M – Arrays – Sorted

Code Samples - Documentation

# Examples

The following examples are used to illustrate this topic.

1. **BranchVault** – The BranchVault class manages safety deposit boxes. Each box has a unique number, and may be assigned to a particular account number. This class demonstrates the bubble sort and the binary search.
2. **PhoneBook** – The PhoneBook class provides simple management of an array of PhoneNumber objects. The PhoneBook allows for adding and retrieving phone numbers; it supports search for phone numbers by the telephone number or by a last name. The PhoneBook is initialized with a physical limit for the arrays, and the actual number of entries in the phone book is tracked as phone numbers are added to the phone book. The AddPhoneNumber() method demonstrates adding to an array that is to be sorted.
3. **ClassList** –The ClassList example maintains an array of students for a particular course by offering methods to add and remove students from a course. In addition, the ClassList ensures that students are not added to the course twice (based on the student’s Id). The internal array is of a fixed size (ClassList.CLASS\_LIMIT) as well as the logical size of the array. If the ClassList is provided with an array of students in its constructor, that list is “copied” to the internal array of the ClassList object.
4. **BankStatement** – [***Not Implemented*** - Sorting by Date] This class represents a bank statement for a BankAccount for a given month. The statement allows BankTransaction objects to be added, and performs deposits and withdrawals on the BankAccount. The statement reports the starting and ending balance and also summarizes the total amount deposited and withdrawn for the month.

# Overview

The focus of this topic is sorting arrays and performing searches, inserts, and deletions from a sorted array.

# BranchVault

## Problem Statement

The BranchVault class manages safety deposit boxes. Each box has a unique number, and may be assigned to a particular account number. This class demonstrates the bubble sort and the binary search. Code the following methods to complete this class.

* **SortBoxes()** – This private method performs a bubble sort of the array of SafetyDepositBox objects. The boxes are to be sorted in ascending order according to their box number.
* **Find()** – This public method performs a binary search of the SafetyDepositBox array for a particular box number. If the box number is not found, it returns a null.

## Supporting Classes

# PhoneBook

## Problem Statement

The PhoneBook class provides simple management of an array of PhoneNumber objects. The PhoneBook allows for adding and retrieving phone numbers; it supports search for phone numbers by the telephone number or by a last name. The PhoneBook is initialized with a physical limit for the arrays, and the actual number of entries in the phone book is tracked as phone numbers are added to the phone book. The AddPhoneNumber() method demonstrates adding to an array that is to be sorted. Code the following methods to complete this class.

* **BubbleSortByPhoneNumber()** – This private method performs a bubble sort of the array of PhoneNumber objects. They are to be sorted in ascending order, based on the telephone number.
* **InsertSortedByPhoneNumber()** – This private method will insert a PhoneNumber object into the array while preserving the sort order of the array (based on the array being sorted in ascending order by telephone number).

## Supporting Classes

# ClassList

## Problem Statement

The ClassList example maintains an array of students for a particular course by offering methods to add and remove students from a course. In addition, the ClassList ensures that students are not added to the course twice (based on the student’s Id). The internal array is of a fixed size (ClassList.CLASS\_LIMIT) as well as the logical size of the array. If the ClassList is provided with an array of students in its constructor, that list is “copied” to the internal array of the ClassList object (to ensure that the internal array has the appropriate physical size).

All of the Student objects in the ClassList are stored in a sorted array, based on the student Id. Code the following methods to complete this class.

* **AddInsertSortedByStudentId()** – This private method is called by both the AddStudent method and the constructor. It adds a student to the class list while maintaining a sort order (sorted by student Id in ascending order).
* **FindStudent()** – This public method performs a binary search of the array of Student objects. If the Student is found, this method will return that Student object, otherwise it will return null.
* **RemoveStudent()** – This public method will remove a particular Student from the array, while maintaining the sort order of the array. If the Student is found, this method will return that Student object, otherwise it will return a null.
* **GetStudents()** – This public method will generate and return a copy of the array of Student objects.

## Supporting Classes