0)  result = p1.Velocitat.CompareTo(p2.Velocitat);    }  return result;  }  }  } | List<Pokemon> pokes = new List<Pokemon>();  pokes.Add(new Pokemon(...));  pokes.Add(new Pokemon(...));  pokes.Add(new Pokemon(...));  pokes.Add(new Pokemon(...));  //ordenar per defecte  pokes.Sort();  //ordenar per atac  pokes.Sort(new ComapardorAtac());  //ordenar array  int[] nums = {3,4,2,3,4,5,6,7,8,8};  Arrays.Sort(nums);  //ordenar array  Pokemon[] pokes = {new Pokemon(…),new Pokemon(…),new Pokemon(…),new Pokemon(…)};  Arrays.Sort(pokes);  Arrays.Sort(pokes,new ComapardorAtac()); |

|  |  |
| --- | --- |
| TaulaLlistaD<T> | Main |
| public class TaulaLlistaD<T>:IEnumerable<T>{  private Node head;  private Node tail;  private int nElem;  Count  this[index]  GetEnumerator()  Add  IndexOf  Contains  ToString  } | En el main la LlistaTaulaD no té cap diferència amb la LlistaTaula |