|  |
| --- |
| Incidents |
| * ID : string * state : string * date : string * fatalities : int * race : string * gender : string |
| * Incidents() : * Incidents( sID : string, sState : string, sDate : string , sFatalities : int, sRace : string, sGender : string) : * setID(sID : string) : void * setState(sState : string) : void * setDate(sDate : string) : void * setFatalities(sFatalities : int) : void * setRace(sRace : string) : void * setGender(sGender : string) : void * getID() const : string * getState() const : string * getDate() const : int * getFatalities() const : int * getRace() const : string * getGender() const : string |

|  |
| --- |
| Binary Node |
| * item : ItemType * leftPtr : BinaryNode<ItemType>\* * rightPtr : BinaryNode<ItemType>\* |
| * BinaryNode( anItem : ItemType) : * BinaryNode( anItem : ItemType, left : BinaryNode<ItemType>\* , right : BinaryNode<ItemType>\*) : * setItem( anItem : const ItemType) : void * setLeftPtr( left : BinaryNode<ItemType>\*) : void * setRightPtr( right : BinaryNode<ItemType>\*) : void * getItem() const : ItemType * getLeftPtr() const : BinaryNode<ItemType>\* * getRightPtr() const : BinaryNode<ItemType>\* |

|  |
| --- |
| Binary Tree |
| * rootPtr : BinaryNode<ItemType>\* * count : int |
| * BinaryTree() : * BinaryTree( tree : const BinaryTree<ItemType>) : * ~BinaryTree() : virtual * operator = ( sourceTree : const BinaryTree ) : BinaryTree * isEmpty() const : bool * size() const : int * clear() : void * preOrder( visit(obj : ItemType) : void ) const : void * inOrder( visit(obj :ItemType ) : void ) const : void * postOrder( visit(obj : ItemType ) : void ) const : void * levelOrder( visit(obj : ItemType ) : void ) const : void * indentPrint( visit(obj : ItemType) : void, level : int ) : void * insert( newData : const ItemType, compare( one : const ItemType, two : const ItemType) : int ) = 0 : virtual bool * remove( data : const ItemType, compare( one : const ItemType, two : const ItemType) : int ) = 0 : virtual bool * getEntry( anEntry : const ItemType, returnedItem : ItemType,   compare( one : const ItemType, two : const ItemType) : int ) const = 0 : virtual bool * theSmallest( item : ItemType) = 0 : virtual void * theLargest( item : ItemType) = 0 : virtual void * destroyTree( nodePtr : BinaryNode<ItemType>\*) : void * copyTree( nodePtr : const BinaryNode<ItemType>\*) : BinaryNode<ItemType>\* * \_preorder( visit( obj : ItemType) : void, nodePtr : BinaryNode<ItemType>\*) const : void * \_inorder( visit( obj : ItemType) : void , nodePtr : BinaryNode<ItemType>\*) const : void * \_postorder( visit( obj : ItemType) : void, nodePtr : BinaryNode<ItemType>\*) const : void * \_levelorder( visit(obj : ItemType) : void, nodePtr : BinaryNode<ItemType>\*) const : void * \_indentprint(visit(obj : ItemType) : void, nodePtr : BinaryNode<ItemType>\*, level : int) : void |

|  |
| --- |
| Binary Search Tree |
| * bn : BinaryNode<ItemType>\* |
| * \_insert( nodePtr : BinaryNode<ItemType>\*, newNode : BinaryNode<ItemType>\*,   compare( one : const ItemType, two : const ItemType) : int ) : BinaryNode<ItemType>\* * \_remove( nodePtr : BinaryNode<ItemType>\*, target : const ItemType, success : bool,   compare( one : const ItemType, two : const ItemType ) : int ) : BinaryNode<ItemType>\* * deleteNode( targetNodePtr : BinaryNode<ItemType>\*) : BinaryNode<ItemType>\* * removeLeftmostNode( nodePtr : BinaryNode<ItemType>\*, successor : ItemType) : BinaryNode<ItemType>\* * findNode( nodePtr : BinaryNode<ItemType>\*, target : const ItemType,   compare( one : const ItemType , two : const ItemType ) : int ) const : BinaryNode<ItemType>\* * findSmallest( nodePtr : BinaryNode<ItemType>\*, item : const ItemType ) const : BinaryNode<ItemType>\* * findLargest( nodePtr : BinaryNode<ItemType>\*, item : const ItemType) const : BinaryNode<ItemType>\* * insert( newEntry : const ItemType, compare( one : const ItemType, two : const ItemType) : int ) : bool * remove( anEntry : const ItemType, compare( one : const ItemType, two : const ItemType) : int ) : bool * getEntry( target : const ItemType, returnedItem : ItemType,   compare( one : const ItemType, two : const ItemType) : int ) const : bool * theSmallest( item : ItemType ) : void * theLargest( item : ItemType ) : void |

|  |
| --- |
| HashTable |
| * key : string * value : Incidents * next : HashTableNode\* * head : HashTableNode\* * collisions : int\* * loadFactor : int * totalNodes : int * totalCollisions : int |
| * HashTable() : * ~HashTable() : * insertNode ( incident : Incidents, key : string ) : void * deleteNode ( key : string ) : bool * searchNode ( key : string, nodeValue : Incidents) : bool * printHashTable () : void ; * printStatistics() : void |