Data Structures In Java

In this section of notes you will learn about two common types of data structures:

Queues

Stacks

Data Structures: Description

A composite type that has a set of basic operations that may be performed on instances of that type:

- ❖The type may be a part of the programming language
    - e.g., arrays are a basic part of the Pascal language
    - Some basic operations on arrays include: adding, deleting or modifying array elements.
- ❖The type may also be created by the programmer
    - e.g. linked lists must be defined by the programmer in Pascal
    - Some basic linked list operations include: creating a new list, adding, deleting and modifying nodes on that list.

Data Structures To Be Covered

Queues

Stacks

Characteristics:

- Both are lists
- The difference is in their behaviour

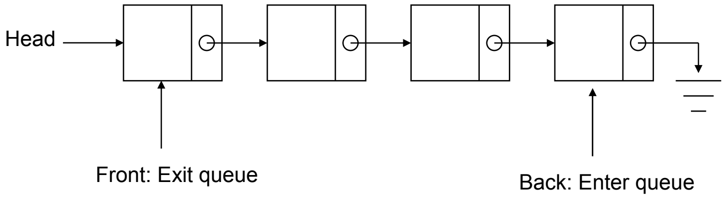
Queues

A list where additions occur only at one end of the list and deletions occur only at the other end.

Front: Exit queue

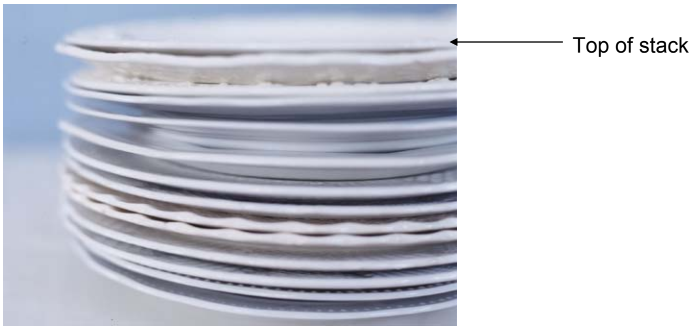
Back: Enter queue

Implementing Queues



Stacks

A list where additions and deletions are made at only one end of the list.



Common Stack Operations

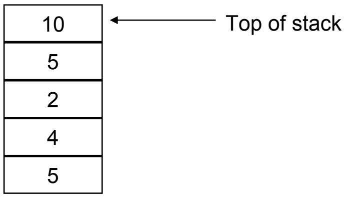
Push

Pop

Peek

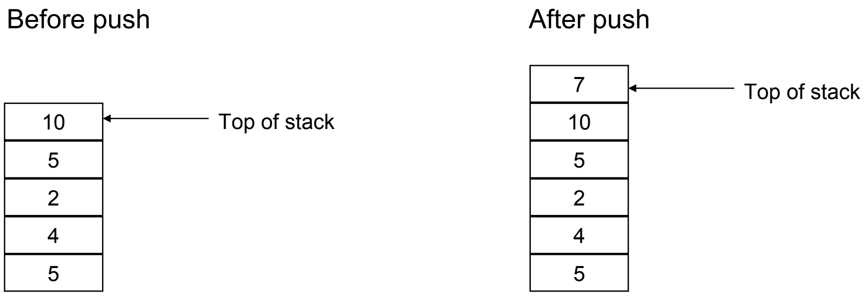
Push Operation

Adding an item to the top of the stack



Push Operation

"7" has been added to the stack and this new item becomes the top of the stack.



Before push

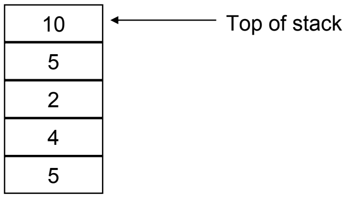
10 ← Top of stack
5
2
4
5

After push

7 ← Top of stack
10
5
2
4
5

Pop Operation

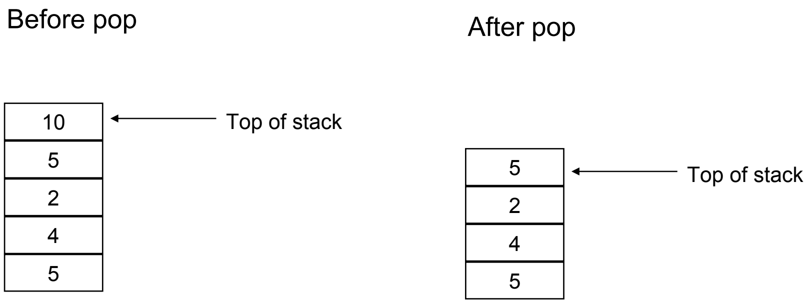
Removing an item from the top of the stack



10 ← Top of stack
5
2
4
5

Pop Operation

“10” has been removed and “5” becomes the new top of the stack.


Before pop

10
5
2
4
5

Top of stack

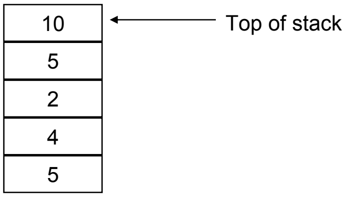
After pop

5
2
4
5

Top of stack

Peek Operation

Examine the item at the top of the stack without removing it


10
5
2
4
5

Top of stack

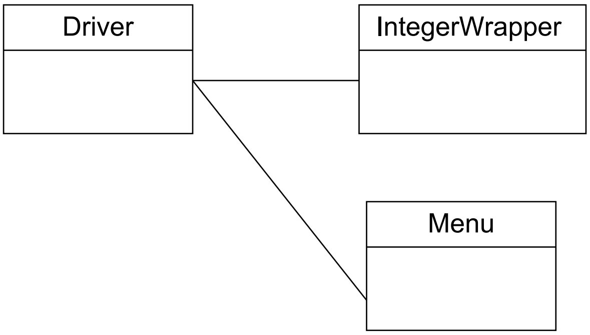
Java Implementation Of A Stack

```
// It's part of the legacy Java code but it still helps illustrate how the
// implementation works.

// Use of the Stack class requires the statement:
import java.util.*;

class Stack
{
   public boolean empty ();
   public Object peek ();
   public Object pop ();
   public Object push ();
   public int search (Object o);
}
```

Example Using The Java Stack



Driver

IntegerWrapper

Menu

The Driver Class

```
import tio.*;
import java.util.*;

class Driver
{
   public static void main (String [] argv)
   {
      int i, noElements, tempNum;
      IntegerWrapper rapper;
      Stack s1;
      int menuSelection;
      boolean quitMenu = false;
      Menu m = new Menu ();

      System.out.print("Enter desired number of elements: ");
      noElements = Console.in.readInt();
      Console.in.readChar();
```

The Driver Class (2)

```
s1 = new Stack ();
System.out.println("Displaying elements in the order they were added...");
for (i = 0; i < noElements; i++)
{
  rapper = new IntegerWrapper();
  System.out.print("Value of element " + i + "..." + rapper.getNum());
  if (i < (noElements-1))
     System.out.println();
  else
     System.out.println("\t<== Top of stack");
  s1.push(rapper);
```

The Driver Class (3)

```
while (quitMenu != true)
{
   m.displayMenu ();
   menuSelection = m.getSelection();
   Console.in.readChar();

   switch (menuSelection)
   {
      // Pop element
      case 1:
      if (s1.empty() == false)
      {
           rapper = (IntegerWrapper) s1.pop();
           System.out.println();
           System.out.println("Value of popped element: " + rapper.getNum());
           System.out.println();
      }
```

The Driver Class (4)

```
      else
      {
           System.out.println();
           System.out.println("Stack is empty: No elements to pop!");
           System.out.println();
      }
      break;

      // Push element
      case 2:
      System.out.println();
      System.out.print("Enter value of element to push onto stack: ");
      tempNum = Console.in.readInt();
      s1.push(new IntegerWrapper(tempNum));
      break;
```

Driver Class (5)

```
case 3:
rapper = (IntegerWrapper) s1.peek ();
System.out.println();
System.out.println(“Element at the top of stack:" + rapper.getNum());
System.out.println();
break;

case 4:
System.out.println("Displaying elements in the order of the stack");
while (s1.empty() == false)
{
   rapper = (IntegerWrapper) s1.pop();
   System.out.println("\tValue of popped element: " + rapper.getNum());
}
System.out.println();
break;
```

Driver Class (6)

```
                 case 5:
                quitMenu = true;
                break;
             } // End of switch
         } // End of while
         System.out.println("Exiting program.");
     }  // End of main
 }  // End of class Driver
```

The IntegerWrapper Class

```
class IntegerWrapper
{
   private int num;

   public IntegerWrapper () {num = (int) (Math.random() * 100); }

   public IntegerWrapper (int no) { num = no; }

   public void setNum (int no) { num = no; }

   public int getNum () { return num; }
}
```

The Menu Class

```
class Menu
{
   public void displayMenu ()
   {
      System.out.println("MENU OPTIONS");
      System.out.println("1: Pop object off stack and display object");
      System.out.println("2: Push new object onto stack");
      System.out.println("3: Peek at object at the top of stack but don't remove");
      System.out.println("4: Pop entire stack and view objects as they are popped");
      System.out.println("5: Quit program");
   }

   public int getSelection ()
   {
      int menuSelection;
      System.out.print("Enter menu selection: ");
      menuSelection = Console.in.readInt();
      return menuSelection;
   }
}
```