**Stacks:**

A stack is a data structure that allows data to be inserted (a 'push' operation), and removed (a 'pop' operation). Many stacks also support a read ahead (a 'peek' operation), which reads data without removing it. A stack is a LIFO-queue, meaning that the last data to be inserted will be the first data to be removed.

When we insert data into a stack, it is placed at the head of a queue. This means that when we remove data, it will be in reverse order. Adding 1,2,3,4 in that order will return 4,3,2,1 when the items are removed from the stack. Stacks aren't the most frequently used data structure, but they are extremely useful for certain tasks.

Think of Stacks like, well, a stack of books. You pile books on top of each other. To get the book at the bottom, you have to remove all of the books that are on top of it. Remember, that the first book that was placed (inserted) in the pile (the Stack) would be the book at the bottom of the pile.

import java.util.Stack;

public class StackDemo

{

public static void main(String args[])

{

// Create a new, empty stack

Stack<Integer> lifo = new Stack<Integer>();

// Let's add some items to it

for (int i = 1; i <= 10; i++)

{

lifo.push ( new Integer(i) );

}

System.out.println("The size of the Stack is "+lifo.size()+".");

System.out.println("What's on top? " +lifo.peek());

System.out.println("Unstack the stack!");

while (!lifo.empty())

{

System.out.print ( lifo.pop() );

if(!lifo.empty())

System.out.print ( ',' );

else System.out.println();

}

}

}

**Stack commands:**

push() – used for insertion into the Stack

peek() – used to look at stack to identify lead value without removal

pop() – used to remove lead value from the Stack

isEmpty() – used to check if Stack is empty

size() – used to determine Stack size