Ask me anything

0 questions
0 upvotes
What is the “unit” tested by a Unit Test?

- A method: 17
- A class: 6
- An object: 2
- A package: 1
- A program: 4
- A scenario: 8
What are the advantages of JUnit 5 over JUnit 3?

- The “rules” got easier.
- Hunit 5 is more flexible
- use of @
- Having annotations is easier to understand
- simpler
- Easier implementation
- no need to have “test” in the name
- tests don't need to have “test” in their names
- we do not need to extend a test case, and do not need to import the framework.TestCase
What are the advantages of JUnit 5 over JUnit 3?

JUnit no longer relies on inheritance to plug concrete tests into the framework and it uses annotations for flag test methods.

Running before the first test in the class is invoked.
What would be boundary conditions for testing the paren matcher?

- an empty String
- null
- When there are more right brackets than left ones.
- a String only with parent characters
- a too large String?
- starting with a right bracket as a first char
- if last char in the string is a left bracket
How would you design tests to pass through points (2) and (4)?

```java
public boolean parenMatch() {
    for (int i=0; i<line.length(); i++) {
        char c = line.charAt(i);
        if (isLeftParen(c)) {
            stack.push(matchingRightParen(c)); // (1)
        } else {
            if (isRightParen(c)) {
                if (stack.isEmpty()) {
                    return false; // (2)
                }
                if (stack.top().equals(c)) {
                    stack.pop(); // (3)
                } else {
                    return false; // (4)
                }
            } else not a paren char (5)
        }
    return stack.isEmpty(); // (6)
}
```

To reach point (2), we must have just read a right parenthesis, but the stack must be empty.

To reach point (4), we must have read a right parenthesis, but it does not match the top of the stack.

Only a single right parentheses as input

String without any parenthesis

And for point 4 something like ()

A empty string(2)

"\)

4
How would you design tests to pass through points (2) and (4)?

```java
public boolean parenMatch() {
    for (int i=0; i<line.length(); i++) {
        char c = line.charAt(i);
        if (isLeftParen(c)) {
            stack.push(matchingRightParen(c)); // (1)
        } else {
            if (isRightParen(c)) {
                if (stack.isEmpty()) {
                    return false; // (2)
                }
                if (stack.top().equals(c)) {
                    stack.pop(); // (3)
                } else {
                    return false; // (4)
                }
            } // else not a paren char (5)
        }
    } // else not a paren char (5)
    return stack.isEmpty(); // (6)
}
```
FizzBuzz TDD Coding Kata

Use TDD to write a class that outputs the numbers 1 to 100, but prints “Fizz” for multiple of 3, “Buzz” for multiples of 5, and “FizzBuzz” for multiples of 15.
What tests would you write for FizzBuzz? What are the boundary tests?

- if 15 is a fizzbuzz
- assert 3 prints fizz
- 2+1 Fizz
- 0?
- 5 is buzz
- A test Case for: 1,3(Fizz),5(Buzz),12(Fizz),15(fizzbuzz),30(fizzbuzz) and 101
- 7 is 7
- 45 (3*15)
- 0 should raise exception
What tests would you write for FizzBuzz? What are the boundary tests?

- 0 could be fizzbuzz or throw an exception
- if(i%15 == 0) return Fizzbuzz;
- if(i%3==0 & & i%5==0)(return "FizzBuzz")
- at 0 we should become nothing, because it's not in the boundary, or an exception
- i % 2 == 0
- if(i%3 == 0 & & i % 15 !=0) return "fuzz";
- i mod 3 == 0 returns fizz
- you could also save the lines to be printed before printing them, and the fizzbuzz method returns it in the for loop listxy.add()
What tests would you write for FizzBuzz? What are the boundary tests?

- if (i/10 == 3 || i%10 == 3) return (Fizz)
- if (i/10 == 5 || i%10 == 5) return (Buzz)
- if (Integer.parseInt(i).contains("3")...
FizzBuzz v2

Modify the program to print “Fizz” also for numbers containing “3” and “Buzz” for numbers containing “5”.
What additional tests would you write for FizzBuzz v2?

13 should return buzz

13 (Fizz) 51 (BUZZ) they will be false

if ((3/10 == 3) || (3%10 == 3)) {return (fizz)}

*fizz

*fizz : i wrote buzz instead of fizz.
13 should return fizz

Is 35 fizzbuzz or buzz?

53 -> buzzfizz?

we need to take a look at the chars.

same with 5
What additional tests would you write for FizzBuzz v2?

- `if i.toString.contains("3")`
- `you can divide it by 3`
- `51 should give Buzz because it containing "5"`
Last chance for questions