IHashTable IteradorSencillo<E> <<interface>> -serialVersionUID = 1**IEstructura** Proyecto 2 Estructura de Datos -actual: Nodo<E> +put(K, V) +get(K) +IteradorSencillo(Nodo<E>) Gabriela Paez y Juan Villamil | October 14, 2019 +hasNext(): boolean +delete(K) +size(): int +next(): E +kevs() +set(int, E): E +isEmpty(): boolean **TablaHashSeparateChaining** +get(int): E +contains(Object): boolean -INIT CAPACITY = 7 ArbolBinarioOrdenado +clear(): void -n:int +addLast(E): boolean -m:int Arbol -serialVersionUID = 1 +add(int, E): void -tablaSimbolos: TablaSimbolos<K, V>[] +remove(Object): boolean -serialVersionUID = 1 +ArbolBinarioOrdenado() +TablaHashSeparateChaining() +remove(int): E -raiz: NodoAbstracto<K, T> +agregar(K, T): void +TablaHashSeparateChaining(int) +toArray(): Object[] +eliminar(K): void +resize(int): void Arbol() +indexOf(E): int +arbolCabeza(K): ArbolBinarioOrdenado<K,T> +hash(K): int vaciar(): void +addFirst(E): boolean +darldentificadorMenor(): K +size(): int contieneIdentificador(K): boolean +removeFirst(): boolean +darldentificadorMayor():K +isEmpty(): boolean contieneElemento(T): boolean +removeLast(): boolean +darSubArbol(K, K): ArbolBinarioOrdenado<K,T> +contains(K) : boolean darNodos(): Collection<NodoAbstracto<K,T>> +darArbolCola(K): ArbolBinarioOrdenado<K,T> +get(K): V buscar(K): T +darInorden(:): Collection <T> +siguientePrimo(int): int estaVacio(): boolean ListaSencillamenteEncadenada +put(K, V): void darIdentificadores(): set<K> +delete(K): V darPeso(): int -serialVersionUID = 1ListaSencillamenteEncadenadaAbstracta +keys(): iterator<K> darPreorden() : Collection <T> +ListaSencillamenteEncadenada() darPosorden() : Collection <T> -serialVersionUID = 1+ListaSencillamenteEncadenada(E) darCamino(K): Collection <T> NodoAbo -cantidadElementos: int +add(int, E): void darRaiz(): NodoAbstracto<K,T> -primerNodo: Nodo<E> +remove(Object): boolean eliminar(K): void -serialVersionUID = 1 -ultimoNodo: Nodo<E> +remove(int): E -izq: NodoAbo<K,T> +addLast(E): boolean -der: NodoAbo<K.T> +toArray(): Object[] +indexOf(E): int +size(): int +addFirst(E): boolean NodoABO(K, T) MaxHeapCP **TablaSimbolos** +set(int, E): E +removeFirst(): boolean darIzquierda(): NodoAbo<K,T> +iterator(): Iterator<E> +removeLast(): boolean darDerecha(): NodoAbo<K,T> -elementos : T[] n:int +isEmpty(): boolean contieneIdentificador(K): boolean -tamano: int primero: NodoST<K, V> +get(int): E contieneElemento(T): boolean +darNodo(int): Nodo<E> +MaxHeapCP() darNodos(Collection<NodoAbstracto<K, T>>): +TablaSimbolos() +contains(Object): boolean NodoAbstracto +darNumElementos(): int void +size(): int +clear(): void +esVacia(): boolean buscar(K): T +isEmpty(): boolean -serialVersionUID = 1+sacarMax(): T eliminar(K): NodoAbo<K.T> +contains(K): boolean -identificador: K +darMax(): T darIdentificadorMenor(): K +get(K): V -elemento: T +agregar(T): void +put(K, V): void arbolCabeza(K, ArbolBinarioOrdenado<K, T>): NodoST +expand(): void +delete(K): V +NodoAbstracto(K, T) +sink(int): void darldentificadorMayor(): K +keys(): iterable <K> +darIdentificador(): K -llave : K +swim(int): void subArbol(K, K, ArbolBinarioOrdenado<K, T>): +darElemento(): T -elementos: ListaSencillamenteEncadenada<V> +contieneIdentificador(K): boolean -siguiente: NodoST<K, V> arbolCola(K, ArbolBinarioOrdenado<K, T>): +contieneElemento(T): boolean Nodo <<Serializable>> void +darNodos(Collection<NodoAbstracto<K, T>>) darInorden(Collection<T>): void +NodoST(K, V) : void -serialVersionUID = 1 darMenor(): NodoAbo<K.T> +darLlave(): K +buscar(K): T -elemento: E darIdentificadores(Collection<K>): void +cambiarSiguienteST(NodoST<K, V>): void +darIdentificadores(Collection<K>): void -siguiente: Nodo<E> darPeso(): int +cambiarElemento(V): void +darPeso(): int darPreorden(Collection<T>): void +darSiguiente(): NodoST<K,V> +darPreorden(Collection<T>): void +Nodo(elemento:E) darPosorden(Collection<T>): void +darElemento(): V +cambiarSiguiente(siguiente:Nodo<E>):void +agregar(T, K...): void agregar(T, K...): void +cambiarSiguiente(NodoST<K, V>): void +darPosorden(Collection<T>): void +darElemento(): E darCamino(K, Collection<T>) +darElementos(): +darCamino(K, Collection<T>): void +cambiarElemento (elemento:E): void darNodo(K): NodoAbo<K,T> ListaSencillamenteEncadenada<V> +darNodo(K): NodoAbstracto<K,T> +darSiguente():Nodo<E>