Data Structures - Stacks

Dr. TGI Fernando [1] [2]

[1] Email: tgi.fernando@gmail.com
[2] URL: http://tgifernando.wordpress.com/

Stacks and Queues

Arrays, linked lists, trees - suitable for database applications

- **E.g.** Personal records, inventories, etc.
- Corresponds to real-world objects or activities
- Provide facilities to insert, delete and search for items.

How stacks and queues are different from arrays, linked lists and trees?

1. Programmer's tools

- Created to carry out a particular task during the operation of a program.
- When the task is completed, they are discarded.

Stacks and Queues (Contd.)

2. Restricted access

- In an array or a linked list, any item can be accessed (no restrictions).
- In stacks and queues, however, access is restricted: only one item can be read or removed at a given time.
- The interface enforces this restricted access.
- Access to the other items is (in theory) not allowed.

3. More abstract

- Stacks and queues are defined primarily by their interface: i.e. permissible operations allowed on them.
- Mechanism used to implement them are not visible to their users.
  E.g. A stack can be implemented as an array or a linked list.

Stacks

A stack allows access to only one data item: the last item inserted. If you remove this item, you can access the next-to-last item inserted, and so on.

E.g. Converting a decimal number into a binary number.

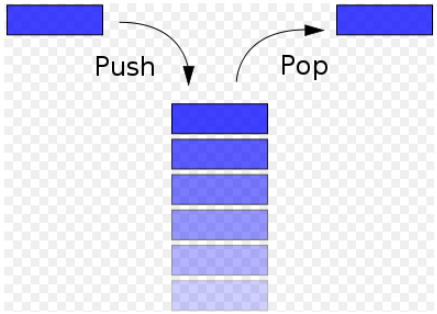
1. Read a number.
2. Iteration (while number is greater than zero)
   2.1 Find out the remainder after dividing the number by 2
   2.2 Print the remainder
   2.3 Divide the number by 2
3. End the iteration

However, there is a problem with this logic. Suppose the number, whose binary form we want to find is 23. Using this logic, we get the result as 11101, instead of getting 10111.

The solution to this problem is use of a stack to store the digits.

Primary stack operations

Push - Placing a data on top of the stack.
Pop - Removing the top element from the stack.



A stack is said to be a **Last-In-First-Out (LIFO)** storage mechanism because the last item inserted is the first one to be removed.

Array implementation of a stack

Although, this implementation uses an array, the users can't access elements using the array index.

```
class StackArray {
    private int maxSize; // size of stack array
    private int [] stackData;
    private int top; // top of stack
    //--------------------------------------------
    public StackArray (int s) { // constructor
        maxSize = s; // set array size
        stackData = new int[maxSize]; // create array
        top = -1; // no items yet
    }
    //--------------------------------------------
    // ...
}
```

top - is an `int` that holds the index of the top element.

Implementation of stack operations

Push

- Increment `top` by `1`.
- Copy the new item into the cell at `top`.
- If the stack is full and try to push another item, an error message `'Can't insert: stack is full'` will be displayed.

Pop

- First, the item is removed from the cell pointed to by `top`.
- Then, `top` is decremented by `1`.
- **Output** - Returns removed data item.
- If the stack is empty and you try to pop an item, you'll get the `Can't pop: stack is empty` message.

Peek - Read the value at `top` without removing it.

The `StackArray` class

```
class StackArray {
    private int maxSize; // size of stack array
    private int [] stackData;
    private int top; // top of stack
    //--------------------------------------------
    public StackArray (int s) {...} // constructor
    public void push (int j) {...} // put item on top of
    stack
    public int pop () {...} // take item from top of stack
    public int peek () {...} // peek at top of stack
    public boolean isEmpty () {...} // true if stack is
    empty
    public boolean isFull () {...}
} // end class StackArray
```

Lab Work 06 - StackArray class

1. Complete the coding of the `StackArray` class.
2. Write a Java program to convert a decimal number into a binary number.