HDSE 222

# Data Structures and Algorithms Practice

# P1 – Array As data structure

By,
Dr. Tharinda Nishantha Vidanagama.
Senior Lecturer,
CMIS.

# Evaluating your performance!


- Continuous assessment 20%

- Assignments 40%
- Semester-end Practical Exam 40%

100%

| | |
|---|---|
| Mid Semester Exam | 10% |
| Tutorials and Assignments | 10% |

# Java - Arrays

- Declaring an array

```
int[] arr;     // reference
arr = new int[100];          // make array
```

- Or

```
int[] arr = new int[100];
```

- Arrays have a “length” field to find the length of the array.

```
int arrLength = arr.length;
```

It is the total number of bytes allocated for the array and not the number of elements that you may have put in it!

# Java – Initializing and Accessing array elements

- Java integer array elements are automatically initialized to zero at declaration.
- Also it is possible to assign values at initialization.

```
int[] arr2 = {2, 5, 4, 3, 23, 43, 55, 33};
```

- Insertion: Assign a value to an array element

```
arr[55] = 23;
```

- Access the content of an array element.

```
int temp = arr[40];
```

# Array class

- Create a class to define the size and methods for your array
- Number of elements are kept in a separate variable for better operation.

```
class MyArray {
    private double[] a; // ref to array a
    private int nElems; // number of data items

    public MyArray(int max){ // constructor
        a = new double[max]; // create the array
        nElems = 0; // no items yet
    }
}
```

# Inserting elements to the array class

- Elements are inserted by using a special method rather than using direct assignment.

```
public void insert(double value) // put element into array
{
    a[nElems] = value; // insert it
    nElems++; // increment size
}
```

# Array traverse

- Access and display the array elements exactly once!

```
public void display(){ // displays array contents
   for (int j = 0; j < nElems; j++) // for each element,
         System.out.print(a[j] + " "); // display it
   System.out.println("");
}
```

# Search for an element

- Since the number of elements are known the search is only performed for the number of elements in the array, not for the entire length of the array.
- Searching element by element sequentially or linearly

```
public boolean find(double searchKey) { // find specified value
    int j;
    for (j = 0; j < nElems; j++) // for each element,
        if (a[j] == searchKey) // found item?
            break; // exit loop before end
    if (j == nElems) // gone to end?
        return false; // yes, can't find it
    else
        return true; // no, found it
} // end find()
```

# Deleting an Array element

- Search for the element first and if found, then shift the latter elements one position higher.

```
public boolean delete(double value) {
    int j;
    for (j = 0; j < nElems; j++) // look for it
        if (value == a[j])
            break;
    if (j == nElems) // can't find it
        return false;
    else // found it
    {
        for (int k = j; k < nElems; k++) // move higher ones down
            a[k] = a[k + 1];
        nElems--; // decrement size
        return true;
    }
} // end delete()
```

# Array Application class

- Create an application class to use the array class

```
class MyArrayApp {
   public static void main(String[] args) {
      int maxSize = 100; // array size
      MyArray arr; // reference to array
      arr = new MyArray(maxSize); // create the array

      arr.insert(77); // insert 10 items
      arr.insert(99);
      arr.insert(44);
      arr.insert(55);
   }
}
```

# Calling the methods in the Array Application class

- After inserting some elements to the array try calling the other methods used.

```
arr.display(); // display items

int searchKey = 35; // search for item
if (arr.find(searchKey))
    System.out.println("Found " + searchKey);
else
    System.out.println("Can't find " + searchKey);

arr.delete(00); // delete 3 items
arr.delete(55);
arr.delete(99);

arr.display(); // display items again
```