CSP2348/CSP5243 Data Structures

Review Questions 09

Topics: Stacks and Vector

1. Would it make sense to call a stack a FILO (first-in-last-out) structure?

2. What is the difference between a SLL and a stack?

3. Trace the following code, showing the contents of the stack after each invocation [note: `push()= addLast(); pop() = removeLast()`]

```
Stack stack = new Stack();
stack.push("A");
stack.push("B");
stack.push("C");
stack.pop();
stack.pop();
stack.push("D");
stack.push("E");
stack.push("F");
stack.pop();
stack.push("G");
stack.pop();
stack.pop();
stack.pop();
```

4. The following lists two implementations of the Stack ADT. The first one is an array based implementation and the second is an SSL based one.

   In the first implementation, the method `addLast()` is implemented by calling another method `expand()`, while in the second implementation the method `addLast()` is implemented without calling any other methods. Explain what causes this difference in the implementation of the same ADT, and then analyse the performances of these two implementations in terms of big-*O* notation.

//Array-based implementation of Stack (see Java Collection, program 6.8)

```
public class ArrayStack implements Stack {
   //Each ArrayStack object is a bounded stack whose elements are objects.
   // This stack is represented as follows: its depth is held in depth, and its
   // elements occupy the subarry elems[0…depth].
   Private Object [] elems;
   Private int depth;
   //////////////////constructor /////////////////
    public ArrayStack (int maxDepth) {
   //Construct a stack, initially empty, whose depth will be bounded by maxDepth.
      elems = new Object [maxDepth];
   // …. All components of elems are initially null.
      depth = 0;
   }
   ////////////// accessor ////////////
   Public Boolean isEmpty () {
   // Return true if and only if this stack is empty.
      return (depth == 0);
   }
   public Object getLast() {
   //Return the element at the top of this stack. Throw a
   //NoSuchElementException if this stack is empty.
      if (depth == 0)
         throw new NoSuchElementException();
      return elems [depth-1];
   }
   //////////////// Transformers ////////////
   public void clear () {
   //Make this stack empty;
       for (i = 0; i < depth; i++)
         elems[i] = null;
       depth = 0;
   }

   public void addLast(Objet elem) {
   // Add elem as the top element on this stack.
       if (depth == elems.length)
             expand();
          elems[depth++] = elem;
   }
   public void removeLast() {
   // Remove and return at the top element on this stack. Throw a
   //NoSuchElementException if this stack is empty.
        if (depth == 0)
           throw new NoSuchElementException();
        Object topElem = elems[--depth];
         elems[depth] = null;
        return topEms ;
   }
   //////////Auxiliary method////////////
   private void expand() {
   // Make the elems array longer;
       Object [] newElems = new Object[2*elemg.length];
       for (int i = 0; i < depth; i++)
           newElems[i] = elems[i];
       elems =newElems;
   }
}
```

```
//SSL-based implementation of Stack (see Java Collection, program 6.10)

public class LinkedStack implements Stack {
    //Each LinkedStack object is a unbounded stack whose elements are objects.
    // This stack is represented as follows: top is a link to the first node of an
    // SSL containing the stack's elements, in top-to-bottom order.
    Private SLLNode top;
    /////////////////// Constructor   /////////////////
     public LinkedStack () {
    //Construct a stack, initially empty.
        top = null;
    }
    ////////////// accessor ////////////
    Public Boolean isEmpty () {
    // Return true if and only if this stack is empty.
        return (top == null);
    }
    public Object getLast () {
    //Return the element at the top of this stack. Throw a
    //NoSuchElementException if this stack is empty.
        if (top == null)
            throw new NonSuchElementException();
        return top.element;
    }
    //////////////// Transformers ////////////
    public void clear () {
    //Make this stack empty;
         top = null;
    }

    public void addLast(Objet elem) {
    // Add elem as the top element on this stack.
         top = new SSLNode (elem, top);
    }
    public void removeLast() {
    // Remove and return the top element on this stack. Throw a
    //NoSuchElementException if this stack is empty.
          if (top == null)
                throw new NoSuchElementException();
          Object topElem = top.element;
           top = top.succ;
          return topEms ;
    }
    //////////Auxiliary method////////////
    private void expand() {
    // Make the elems array longer;
         Object [] newElems = new Object[2*elemg.length];
         for (int i = 0; i < depth; i++)
              newElems[i] = elems[i];
         elems =newElems;
    }
}
```