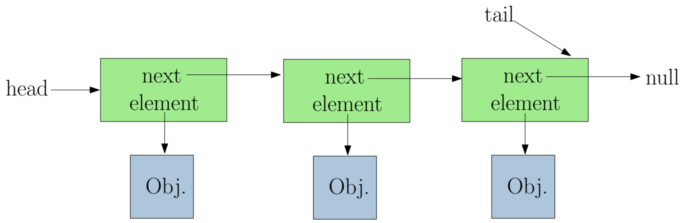
ADT Review and Stacks

Daniel Archambault

Previously in CS-115



Linked Lists: Know the Ends. Know how to get to the middle.

Previously in CS-115

- What are the two attributes of Link in a linked list?

Previously in CS-115

- What are the two attributes of Link in a linked list?
- What are the two attributes of a linked list?

Previously in CS-115

- What are the two attributes of Link in a linked list?
- What are the two attributes of a linked list?
- Name an advantage of a linked list?

Previously in CS-115

- What are the two attributes of Link in a linked list?
- What are the two attributes of a linked list?
- Name an advantage of a linked list?
- Name a disadvantage of a linked list?

Previously in CS-115

- What are the two attributes of Link in a linked list?
- What are the two attributes of a linked list?
- Name an advantage of a linked list?
- Name a disadvantage of a linked list?
- How do you get to an item that is not the ends?

Previously in CS-115

- What are the two attributes of Link in a linked list?
- What are the two attributes of a linked list?
- Name an advantage of a linked list?
- Name a disadvantage of a linked list?
- How do you get to an item that is not the ends?
- Can queues be implemented using arrays?

Previously in CS-115

- What are the two attributes of Link in a linked list?
- What are the two attributes of a linked list?
- Name an advantage of a linked list?
- Name a disadvantage of a linked list?
- How do you get to an item that is not the ends?
- Can queues be implemented using arrays?
- What is the difference between an ADT and data structure in the context of Queues?

Previously in CS-115

- We now look at an ADT and a possible implementation.

Stacks

More about ADTs

- Implementation not specified!
- ADT specifies the behaviour of the software construct
- The data structure implements those behaviours
    - Which data structures do we have to implement a Queue?
- For ADTs, think about state and operations on the state
    - don't think about main or the rest of the program

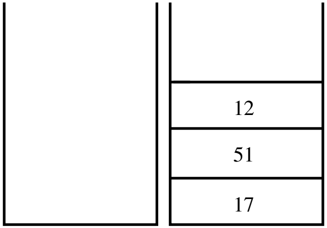
A Stack Interface

- *Last in First Out*

```
public interface Stack {
  public boolean isEmpty ();
  public void push (Object newItem);
  public void pop ();
  public Object peek ();
}
```

`isEmpty` behaviour

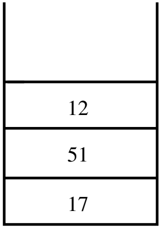
- Returns true if there are zero elements in the stack
- Otherwise, returns false



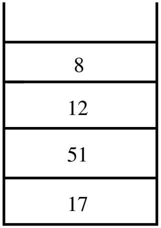
(a) true (b) false

`push` behaviour

- Adds an item to the top of the stack



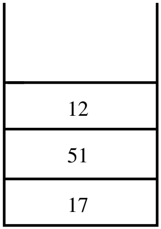
(c) before push of 8



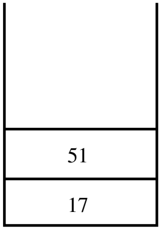
(d) after push

`pop` behaviour

- Removes an item from the top of the stack



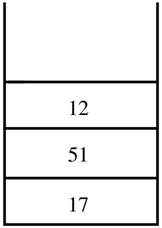
(e) before pop 12



(f) after pop

`peek` behaviour

- Returns the top element of the stack



(g) returns 12

Implementation of Stack

- To turn a ADT Stack into a Stack data structure we can choose either an array/ArrayList, or linked list
    - both linked list and array/ArrayList are about same difficulty
- I'll explain this implementation with a linked list first
- Then an ArrayList second

Attributes of Stack Implemented with Linked List

```
 public class Stack {
   private Link head;
   private Link tail;

    public Stack () {
      head = null;
      tail = null;
   }
}
```

`isEmpty` implementation

- Returns true if there are zero elements in the stack
- Otherwise, returns false
- Really, could just check head, but this prevents bugs

```
...
return ((this.head == null) && (this.tail == null));
...
```

`push` implementation

- Adds an item to the top of the stack
- It is the end of the linked list
- If tail is null, you need to set head and tail...

```
...
Link newNode = new Link (element, head);
this.head = newNode;
...
```

pop implementation

- Removes an item from the top of the stack
- Simply remove the link from the back

```
...
if this.isEmpty () {
   throw new NoSuchElementException ();
}
this.head = this.head.next;
...
```

- Check if tail is null. If so, set head null. (stack empty)

`peek` behaviour

- Returns the top element of the stack

```
...
if this.isEmpty () {
   throw new NoSuchElementException ();
}
return this.head.element;
...
```

Implementation of Stack: ArrayList

- ADT says nothing about how implemented
- It only specifies what is implemented.
- We can also implement a stack with an `ArrayList`
  - Why is this easier than using an array?

Attributes of Stack Implemented with ArrayList

- I'm using generics because it is convenient
- `Object` can also be used.

```
public class Stack<T> {
  private ArrayList<T> stack;

   public Stack () {
     this.stack = new ArrayList<T> ();
  }
}
```

`isEmpty` implementation

- Returns true if there are zero elements in the stack
- Otherwise, returns false
- Just check the size of the array list

```
 public boolean isEmpty () {
   return this.stack.size () == 0;
}
```

- could also use isEmpty () of ArrayList.

`push` implementation

- Adds an item to the top of the stack
- Just use the add operation provided

```
public void push (T e) {
  this.stack.add (e);
}
```

`pop` implementation

- Removes an item from the top of the stack
- Use the remove method to delete last element

```
public void pop () {
  if (this.stack.isEmpty ()) {
  throw new NoSuchElementException ();
  }
  this.stack.remove (this.stack.size () - 1);
}
```

`peek` behaviour

- Returns the top element of the stack

```
 public T peek () {
   if (this.stack.isEmpty ()) {
   throw new NoSuchElementException ();
   }
   return this.stack.get (this.stack.size() - 1);
}
```