Agenda

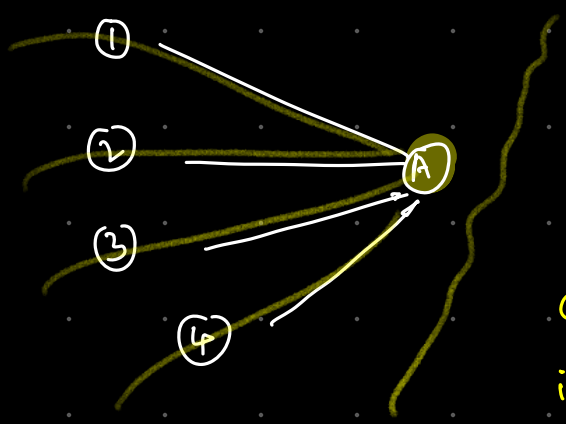
1. Why testing
2. TDD
3. Flaky tests
4. Types of testing.

```
int findNOOFWheels(int noOFBikes)
    return multiplierUtility(noOFBikes)

int multiplierUtility(int x)
    return x*3;

int totalMarriedPeople(int pairs)
    return multiplierUtility(pairs);
```

Every bike will have 3 wheels--



- cost of new change increase.
- nobody remembers the code
- manually find out the issues

Can I somehow automatically get to know if something breaks, when I push a new code. (before prod).

Testing

→ A dev shouldn't only focus on writing the code.

→ dev's should also focus on writing automated test cases.

Doesn't require human intervention

Code that'll test the code we've written.

Development approach

1. Before you push any new changes, Run all test cases If all the tests are passing and assuming that you've enough test cases to cover all scenarios, then you can push new change.

Sample test case

```
int x = multiplierUtility(10);
if (x != 20)
    throw exception()
```

multiple inputs..

20, ~~45.2~~ → throw , 0, -2, INT_MAX.

Why doesn't all the devs write tests?

1. lack of visibility.
2. No immediate return of investment.
3. We want our code to get deployed fast
4. Under appreciation

Most of the companies are having tools to make the test cases compulsory.

Two ways of writing tests...

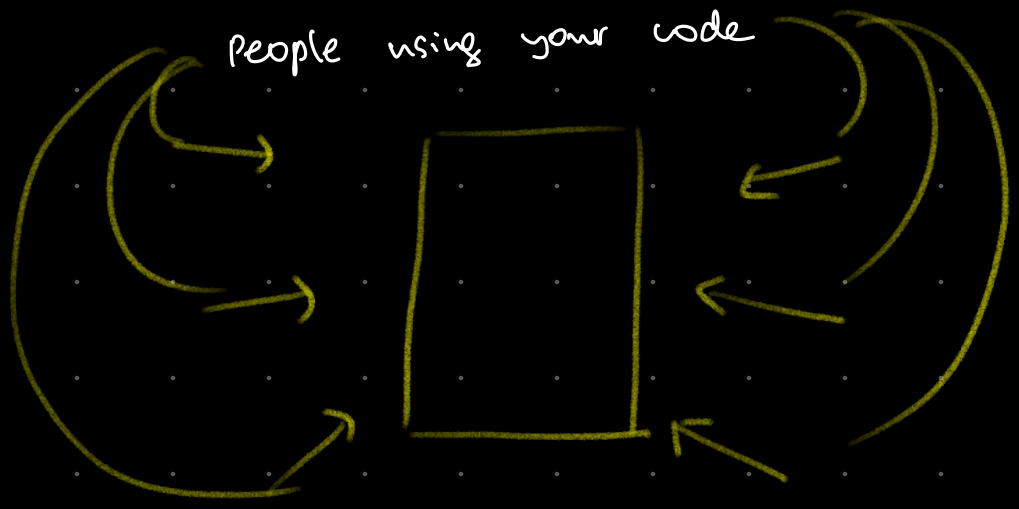
1. feature → test case → submit
2. test cases → feature → submit [TDD].

Find max subarray sum => Kadane's algo.

```
int findMaxSubArraySum(int[] input)
|
|   // code 1
|   // code 2        } submit
|
|   // code 3
|   // code 4        } submit
|
|
|
```

When do you say a code is good code..

You can't say it for your own code, somebody else needs to certify it.



Tax Calculator.

Class TaxCalculatorTest

test1()

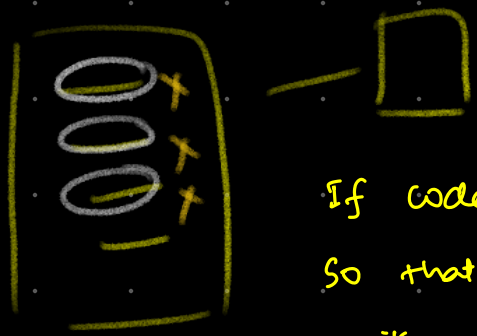
builder = TaxCalculator.builder()
- setBase(___)
- setHRA(___)
- setAllowance(___)
- setDonations(___)
:

int x = builder.calculateTax()

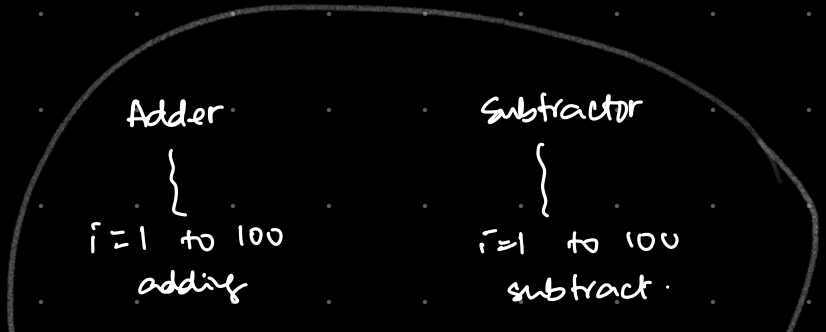
if (x != 500)

Flaky test

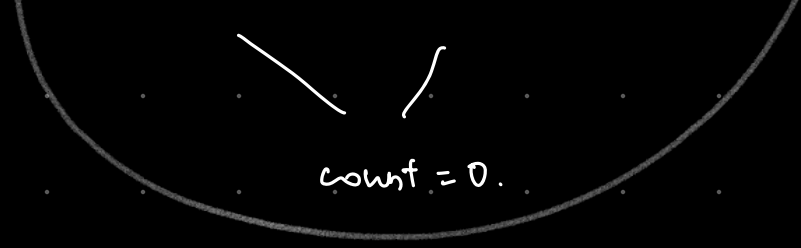
* unreliable, passing for some i/p and failing sometimes.



If code is breaking, than don't fix test cases so that they pass. Fix the code first & than write new tests.



Count = 50.
70.



AdderSubtractorTest

test()

if (x! = 50) 70

start two threads

t1. adder()

t2. subtractor()

x! = 0

throw error

modifying

Writing test cases just to pass them is the worst thing.

It creates more problems than earlier.

Types of testing

b()

a()

c()

test for b is failing.

Which fun is having an issue?

b(), a(), c()

If test case of 'b' is failing. It means the bug is in 'b' only and not in 'a' or 'c'.

=> Mocking

unit tests

-> test cases just for that one method.

-> all other dependencies must be mocked (can be assumed that they work as expected).

Test coverage

100 lines of code.

98 lines of code.

$$\text{coverage} = \frac{98}{100} \times 100 = 98\% \text{ coverage.}$$

(80 - 90)% mandatory for most of the companies.

```
long maxSubsum(int[] ar, int n)
    long sum = ar[0]
    long ans = ar[0]
    for(int i=1 ; i<N ; i++)
        if(sum<0)
            sum=0
        sum = sum + ar[i]
        if(sum > ans)
            ans = sum
            // keep track of index.
    return ans
```

tl: [1 2 3 4].

i/p & o/p must be hard-coded always.

```
mergeSortTester()
    int[] input = {4, -1, -8, 2, 5}
    output = input.quickSort();
    check if output == input.mergeSort();
```

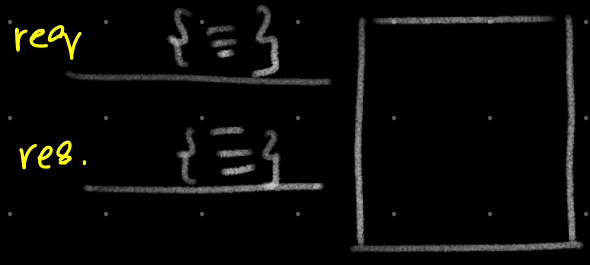
Integration test

Testing of fn by actually making the call to the depe -ndencies.

unit test > # integration tests.

Functional test

E2E working of my flow from client perspective.



unit tests > # integration tests > # functional tests.