# Introduction to Data Structures and Algorithms

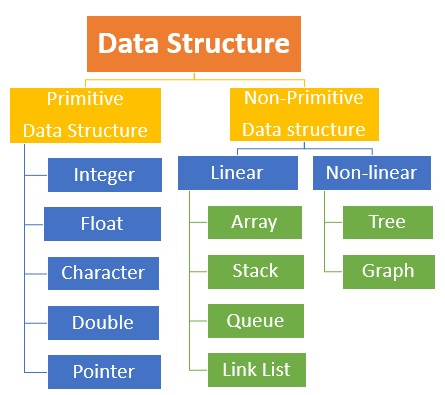
Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage. For example, we have some data which has, player's **name** "hamza" and **age** 26. Here "hamza" is of **String** data type and 26 is of **integer**data type.

We can organize this data as a record like **Player** record, which will have both player's name and age in it. Now we can collect and store player's records in a file or database as a data structure. **For example**: "hamza" 30, "Amir" 31, "ahmed" 33

If you are aware of Object Oriented programming concepts, then a class also does the same thing, it collects different type of data under one single entity. The only difference being, data structures provides for techniques to access and manipulate data efficiently.

In simple language, Data Structures are structures programmed to store ordered data, so that various operations can be performed on it easily. It represents the knowledge of data to be organized in memory. It should be designed and implemented in such a way that it reduces the complexity and increases the efficiency.

**Basic types of Data Structures**



1-arrays

**DESCRIPTION:**

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.To declare an array, define the variable type with **square brackets**:

String[] cars;  
We have now declared a variable that holds an array of strings. To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

Operations I have used in array project

* Insertion
* Deletion
* Searching
* Traversing

Time complexity

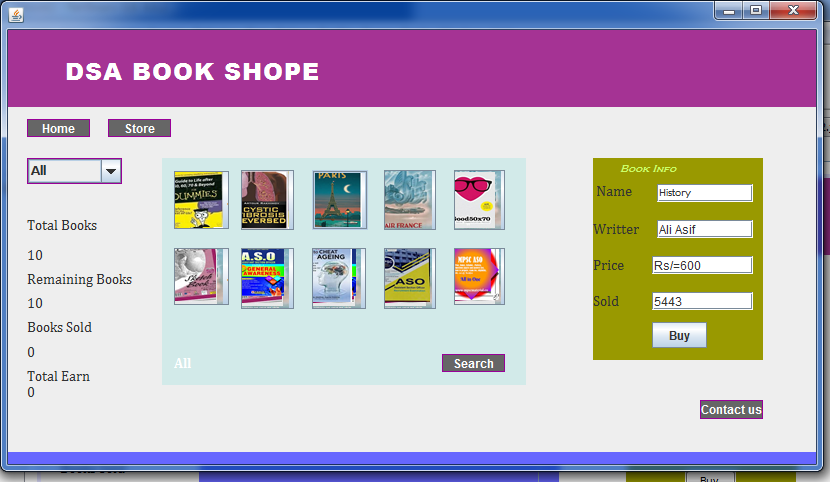
Search O(N)

Add O(N)

Delete O(N)

Insertion O(N)

**Snap shots:**

**Classes and Packages**

Import java.Util.\*;

Import java.awt.\*

Import java.awt.event.\*;

Import java.swing\*

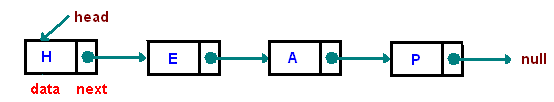
**Tool description**

Netbeans IDE.

**2-Linked list**

**DESCRIPTION:**

One disadvantage of using arrays to store data is that arrays are static structures and therefore cannot be easily extended or reduced to fit the data set. Arrays are also expensive to maintain new insertions and deletions. In this chapter we consider another data structure called Linked Lists that addresses some of the limitations of arrays.A linked list is a linear data structure where each element is a separate object.



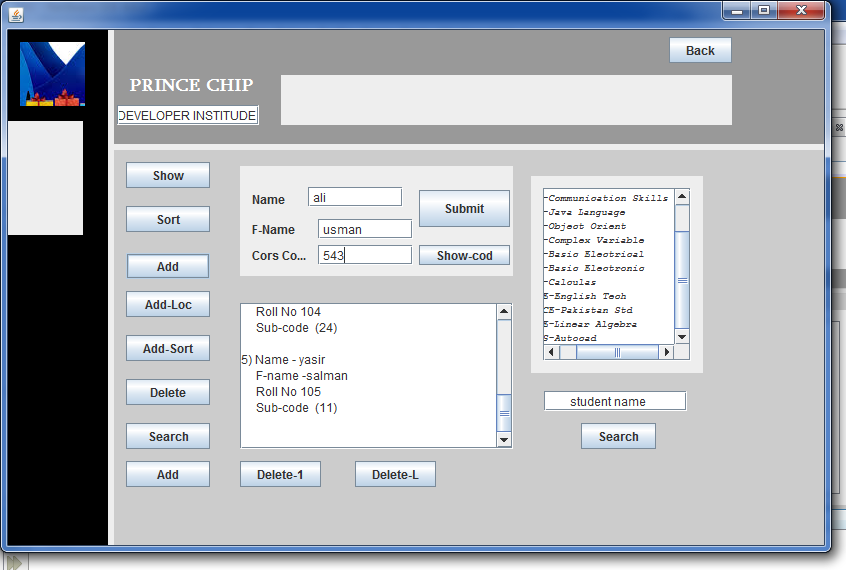
Each element (we will call it a **node**) of a list is comprising of two items - the data and a reference to the next node. The last node has a reference to null. The entry point into a linked list is called the **head** of the list. It should be noted that head is not a separate node, but the reference to the first node. If the list is empty then the head is a null reference.

One disadvantage of a linked list against an array is that it does not allow direct access to the individual elements. If you want to access a particular item then you have to start at the head and follow the references until you get to that item.

**Time Complexity of Linklist**

For searching O(n)

For Deleting O(1) **Snap shots:**

For Traversing O(n)

For Inserting O(1)

**Classes and Packages**

Import java.Util.\*;

Import java.awt.\*

Import java.awt.event.\*;

Import java.swing\*

**Tool description**

Netbeans IDE.

**3-Queue**

**DESCRIPTION:**

Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is open at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

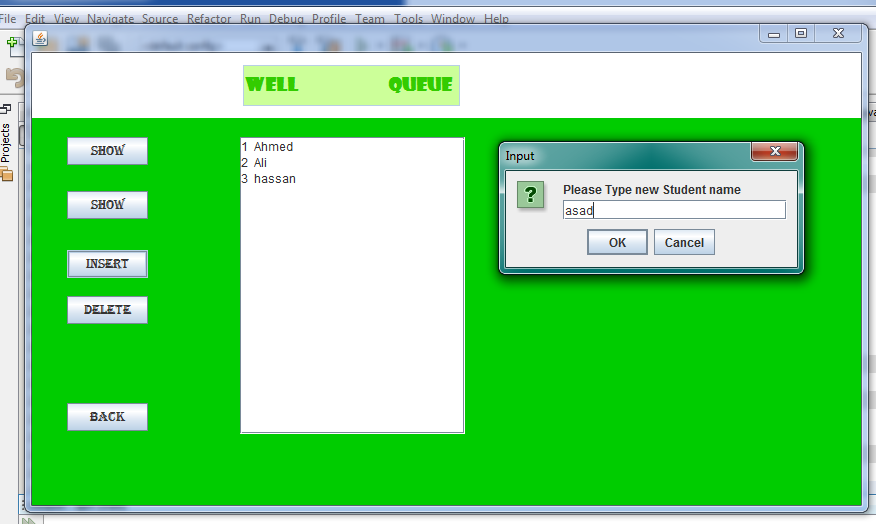
Queue representation

* **enqueue()** − add (store) an item to the queue.
* **dequeue()** − remove (access) an item from the queue.

Few more functions are required to make the above-mentioned queue operation efficient. These are −

* **peek()** − Gets the element at the front of the queue without removing it.
* **isfull()** − Checks if the queue is full.
* **isempty()** − Checks if the queue is empty.

**Time Complexity of Linklist**

For Deleting O(1) **Snap shots:**

For Traversing O(n)

For Inserting O(1)

**Classes and Packages**

Import java.Util.\*;

Import java.awt.\*

Import java.awt.event.\*;

Import java.swing\*

**Tool description**

Netbeans IDE.

**4-Stack**

**DESCRIPTION:** A Stack is a list of elements in which an element may be inserted ordeleted only at one end called the top of the stack Lists( LIFO (Last In,First Out) )

**Basic operations of stack**

“Push” is the term used to insert an element into a stack.

“Pop” is the term used to delete an element from stack. etc.

Array Implementation

⇐ Need to declare an array size ahead of time

⇐ Associated with each stack is TopOfStack for an empty stack, set TopOfStack to -1

Push

(1) Increment TopOfStack by 1.

(2) Set Stack[TopOfStack] = X

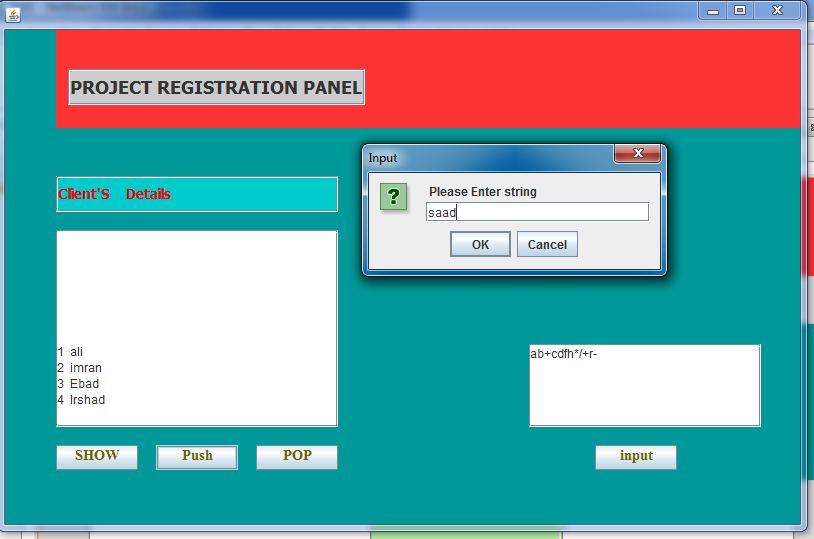
Pop

(1) Set return value to Stack[TopOfStack]

(2) Decrement TopOfStack by 1

Application of Stacks

* Postponed Decisions
* Quick Sort.
* Arithmetic Expressions(Postfix Notation)

 **Time Complexity of Linklist Snap shots:**

For Deleting O(1)

For Traversing O(n)

For Inserting O(1)

**Classes and Packages**

Import java.Util.\*;

Import java.awt.\*

Import java.awt.event.\*;

Import java.swing\*

**Tool description**

Netbeans IDE.

5-Hashing

**DESCRIPTION:**

Hashing is an important Data Structure which is designed to use a special function called the Hash function which is used to map a given value with a particular key for faster access of elements. The efficiency of mapping depends of the efficiency of the hash function used.

Let a hash function H(x) maps the value  at the index **x%10** in an Array. For example if the list of values is [11,12,13,14,15] it will be stored at positions {1,2,3,4,5} in the array or Hash table respectively.

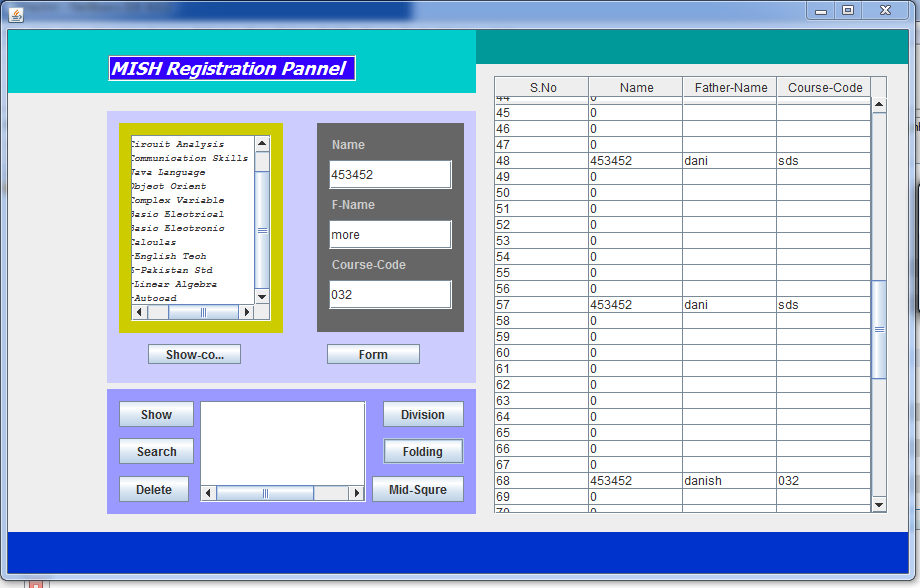
## Basic Operations

Following are the basic primary operations of a hash table.

* **Search** − Searches an element in a hash table.
* **Insert** − inserts an element in a hash table.
* **delete** − Deletes an element from a hash table.

**Time Complexity of Linklist**

For searching O(1) **Snap shots:**

For Deleting O(1)

For Traversing O(n)

For Inserting O(1)

**Classes and Packages**

Import java.Util.\*;

Import java.awt.\*

Import java.awt.event.\*;

Import java.swing\*

**Tool description**

Netbeans IDE.

Refrence

* <https://www.tutorialspoint.com/index.htm>
* <https://www.oracle.com/index.html>
* youtube