```java
package nyc.c4q;
import java.util.ArrayList;
import java.util.List;


public class ConstructorChallenge {
    int result = 0;
    public ConstructorChallenge(int value)  {  result = value;  }

    public ConstructorChallenge(String value) { result = Integer.valueOf(value); }

    public ConstructorChallenge(List<Integer> values) {
        for (int index = 0; index < values.size(); index++){
            result += values.get(index);
        }
    }

    public ConstructorChallenge increment(){
        return new ConstructorChallenge(result+1);
    }

    public static void question1(){
        ConstructorChallenge result = new ConstructorChallenge(9);
        System.out.println(result.result);
    }

    public static void question2(){
        ConstructorChallenge result = new ConstructorChallenge("1943");
        System.out.println(result.result);
    }

    public static void question3(){
        ArrayList<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list = new ArrayList<>();
        list.add(4);
        list.add(5);
        list.add(6);
        ConstructorChallenge result = new ConstructorChallenge(list);
        System.out.println(result.result);
    }

    public static void question4(){
        ConstructorChallenge temp = new ConstructorChallenge("1000");
        ConstructorChallenge result = temp.increment();
        System.out.println(result.result);
    }
}
```

java.util

Class Stack<E>

- java.lang.Object
  - java.util.AbstractCollection<E>
    - java.util.AbstractList<E>
      - java.util.Vector<E>
        - java.util.Stack<E>

- All Implemented Interfaces:
  Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

---

```
public class Stack<E>
extends Vector<E>
```

The `Stack` class represents a last-in-first-out (LIFO) stack of objects. It extends class `Vector` with five operations that allow a vector to be treated as a stack. The usual `push` and `pop` operations are provided, as well as a method to `peek` at the top item on the stack, a method to test for whether the stack is `empty`, and a method to `search` the stack for an item and discover how far it is from the top.

When a stack is first created, it contains no items.

Constructor	
	Stack() Creates an empty Stack.
Modifier and Type	**Method and Description**
boolean	**empty**() Tests if this stack is empty.
E	**peek**() Looks at the object at the top of this stack without removing it from the stack.
E	**pop**() Removes the object at the top of this stack and returns that object as the value of this function.
E	**push**(E item) Pushes an item onto the top of this stack.
int	**search**(Object o) Returns the 1-based position where an object is on this stack.

```java
package nyc.c4q;
import java.util.Stack;


public class Stacks {

    public static int searchStack(String value, Stack<String> searchSpace){
        if (searchSpace.size() == 0){
            return 0;
        }
        String firstValue = searchSpace.pop();
        if (firstValue.equals(value)) {
            return 1 + searchStack(value, searchSpace);
        } else {
            return searchStack(value, searchSpace);
        }
    }


    public static int countNonValues(String value, Stack<String> searchSpace) {
        int count = searchSpace.size();
        while (searchSpace.size() > 0){
            if (searchSpace.peek().equals(value)){
                count--;
            }
            searchSpace.pop();
        }
        return count;
    }


    public static Stack<String> buildStack(){
        Stack<String> input = new Stack<>();
        for (int index = 0; index < 10; index++) {
            input.add("a");
        }
        for (int index = 0; index < 5; index++) {
            input.add("b");
        }
        for (int index = 0; index < 0; index++) {
            input.add("c");
        }
        return input;
    }
}
```

Midterm
12/10/2016
Student Name:
Student Id:

Question 1: What is the output of ConstructorChallenge.question1()?

Question 2: What is the output of ConstructorChallenge.question2()?

Question 3: What is the output of ConstructorChallenge.question3()?

Question 4: What is the output of ConstructorChallenge.question4()?

Question 5: How do you determine which constructor will be run when instantiating ConstructorChallenge?

Question 6:
Write a function named **isEven**. **IsEven** has one parameter, a stack of integers. **IsEven** returns a boolean value, true if the first item divided 2 equals 0, and false otherwise. **IsEven** should not modify the passed in parameter.

Question 7:
Write a function named **isEven2**. **IsEven2** has one parameter, a stack of integers. **IsEven2** returns a boolean value, true if the first item modulus 2 equals 0, and false otherwise. **IsEven2** should remove the first item from the passed in parameter.

Question 8:
What is the return value of the following code:

```
Stacks.searchStack("a", buildStack());
```

Question 9:
What is the return value of the following code:

```
Stacks.searchStack("c", buildStack());
```

Question 10:
What is the return value of the following code:

```
Stacks.countNonValues("b", buildStack());
```

The printListMinusHead function is supposed to print all values in the list, then remove item at index 0 and return the modified list.

```
public ArrayList<String> printListMinusHead(ArrayList<String> input){
     ArrayList<String> result = new ArrayList<>();
     int index = 0;
     while (!input.isEmpty()){
          System.out.println(input.get(index));
          index++;
     }
     return result;
}
```

Question 11:
Write a function that will test whether or not printListMinusHead returns the correct value. Since this is a test function, it must use asserts to check whether or not the value is correct.

Practical Test

We will be building an app using a test server from an actual company. We will be using an interface from Vine:

> https://vine.co/api/timelines/users/918753190470619136

Your assignment is to build an app with a single screen.

The screen should display the list of records received from the above url.

For each record display the following attributes:

> **liked**
>
> **The username in the user object**
>
> **The background of the view should be the value of profileBackground**

The app should work in both portrait and landscape.

This portion of the test will be graded according to git commits. Each commit must have the described message in bold. The commit must demonstrate the described feature.

First Step – Your android app runs, displays nothing on the screen, and has at least one activity
Second Step – Your android app successfully downloads data from the url.
Third Step - Your android app displays the data for at least one post on the screen
Fourth Step – Your android app displays the data for several posts on the screen
Fifth Step – Your android app displays the data for several posts on the screen, and the posts can be scrolled to see all of the content

You must also submit a text file in your repository named README.md.
In this file include the following:

Student Name
Student Id
Emulator version or phone api tested on
An explanation of what you got stuck on if anything