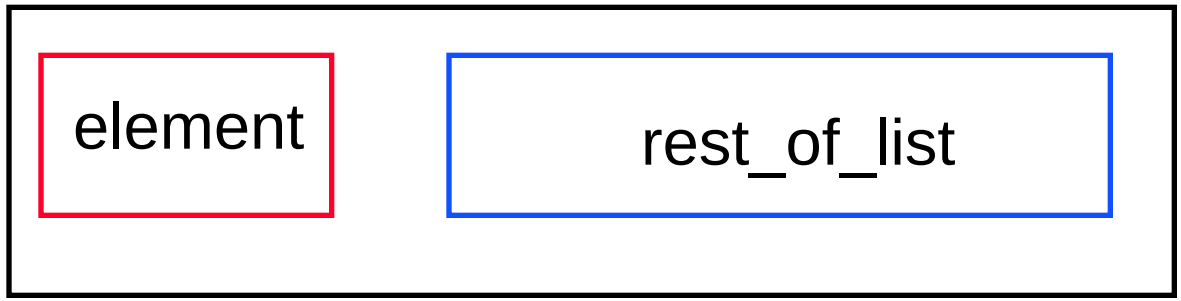
Stacks

- **Common data structures - useful for organizing data for specific tasks**
- **Lists**
- **Stacks - an Abstract Data Type**
  - **Class interface**
  - **Polymorphism**
  - **Use of List as representation of Stacks**
  - **Pop versus Peek**

Lists

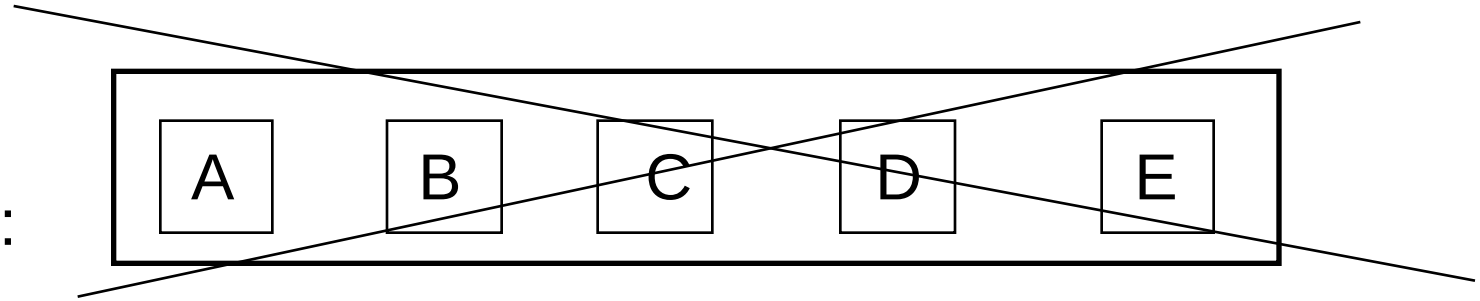
- **A list is a sequence of objects**
  - **Bad view for thinking about operations on lists**
- **A list is a pair, a first element and a rest_of_list, which is a sublist**



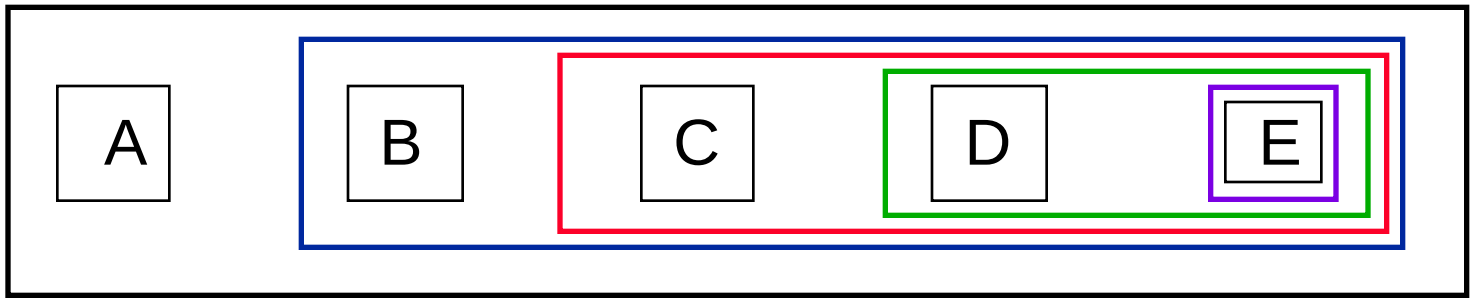
Lists

cs111.util.List.*

not this:



Preferable



Details here are hidden by List class implementation!

Lists

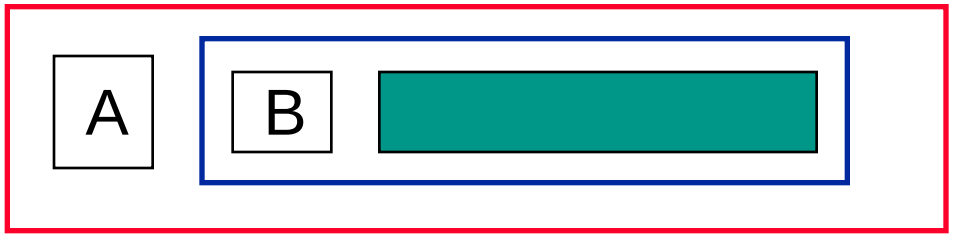
```
public class List extends Object{
   protected Object info;//field is accessible only
   protected List subList;//by classes in same package
             //means field is private to package
public List{
   info = null;
   subList = null;
}
public List (Object element, List oldList){
   info = element;
   subList = oldList;
}
```

List Construction

element: A

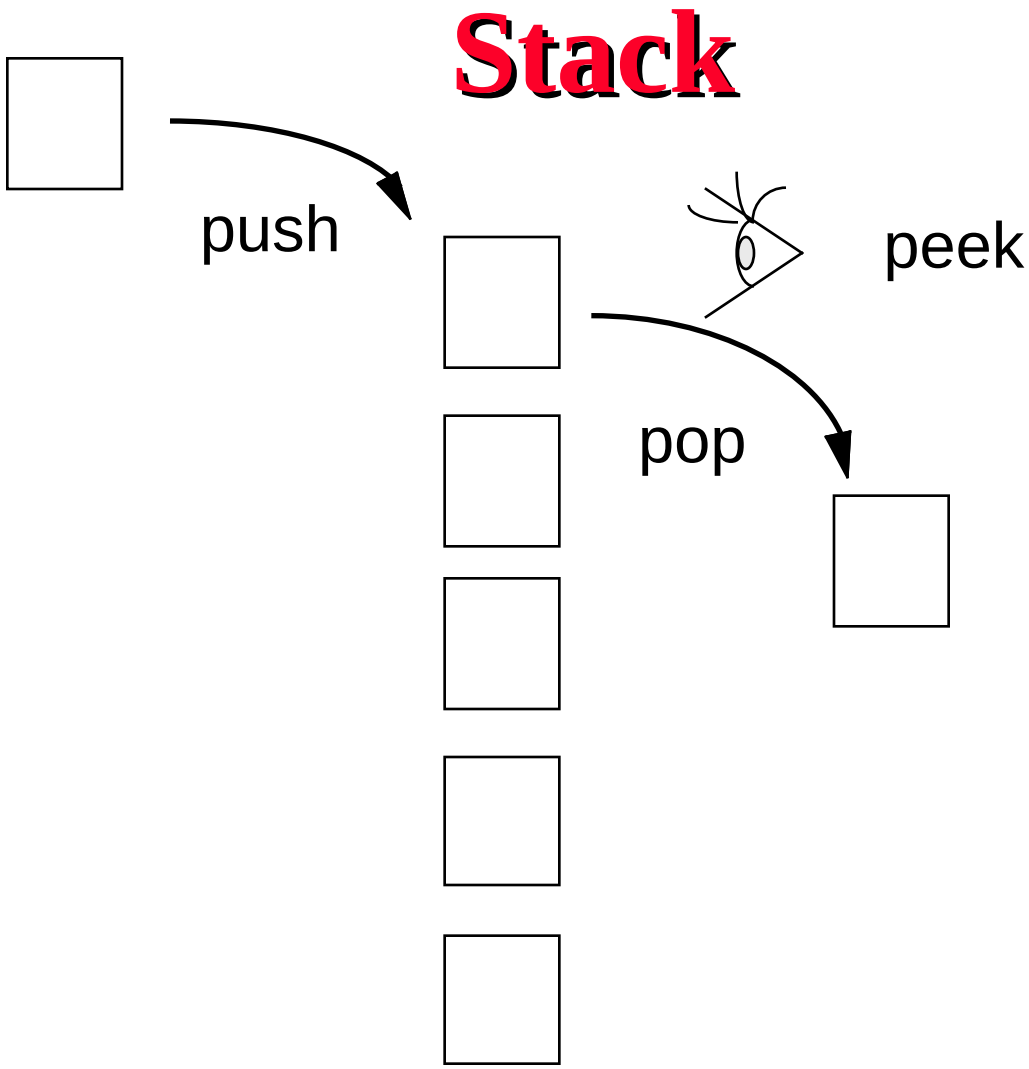
oldList:

result:



new list

Stack



Stacks

- **Stacks in real-life**
  - **Redial button on telephone - calls the last number dialed**
  - ***history* (*his*) command in Unix (!! executes your last typed command)**
  - **Job layoffs of people with least seniority**
  - **Pile of plates in restaurant**

Stacks - an Abstract Data Type

- **Defined in terms of necessary operations**
- **Abstract Data Type (ADT)**
  - **Something defined by its behavior**
  - **Doesn't matter how these behaviors are implemented as long as semantics are preserved**
- **Implementation "protected" from disturbance by a user - *encapsulation* or *data hiding***

Stack Class Interface

- **Instance variables:**
  - **private List top**
  - **private int length**
- **Instance methods:**
- **public Stack() //constructor**
- **public int getLength() //# of elements**
- **public boolean empty()**
- **public String toString()**
- **public Enumeration getEnumeration()**

Stack Methods

```
public Stack() {//empty stack is top as null List
   top = null;
   length = 0;
}
public int getLength(){//observer method
   return length;
}
public boolean empty(){//true only if length!=0
   return (length == 0);
}
```

Stack Interface

- **public void push (Object newItem)**
  - **adds element newItem to stack**
  - **polymorphic abstract data type (ADT)**
- **public Object pop() throws StackException**
  - **removes element from Stack and returns it**
  - **polymorphic**
- **public Object peek() throws StackException**
  - **allows examination of top element on Stack without removing it**
  - **polymorphic**

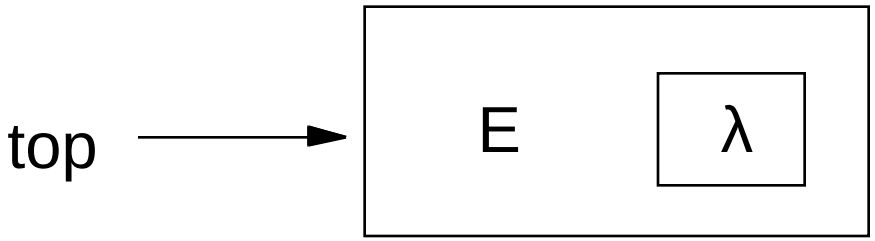
Stack Class: How to build?

- **How to represent Stacks?**
  - **Use List class (first element, rest_of_list) to hold elements in a stack**
- **Potential special cases**
  - **Pop off or peek at an empty stack**
  - **Push onto an empty stack**
  - **Both can be handled by encoding the empty stack as top == null and length == 0**
- **Can use length== 0 to check for empty stack**

Push onto empty stack

Initially,
top is null
length is 0

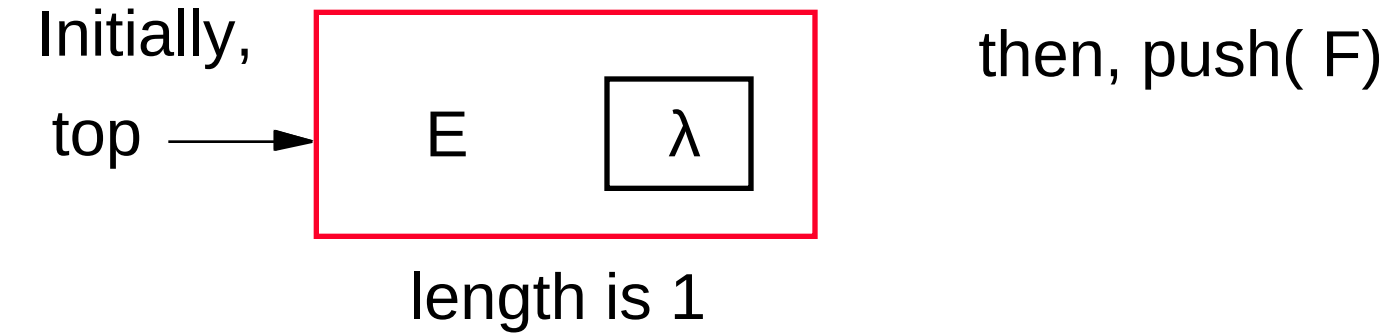
Perform push(E)

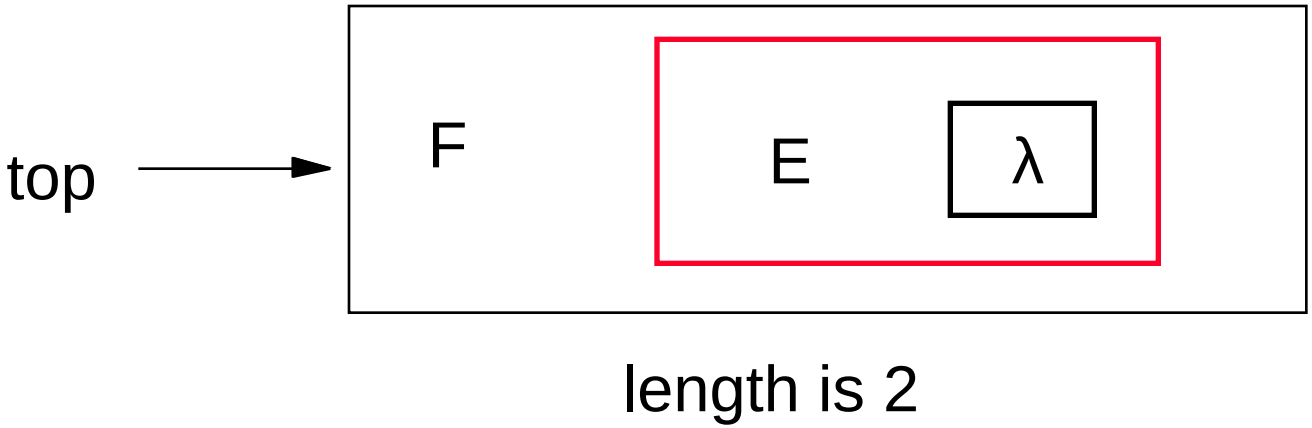


length is 1

```
List nl = new List(newItem, top)
top = nl;
```

Push onto non-empty Stack





Push Method

```
//create new List with old List as subList and
//newItem as first element
public void push(Object newItem){
   List nl = new List(newItem, top);
   top = nl;
   length++;
}
```

Pop Method

```
public Object pop() throws StackException{
  if (empty()) throw new StackException
      ("Attempt to pop from empty Stack");
  Object ret = top.info;
  top = top.subList;
  length--;
  return ret;
}
```

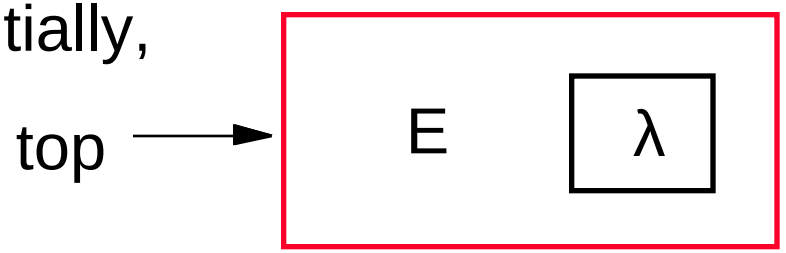
Pop off empty stack

Initially,
top is null
length is 0

empty() yields true
method throws a StackException.

Pop off non-empty stack

Initially,



length is 1

pop()

return E
top is null
length is 0

```
Object ret = top.info;
top = top.subList;
length--;
return ret;
```

User-defined Exception

```
public class StackException extends Exception{
   String msg;
   StackException (String str){
       msg = str;
   }
}
```

- Define as extension of built-in class Exception in java.lang.*
- Pass StackException object with private String instance variable to exception handler for possible printing
- No handler in Stack class means user of Stack class can handle or pass along to default handler in class Object

Pop( ) versus Peek( )

```
public Object pop() throws StackException{
   if (empty()) throw new StackException
       ("Attempt to pop from empty Stack");
   Object ret = top.info;
   top = top.subList;
   length--;
   return ret;
}
public Object peek() throws StackException{
   if (empty()) throw new StackException
       ("Attempt to peek at an empty Stack");
   return top.info;
}
```

changes the stack

does not change the stack

toString Method

```
//uses toString() method in List to build String rep
//of contents of Stack
public String toString(){
  String ret = “Stack length is “ + length + “\n”;
  return ret + “stack is:  “ + top.toString;
}
```