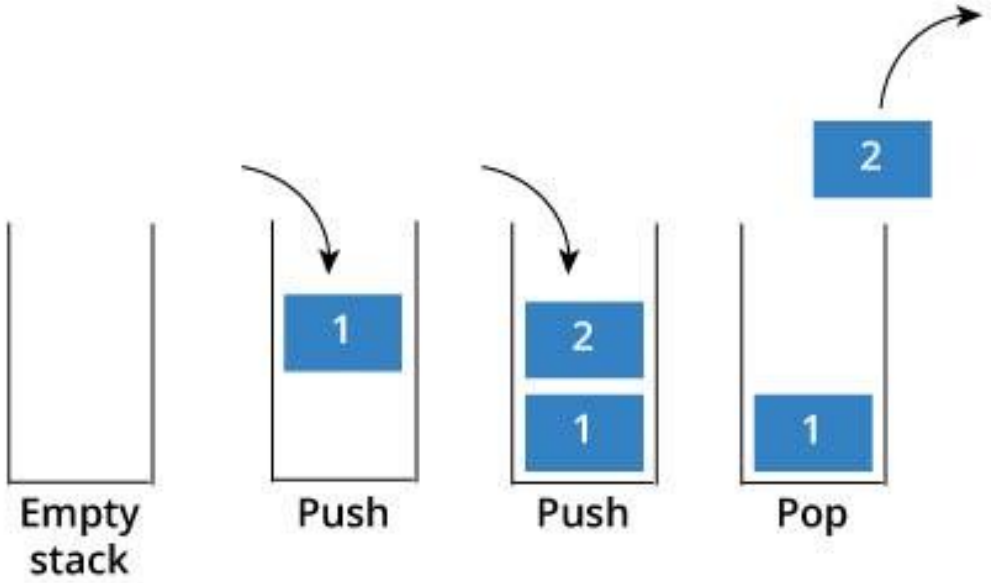
Stacks

An example of a complex object



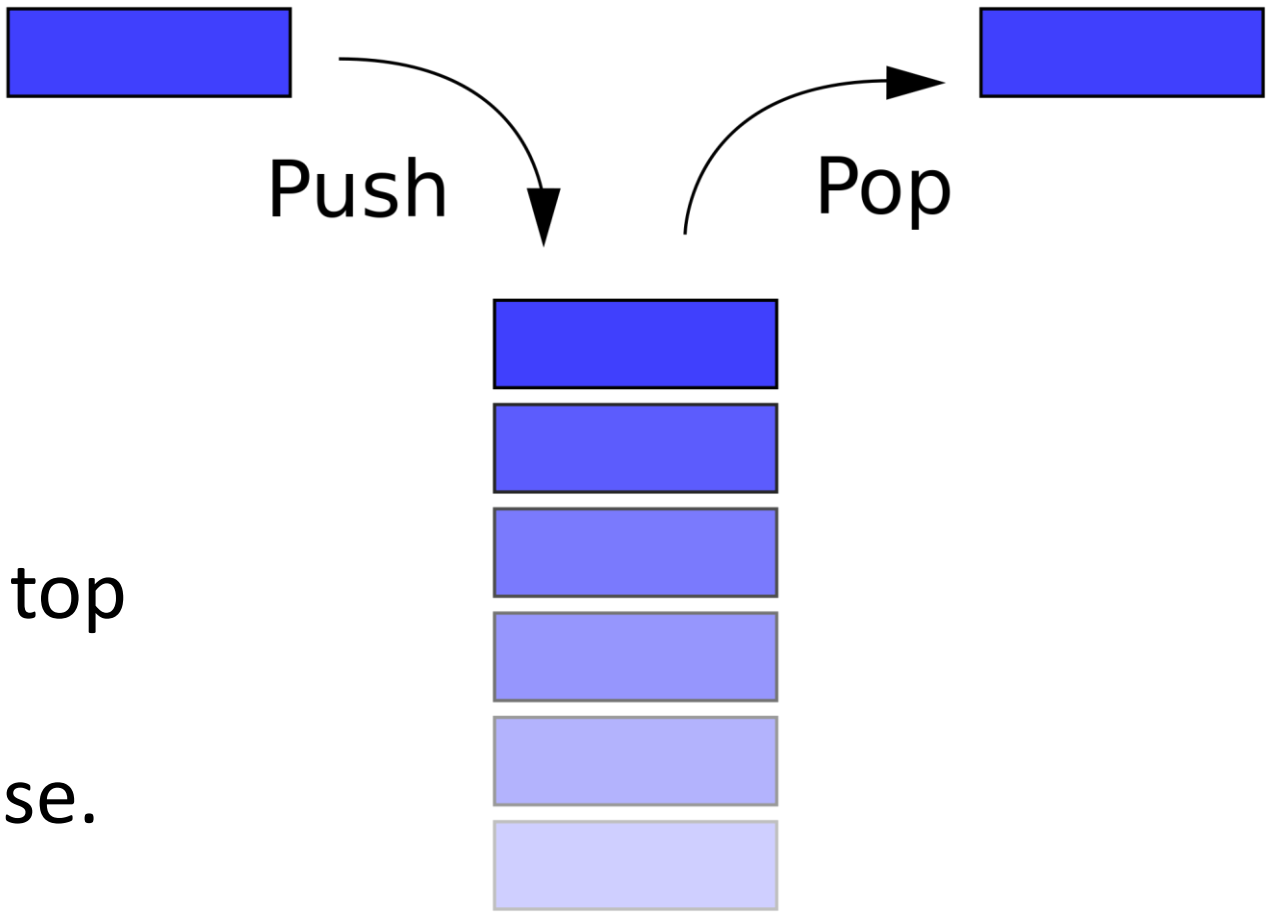
ADT

- Abstract Data Type
- An standard object that is used for storing data; pop up frequently in coding.
- They are well designed and coded
- They do ONE task, really well. Nothing else.
- Examples: Stack, Queue, Tree, Heap, Hash Table

Stack - LIFO

- An standard object type (an ADT)
- Last in, first out (LIFO)
- Everything occurs to the top
- Used for back and undo buttons; Also recursive calls (recall: Stack overflow)

Stack Functions



Push = Add to the top
Pop = Remove from the top

You can’t do anything else.
No sorting.

```
1. Stack s = new Stack();
2. System.out.println(s.isEmpty());
3. s.push("9");
4. s.push("8");
5. System.out.println(s.size());
6. s.push("7");
7. System.out.println(s.pop());
8. s.push("6");
9. System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

```
1. Stack s = new Stack();
2. System.out.println(s.isEmpty());
3. s.push("9");
4. s.push("8");
5. System.out.println(s.size());
6. s.push("7");
7. System.out.println(s.pop());
8. s.push("6");
9. System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

```
1. Stack s = new Stack();
2. System.out.println(s.isEmpty());
3. s.push("9");
4. s.push("8");
5. System.out.println(s.size());
6. s.push("7");
7. System.out.println(s.pop());
8. s.push("6");
9. System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

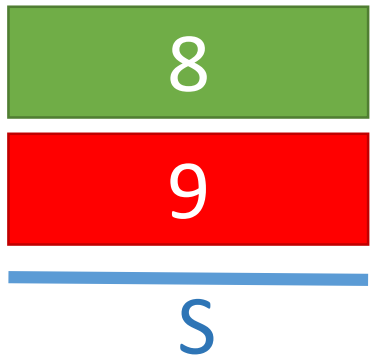
true

```
1. Stack s = new Stack();
2. System.out.println(s.isEmpty());
3. s.push("9");
4. s.push("8");
5. System.out.println(s.size());
6. s.push("7");
7. System.out.println(s.pop());
8. s.push("6");
9. System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

true

```
1. Stack s = new Stack();
2. System.out.println(s.isEmpty());
3. s.push("9");
4. s.push("8");
5. System.out.println(s.size());
6. s.push("7");
7. System.out.println(s.pop());
8. s.push("6");
9. System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

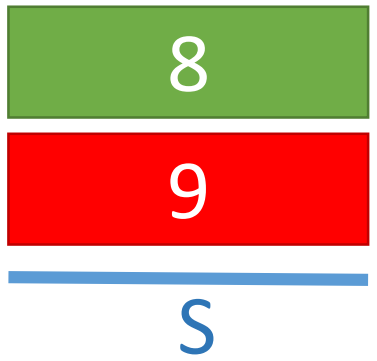
true



```
1.  Stack s = new Stack();
2.  System.out.println(s.isEmpty());
3.  s.push("9");
4.  s.push("8");
5.  System.out.println(s.size());
6.  s.push("7");
7.  System.out.println(s.pop());
8.  s.push("6");
9.  System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

true

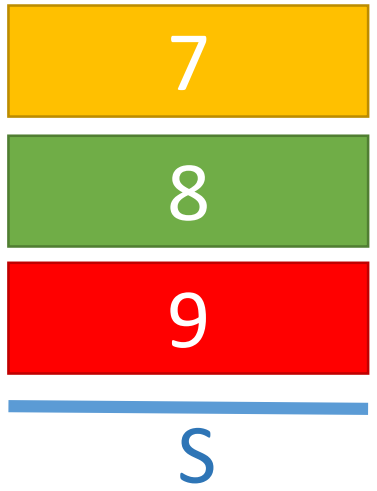
2



```
1. Stack s = new Stack();
2. System.out.println(s.isEmpty());
3. s.push("9");
4. s.push("8");
5. System.out.println(s.size());
6. s.push("7");
7. System.out.println(s.pop());
8. s.push("6");
9. System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

true

2

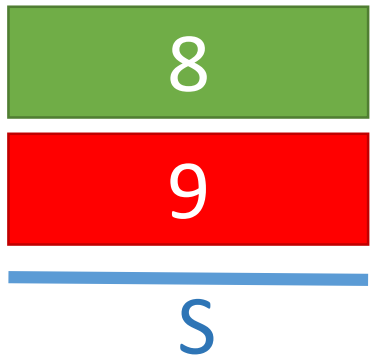


```
1.  Stack s = new Stack();
2.  System.out.println(s.isEmpty());
3.  s.push("9");
4.  s.push("8");
5.  System.out.println(s.size());
6.  s.push("7");
7.  System.out.println(s.pop());
8.  s.push("6");
9.  System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

true

2

7

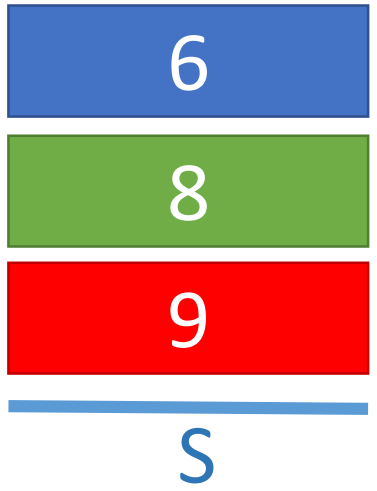


```
1. Stack s = new Stack();
2. System.out.println(s.isEmpty());
3. s.push("9");
4. s.push("8");
5. System.out.println(s.size());
6. s.push("7");
7. System.out.println(s.pop());
8. s.push("6");
9. System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

true

2

7



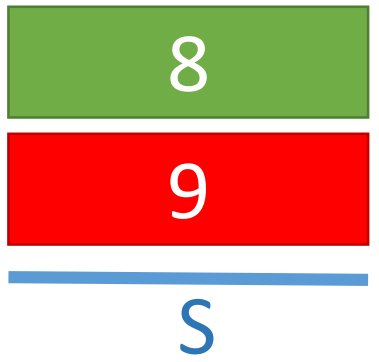
```
1.  Stack s = new Stack();
2.  System.out.println(s.isEmpty());
3.  s.push("9");
4.  s.push("8");
5.  System.out.println(s.size());
6.  s.push("7");
7.  System.out.println(s.pop());
8.  s.push("6");
9.  System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

true

2

7

6



```
1.  Stack s = new Stack();
2.  System.out.println(s.isEmpty());
3.  s.push("9");
4.  s.push("8");
5.  System.out.println(s.size());
6.  s.push("7");
7.  System.out.println(s.pop());
8.  s.push("6");
9.  System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

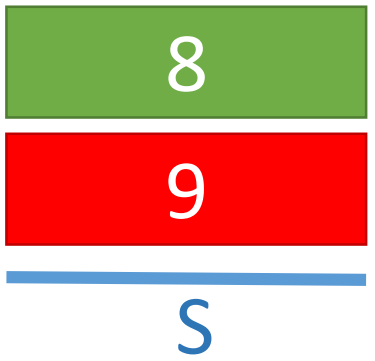
true

2

7

6

2



```
1. Stack s = new Stack();
2. System.out.println(s.isEmpty());
3. s.push("9");
4. s.push("8");
5. System.out.println(s.size());
6. s.push("7");
7. System.out.println(s.pop());
8. s.push("6");
9. System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

true

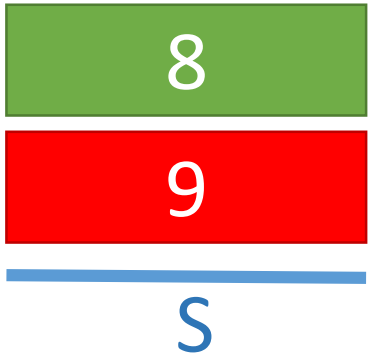
2

7

6

2

false



```
1.  Stack s = new Stack();
2.  System.out.println(s.isEmpty());
3.  s.push("9");
4.  s.push("8");
5.  System.out.println(s.size());
6.  s.push("7");
7.  System.out.println(s.pop());
8.  s.push("6");
9.  System.out.println(s.pop());
10. System.out.println(s.size());
11. System.out.println(s.isFull());
12. System.out.println(s.isEmpty());
```

true

2

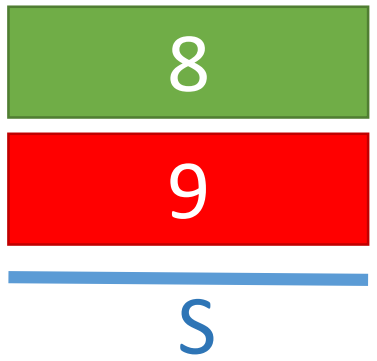
7

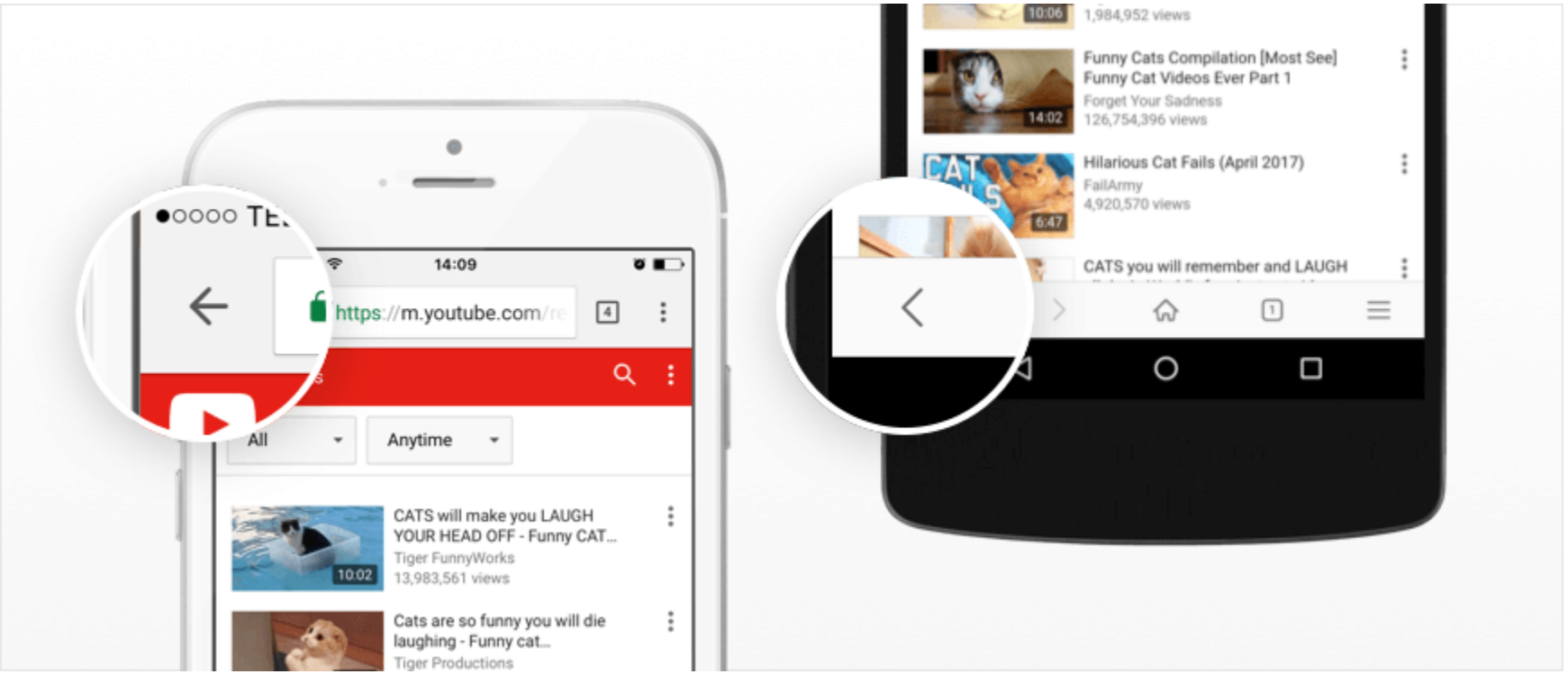
6

2

false

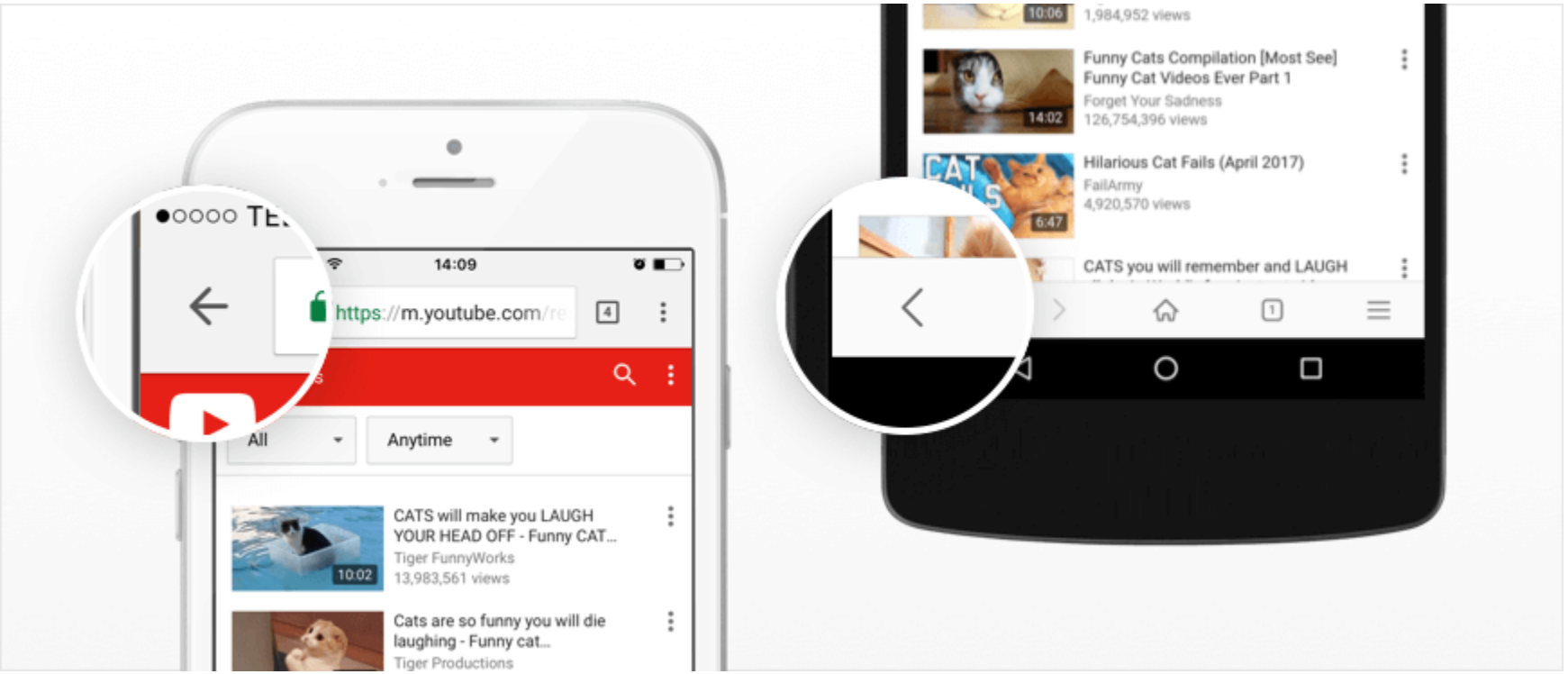
false





YouTube

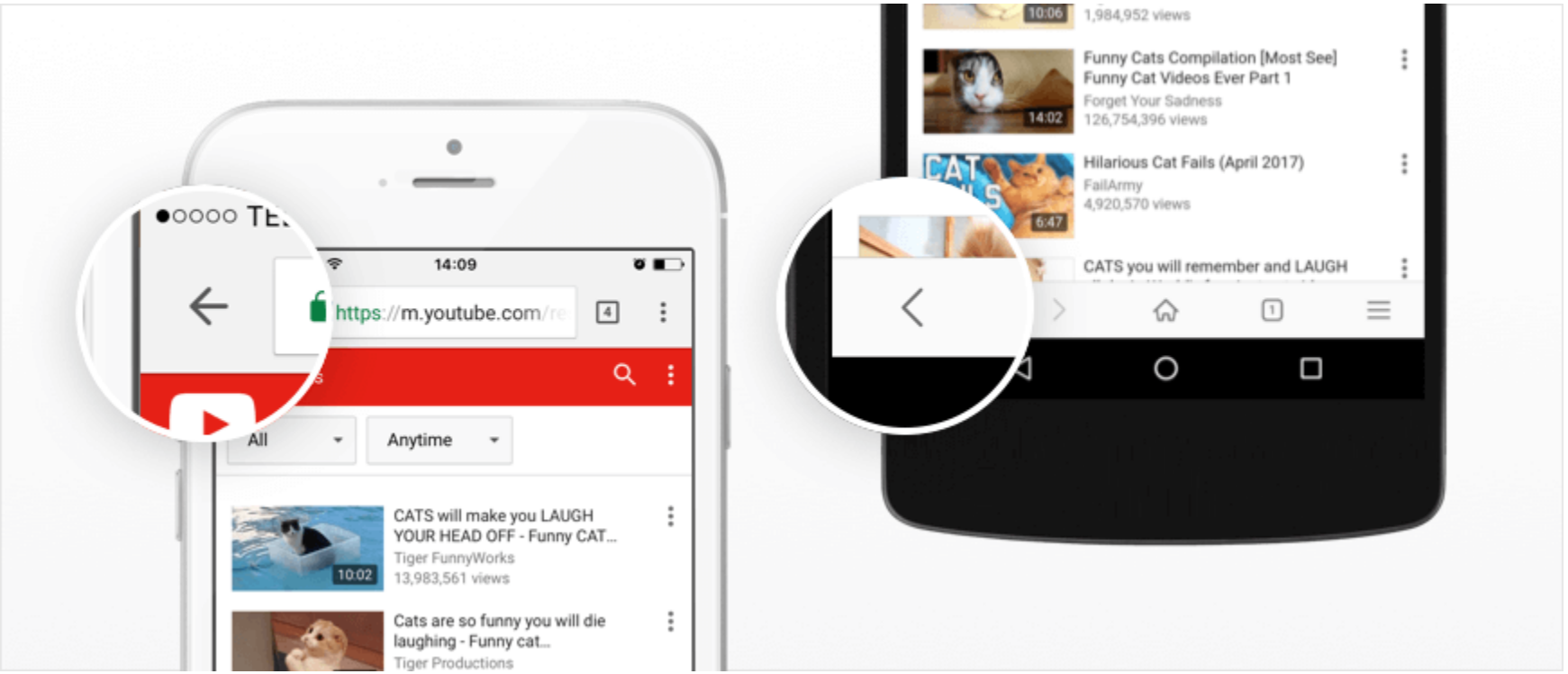
back



GorskiCompSci

YouTube

back

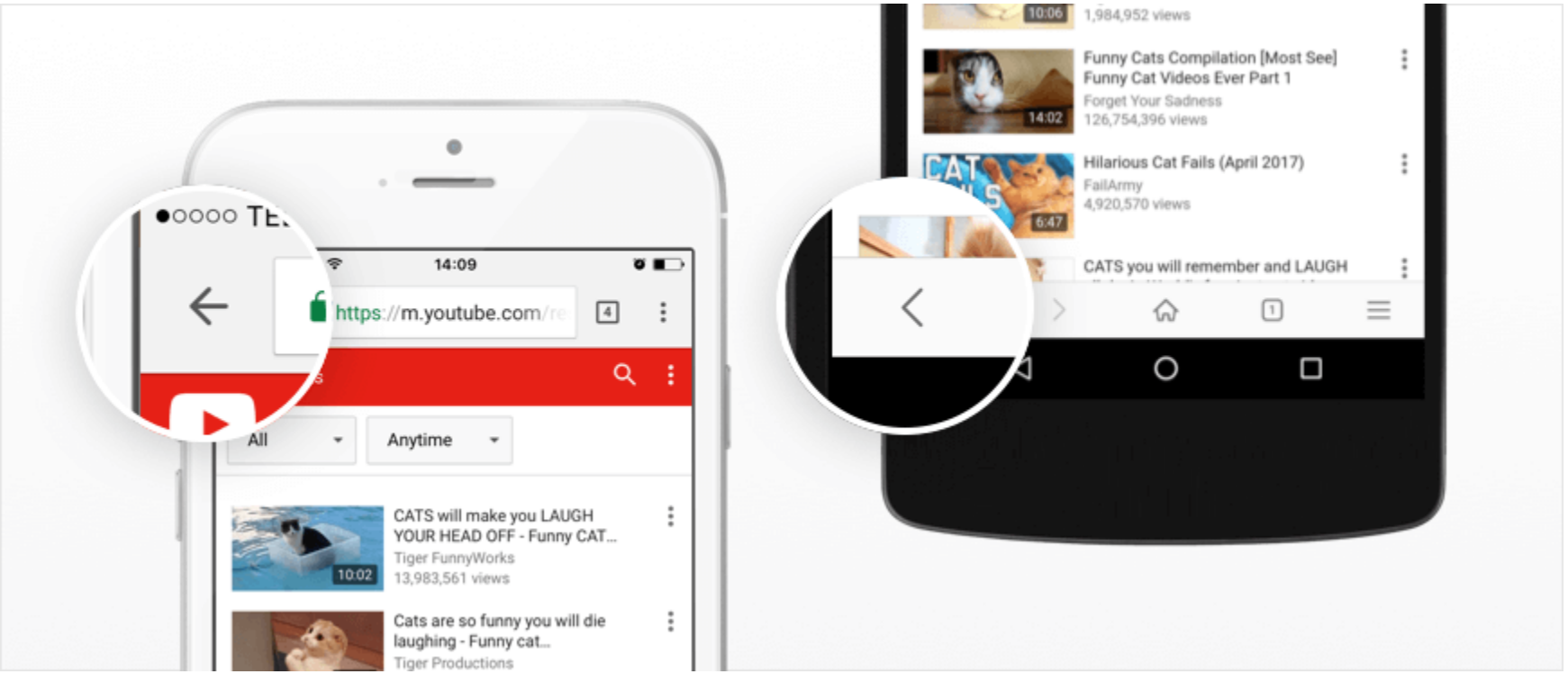


Google

GorskiCompSci

YouTube

back



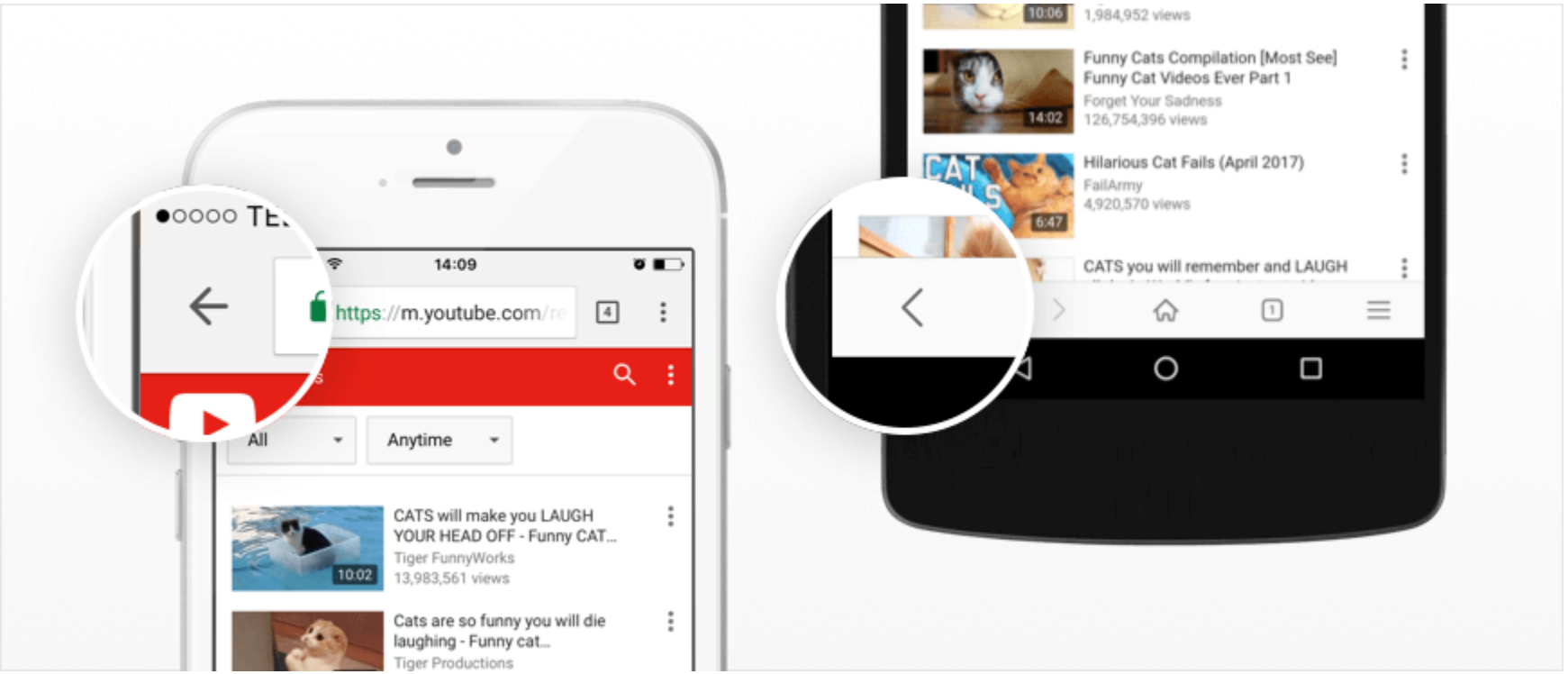
Twitch

Google

GorskiCompSci

YouTube

back



BACK

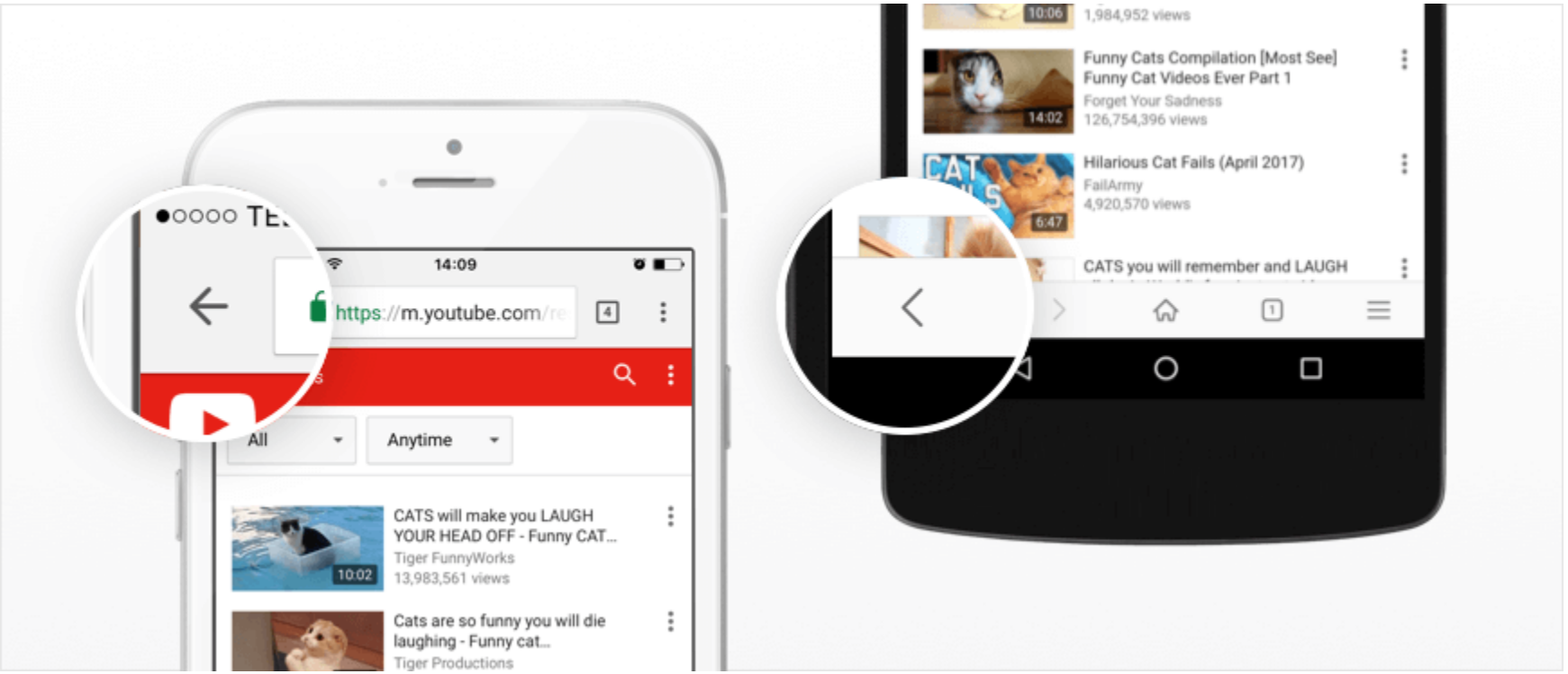
Twitch

Google

GorskiCompSci

YouTube

back

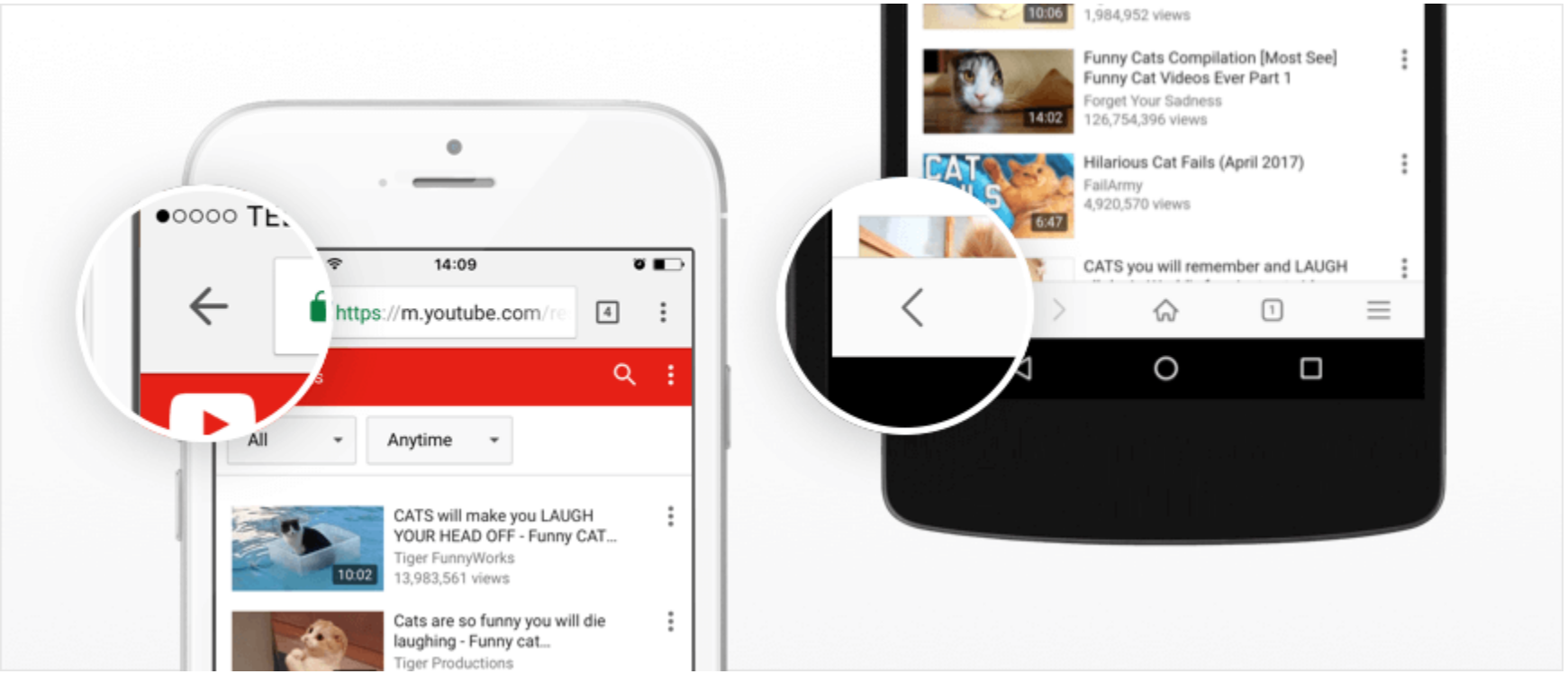


Google

GorskiCompSci

YouTube

back



GorskiCompSci

Google

GorskiCompSci

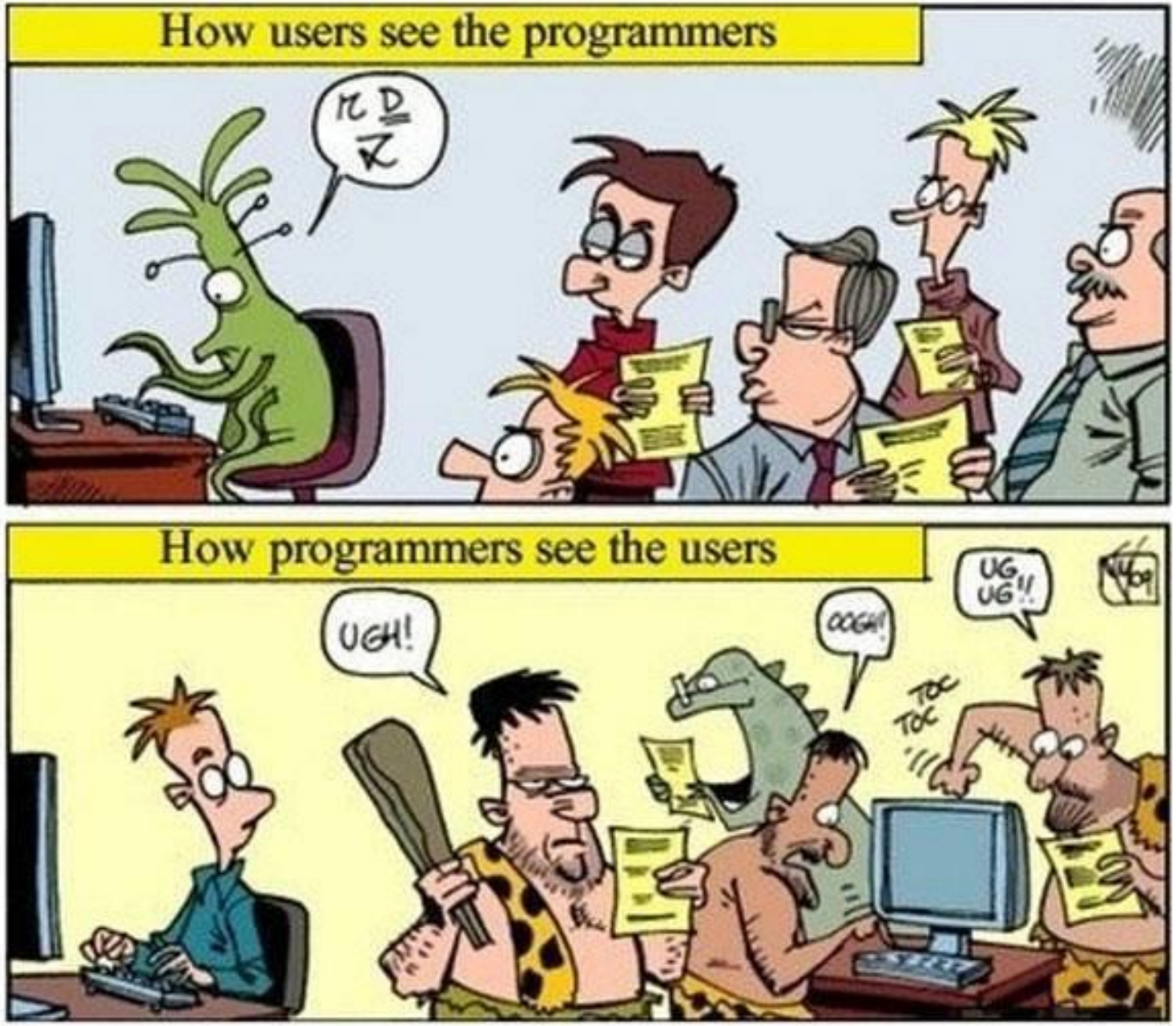
YouTube

back

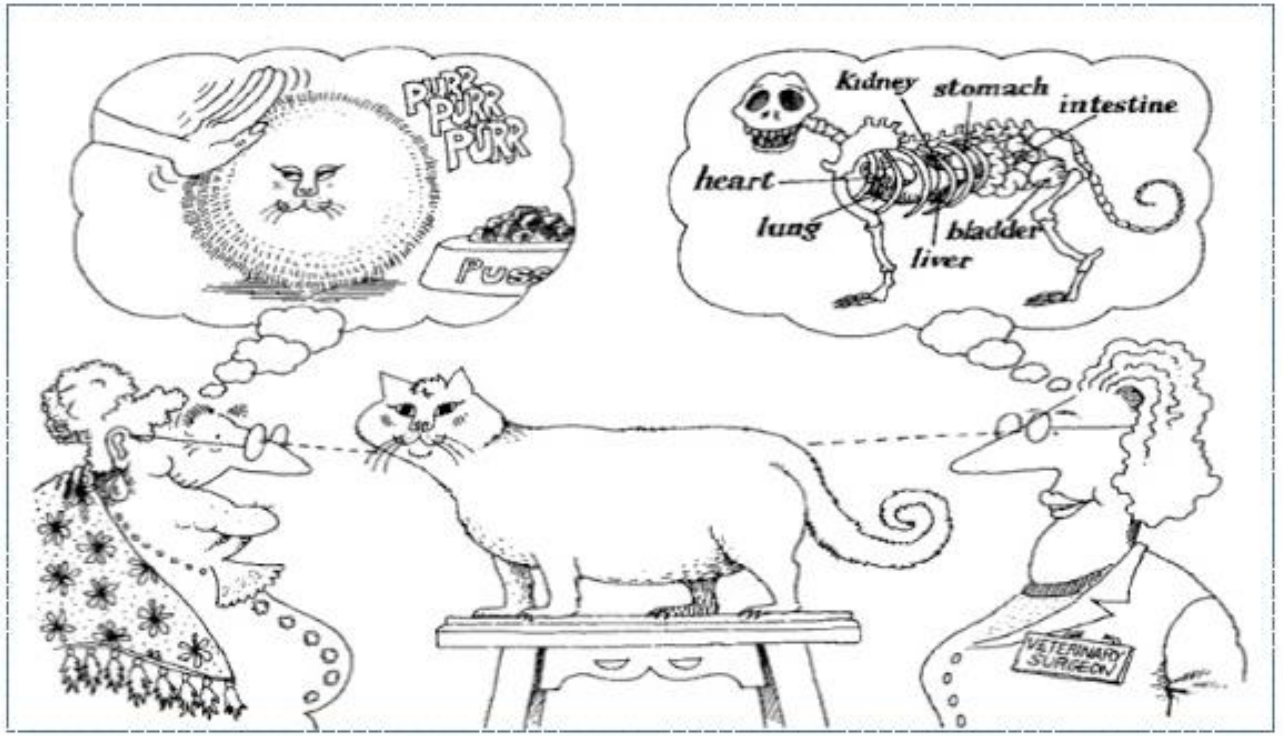
Object = data + methods

- We are grouping together the stack data AND its methods.
- Our stack will be self-contained and programmers who need a back or undo button can use it easier.

Remember that with objects, our user has changed!



With Objects, your user changes to other programmers.

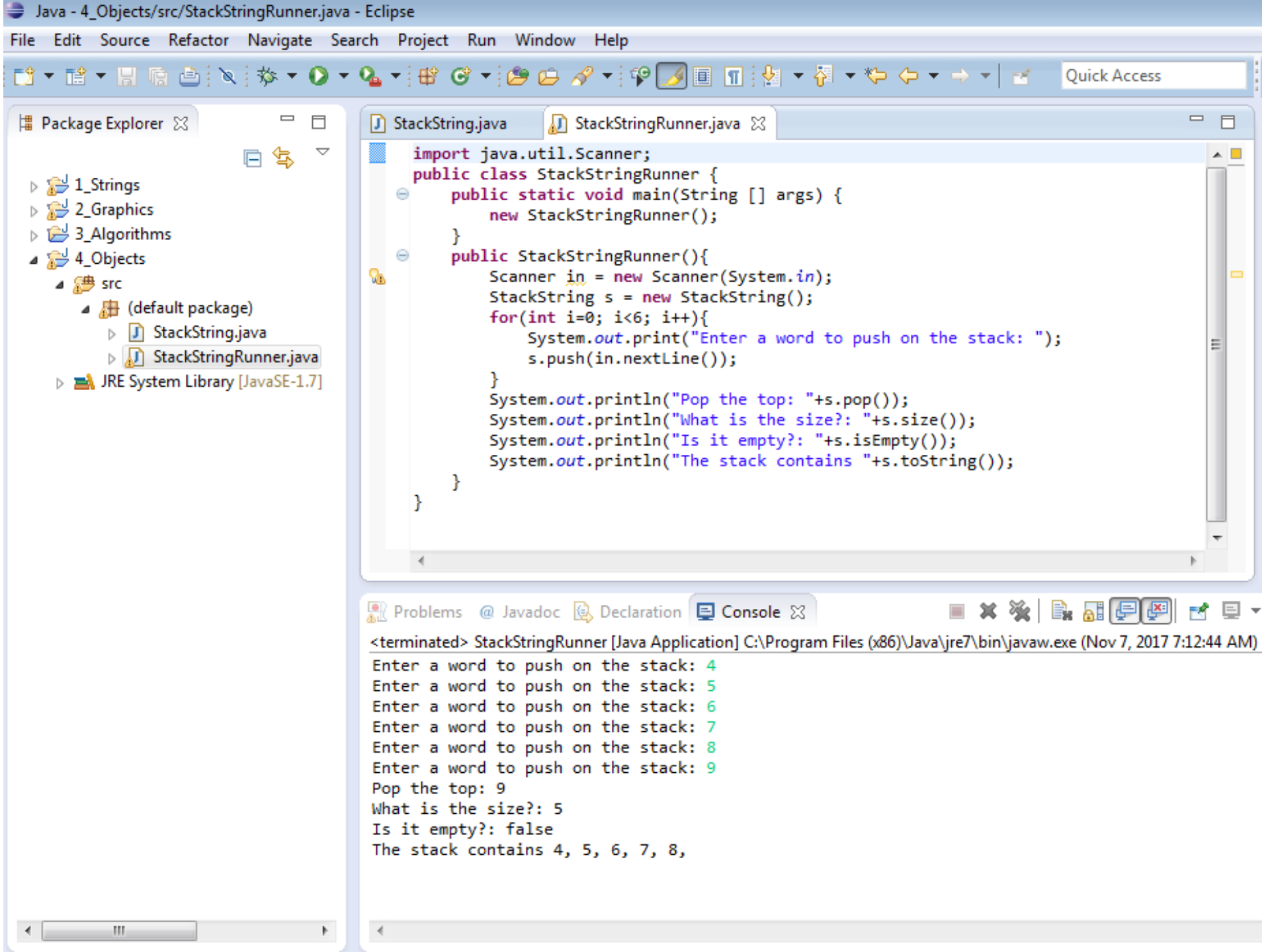


- Stacks are also a good example of information hiding.
- Who cares how push and pop are coded?
- You just need to be aware of how they are called and you can make a back button.

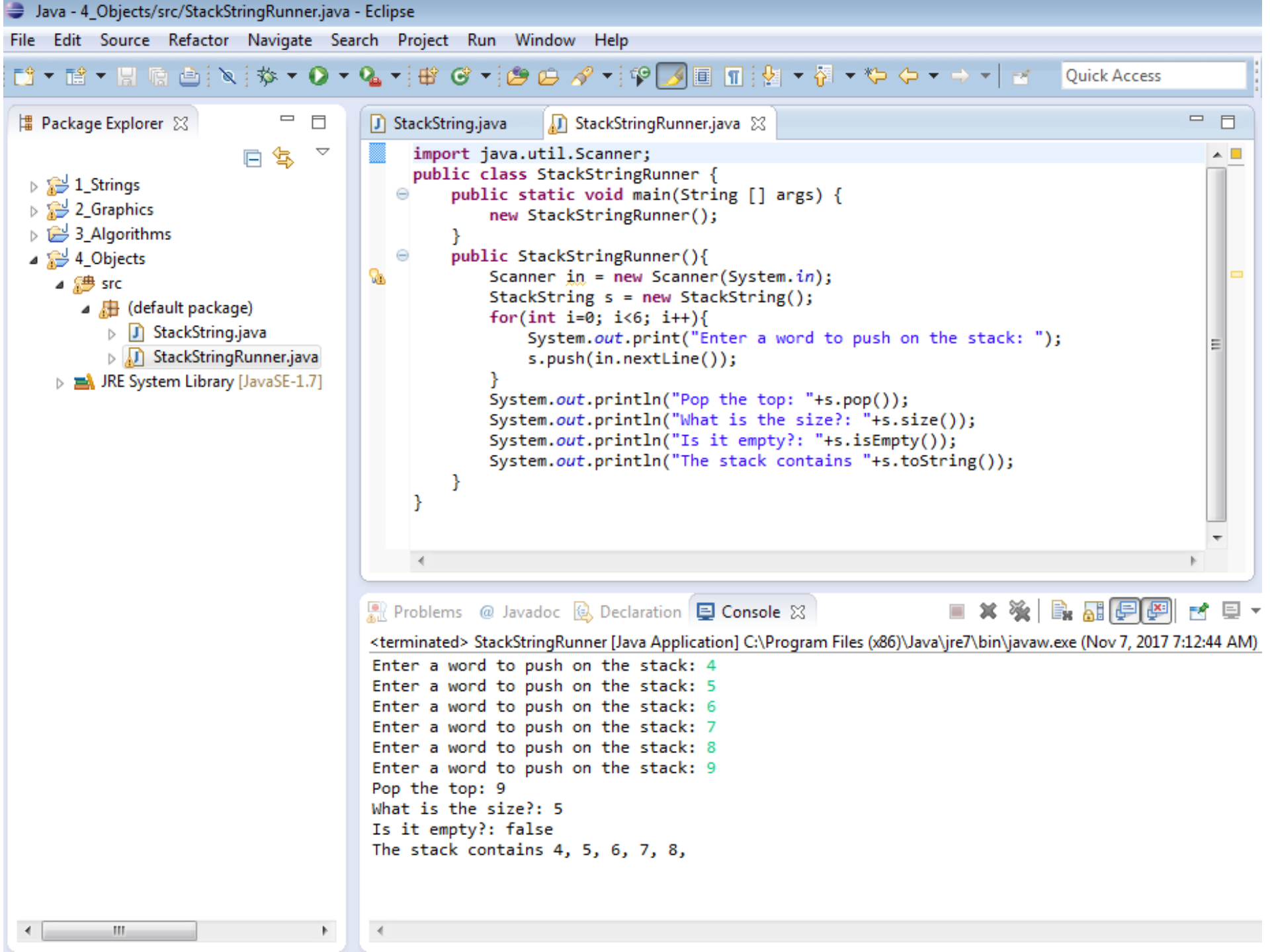
Object = data + methods

```
public class StackString {

private int count;
private String data[] = new String [50];

public StackString () {
  count = 0;
}

public void push (String addMe) {
  if(!isFull()){
    data [count] = addMe;
    count++;
  }
}

public int size () {
  return count;
}

public boolean isFull () {
  return (count == 50);
}
```

```
public String pop () {
  if(count>0){
     count--;
     return data [count];
  }
  else
    return "Stack empty";
}

public String peek () {
  return data [count--];
}

public boolean isEmpty ()  {
  return count == 0;
}

public void clear () {
  count = 0;
}

}
```


Java - 4_Objects/src/StackStringRunner.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
1_Strings
2_Graphics
3_Algorithms
4_Objects
src
(default package)
StackString.java
StackStringRunner.java
JRE System Library [JavaSE-1.7]
StackString.java
StackStringRunner.java
import java.util.Scanner;
public class StackStringRunner {
public static void main(String [] args) {
new StackStringRunner();
}
public StackStringRunner(){
Scanner in = new Scanner(System.in);
StackString s = new StackString();
for(int i=0; i<6; i++){
System.out.print("Enter a word to push on the stack: ");
s.push(in.nextLine());
}
System.out.println("Pop the top: "+s.pop());
System.out.println("What is the size?: "+s.size());
System.out.println("Is it empty?: "+s.isEmpty());
System.out.println("The stack contains "+s.toString());
}
}
Problems Javadoc Declaration Console
<terminated> StackStringRunner [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (Nov 7, 2017 7:12:44 AM)
Enter a word to push on the stack: 4
Enter a word to push on the stack: 5
Enter a word to push on the stack: 6
Enter a word to push on the stack: 7
Enter a word to push on the stack: 8
Enter a word to push on the stack: 9
Pop the top: 9
What is the size?: 5
Is it empty?: false
The stack contains 4, 5, 6, 7, 8,

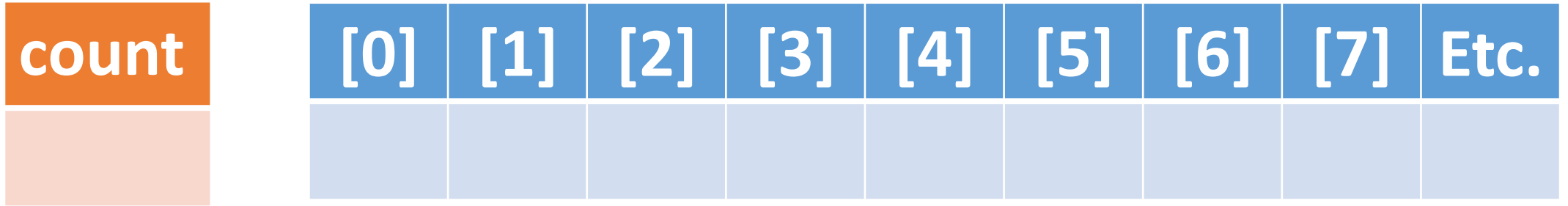
```
import java.util.Scanner;
public class StackStringRunner {
    public static void main(String [] args) {
        new StackStringRunner();
    }

    public StackStringRunner(){
        Scanner in = new Scanner(System.in);
        StackString s = new StackString();

        for(int i=0; i<6; i++){
            System.out.print("Enter a word to push on the stack: ");
            s.push(in.nextLine());
        }

        System.out.println("Pop the top: "+s.pop());
        System.out.println("What is the size?: "+s.size());
        System.out.println("Is it empty?: "+s.isEmpty());
        System.out.println("The stack contains "+s.toString());
    }
}
```


Java - 4_Objects/src/StackStringRunner.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
1_Strings
2_Graphics
3_Algorithms
4_Objects
src
(default package)
StackString.java
StackStringRunner.java
JRE System Library [JavaSE-1.7]
StackString.java
StackStringRunner.java
import java.util.Scanner;
public class StackStringRunner {
public static void main(String [] args) {
new StackStringRunner();
}
public StackStringRunner(){
Scanner in = new Scanner(System.in);
StackString s = new StackString();
for(int i=0; i<6; i++){
System.out.print("Enter a word to push on the stack: ");
s.push(in.nextLine());
}
System.out.println("Pop the top: "+s.pop());
System.out.println("What is the size?: "+s.size());
System.out.println("Is it empty?: "+s.isEmpty());
System.out.println("The stack contains "+s.toString());
}
}
Problems Javadoc Declaration Console
<terminated> StackStringRunner [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (Nov 7, 2017 7:12:44 AM)
Enter a word to push on the stack: 4
Enter a word to push on the stack: 5
Enter a word to push on the stack: 6
Enter a word to push on the stack: 7
Enter a word to push on the stack: 8
Enter a word to push on the stack: 9
Pop the top: 9
What is the size?: 5
Is it empty?: false
The stack contains 4, 5, 6, 7, 8,

Stack

- A common ADT, An object
- Has a LIFO structure (Last in, first out)
- Functions: size, isFull, isEmpty, push (add), pop (remove), peek (look at the top)
- All operations are O(1), but you are limited to stack functions. The speed comes from the fact that you restricted the functionality a lot.
- For example: You can’t sort a stack: why? Because the order of the stack is the order you want. If you don’t want that order, then you don’t want a stack. Go use an array instead.

```
public class StackString {
     private int count;
     private String data[] = new String [50];
}
```



```
public class StackString {
   private int count;
   private String data[] = new String [50];

public StackString () {
   count = 0;
}
```

count
0

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	Etc.

```
public int size () {
   return count;
}

public boolean isFull () {
   return (count == 50);
}
```

count
0

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	Etc.

```
public String peek () {
   return data [count--];
}

public boolean isEmpty ()  {
   return count == 0;
}

public void clear () {
   count = 0;
}
```

count
0

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	Etc.

```
public void push (String addMe) {
   if(!isFull()){
      data [count] = addMe;
      count++;
   }
}
```

count
0

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	Etc.

```
public String pop () {
   if(count>0){
      count--;
      return data [count];
   }
   else
      return "Stack empty";
}
```

count
0

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	Etc.

```
public String toString () {
//this is not an official Stack method
//it is useful to see what is in your Stack
String ans = "";
for(int i=0; i<count; i++)
  ans+= data[i]+", ";
return ans;
}
```

count
0

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	Etc.