Set

A Set is a collection of elements, with no strict ordering

Duplicates are not allowed.

Set ADT

Specification

Elements: Set elements can be of any type, but we will assume SetElement

Structure: Any mechanism for tracking the items

Set ADT Continued

function create (s: Set, isCreated: boolean)
results: if s cannot be created, isCreated is false; otherwise, isCreated is true,the Set is created and is empty

function terminate (s: Set)
results: Set s no longer exists

Set ADT Continued

function isFull (s: Set)
results: returns true if the Set is full; otherwise false is returned

function isEmpty (s: Set)
results: returns true if the Set is empty; otherwise, false is returned

function contains(s: Set, e: SetElement, b: Boolean)
results: set b to true if e is in the Set; otherwise set b to false

Set ADT Continued

function insert (s: Set, e: SetElement)
requires: isFull (s) is false, contains(s, e) is false
results: element e is added to the Set

function remove (s: Set, e: SetElement)
requires:contains(s, e) is true
results: the element e is removed from the set

Set ADT Continued

function union (s1: Set; s2: Set; result: Set)
    results: each element that is in either s1 or s2 (non-exclusive or) is added to result

function intersection(s1: Set; s2: Set; result: Set)
    results: each element that is in both s1 and s2 is added to result

Set ADT

Can we use a List to build this data structure?

What other operations would be useful?

Can we print every element of the set to the screen?

Iterator

Design Pattern

Used to traverse all elements in a container

keep track a current pointer in the container (state!)

first()

hasNext()

next()

last()

Generally used in Object Oriented Languages but can be applied to any data structure.

C arrays do not provide this interface.

Bag

A Bag is similar to a Set but duplicates are allowed in the Bag.