Auto-resize stack

Saturday, September 10, 2022 12:25 PM

A stack is a collection of values that follows the rule of LIFO. A stack can automatically resize itself. Below is a full implementation of a Stack class, that allows client to pop or push an item and handle resize automatically.

ALGORITHM 1.1 Pushdown (LIFO) stack (resizing array implementation)

```
import java.util.Iterator;
public class ResizingArrayStack<Item> implements Iterable<Item>
{
   private Item[] a = (Item[]) new Object[1];  // stack items
   private int N = 0;                          // number of items

   public boolean isEmpty()  {  return N == 0; }
   public int size()         {  return N;      }

   private void resize(int max)
   {  // Move stack to a new array of size max.
      Item[] temp = (Item[]) new Object[max];
      for (int i = 0; i < N; i++)
         temp[i] = a[i];
      a = temp;
   }

   public void push(Item item)
   {  // Add item to top of stack.
      if (N == a.length) resize(2*a.length);
      a[N++] = item;
   }

   public Item pop()
   {  // Remove item from top of stack.
      Item item = a[--N];
      a[N] = null;  // Avoid loitering (see text).
      if (N > 0 && N == a.length/4) resize(a.length/2);
      return item;
   }

   public Iterator<Item> iterator()
    {  return new ReverseArrayIterator();  }

   private class ReverseArrayIterator implements Iterator<Item>
   {  // Support LIFO iteration.
      private int i = N;
      public boolean hasNext() {  return i > 0;   }
      public    Item next()    {  return a[--i];  }
      public    void remove()  {                  }
   }
}
```

Required methods: iterator()

An interface that allows the instance of Stack class to be iterable, meaning the values in a Stack object can be displayed in the client program with a **loop**, such as foreach statement to print out all of the elements of a Stack object.

new Object[1] means to construct an Object array of size 1
(Item[]) new Object[1] means to cast the Object array to an Item array (Item is a placeholder for the actual type determined by the client)

push() call the resize() to double the size of the array by creating a new array call temp and copy the current elements in a to temp, then assign temp to a. Then the new item is added to array a, and N is incremented by 1.

pop() copy the last element of a in item by using --N to decrease the value of N by 1 before copying the last element to variable item. a[N] = null means to nullify a[N], so the memory of that element will be free, and this memory can be assign to other thing for the program to use. Wihout this statement, the reference to that object will remain there, preventing its garbage collection. The appropriate time to call the resize() in pop() is when the number of elements in the array is **one fourth** the array size, at this point, **it's absolutely safe to delete the other half of the array.** We still have the remaining 1/4 empty space for the client to push new items to the array. item is returned

This generic, iterable implementation of our Stack API is a model for collection ADTs that keep items in an array. It resizes the array to keep the array size within a constant factor of the stack size.

iterator() should return an iterator<item> object. Then, all the items in this object will be iterable.
An Iterator<item> object should have the method hasNext() to test whether or not the next

element exists. As well as an next() to return the next element in the object and a remove(). Therefore, we need to create a class that implement these methods. To include those specific methods in a class, we use the Iterator<Item> interface (An interface is a mechanism in Java that express the idea that a class impelments a specific method.) Since we want the object to have stack property which it should conform LIFO data structure, the elements in the object must iterates backwards. So the next() returns a[--i]