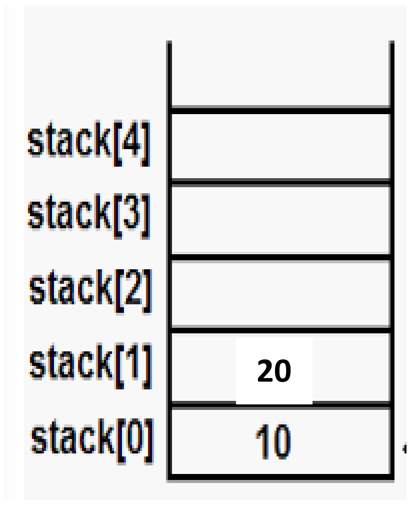
Stack- array

Methods push, pop, peek.

We have created a class of employees or as here we want to store some integers. Where are we going to store them?

We decide we need a stack structure and are going to use an array.

What is the weakness of this idea? How can we overcome it?



←-top

We will make the top pointer variable be the first available place to push onto the stack.

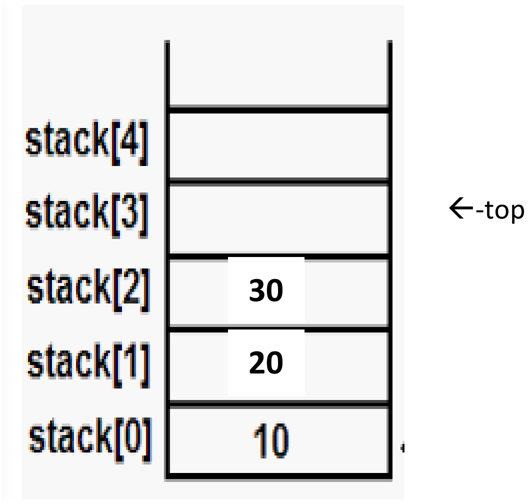
PUSH

Need to check if the stack is full.

To do that check if top = stack (array) length

If it is, we are going to give a message that stack is full or create a new larger array and copy the old one over.

otherwise stack[top ++) = 30 // as the ++follows top 30 goes where top originally pointed to and THEN top is incremented to point to stack[3 which is now the next available space]



POP

Need to check if the stack is empty as can't pop nothing

If this is we return 30. We reduce top by 1 top-1 (- -top) i.e. 20

then make top null. We would get back to the original diagram

PEEK

Peek l

returns the top of the stack but does not remove it

We ensure the stack is not empty and return employee at stack[top-1] . We don't decrement top

Java Code

```
public class ArrayStack {
// create an array to store the employee objects
    private Employee[] stack;
    //We can only manipulate the top of the stack
    private int top; // top points to next available
position

    public ArrayStack(int capacity) {
        stack = new Employee[capacity]; //when we
instantiate a new stack we give the size
    }
    public void push(Employee employee) {
     //WE need to find out if the stack is full as we will
get error if we try to push
        if (top == stack.length) {
            // need to resize the backing array. We double
it here
            Employee[] newArray = new Employee[2 *
stack.length];
            System.arraycopy(stack, 0, newArray, 0,
stack.length);// copy to new array
            stack = newArray;
        }

        stack[top++] = employee;  //this is how we push
    }
//takes top item off the stack
    public Employee pop() {
     //check if it is empty first
        if (isEmpty()) {
            throw new EmptyStackException();
        }

        Employee employee = stack[--top];  //we want to
pop the item under top (1 less than the top)
        stack[top] = null; //
        return employee;
    }

    public Employee peek() {
        if (isEmpty()) {
            throw new EmptyStackException();
        }

        return stack[top - 1];  //not --top as we are
looking not popping
    }

    public int size() {
        return top; //the top is the size
    }
```

```java
    public boolean isEmpty() {
        return top == 0;
    }

    public void printStack() {
        for (int i = top - 1; i >= 0; i--) {
            System.out.println(stack[i]);
        }
    }

}
```