TESTING

BY: AMIT KUMAR

PROJECT GROUP: DATA SCIENCE SUITE III

WHY IS SOFTWARE TESTING NECESSARY?

Software Testing is necessary because we all make mistakes.

- Point out the **defects** and errors that were made during the **development phases**.
- It makes sure that the customer finds the organization **reliable** and their satisfaction in the application is maintained.
- To ensure the **quality** of the product.
- Too **expensive** to fix failures in later stages of development

TYPES OF SOFTWARE DEVELOPMENT TESTS

- Unit tests
- Integration Tests
- Functional tests

TIPS FOR WRITING BETTER UNIT TESTS IN JAVA

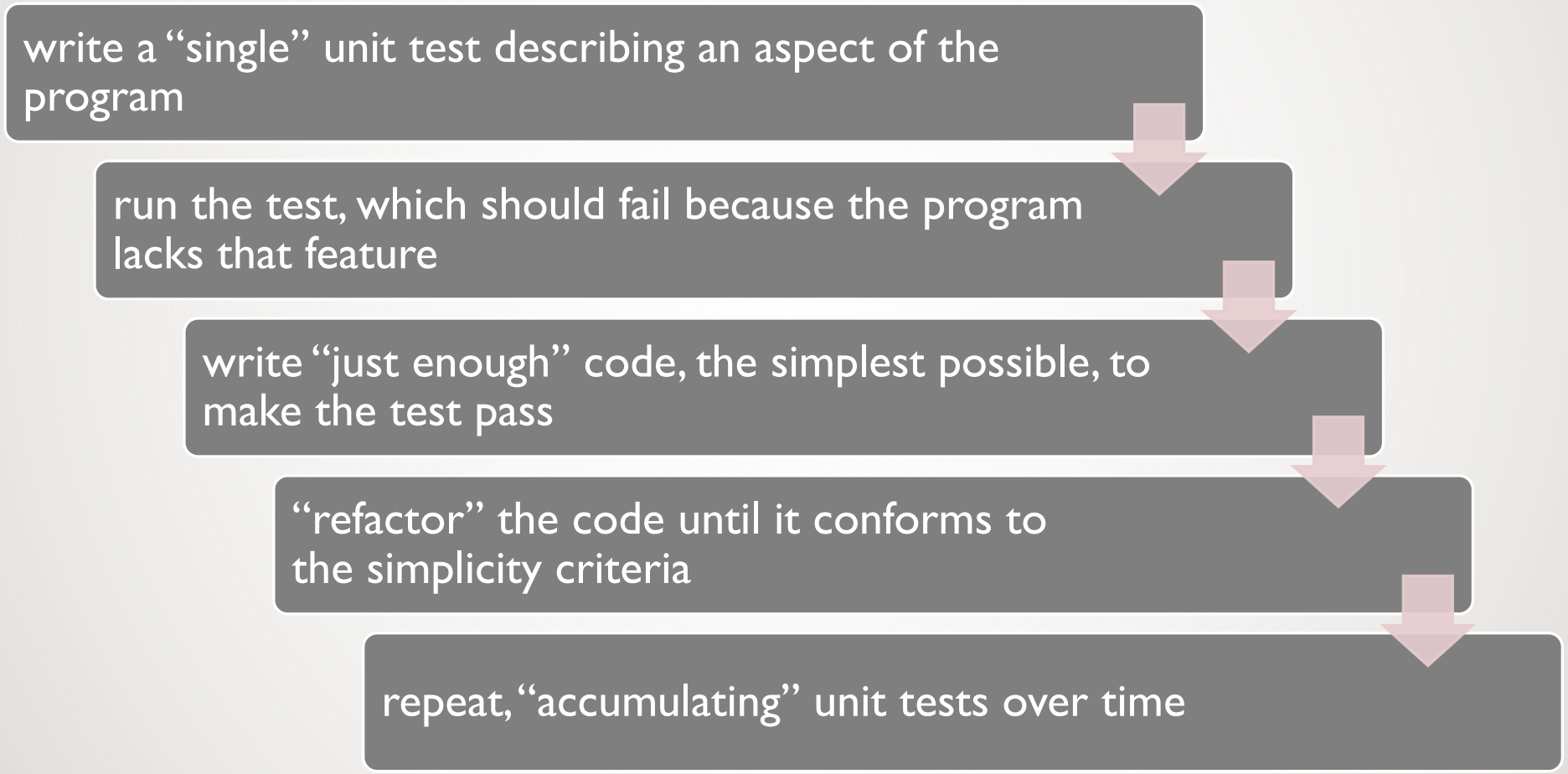
- Use a framework for unit testing
- Use Test Driven Development
- Measure code coverage
- Use assertions instead of print statements
- Build tests that have deterministic results
- Test negative scenarios and borderline cases, in addition to positive scenarios

TEST-DRIVEN DEVELOPMENT

Test-driven development (TDD), is an evolutionary approach to development which combines test-first development where you write a test before you write just enough production code to fulfil that test and refactoring.

- Coding
- Testing (in the form of writing unit tests)
- Design (in the form of refactoring).

TEST-DRIVEN DEVELOPMENT



- Steps for development:
  - Design, write code, test
- In testing steps:
  - Design, write code, write tests, run the tests

JUNIT

Test Runs following are basic steps..

1. Preparation
2. Provide test inputs
3. **Run the tests**
4. Provide expected output
5. **Verify result**
6. **Do something to alert if test failed**

The things that are in bold are something that will always need to be done which is where Junit comes.

```
public class Calculator {

  public int add(int a, int b) {
     return a + b;
  }

}
```

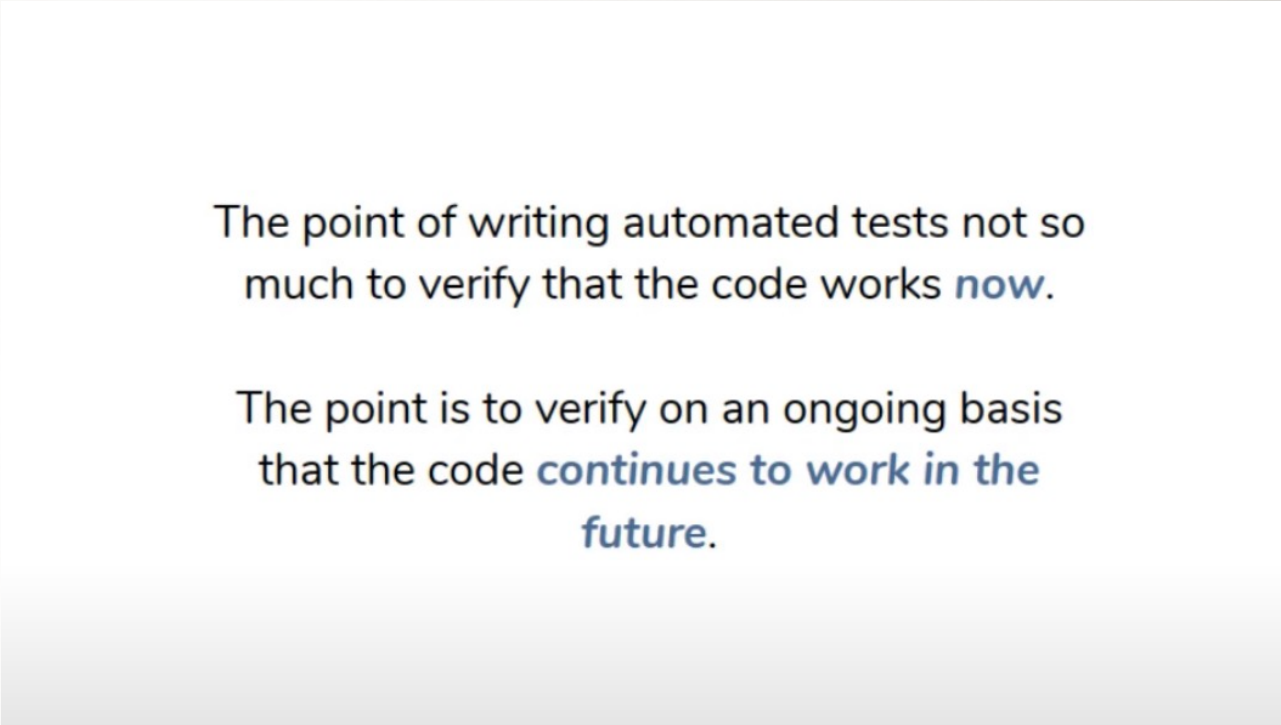
```
Calculator calc = new Calculator();

int sum = calc.add(-1, 3);

if (sum != 2) {
   System.out.println("Test failed");
}
```

JUNIT

It’s the standard for testing on Java. Junit5 is the newest iteration of this framework.

JUnit Framework can be easily integrated with either of the following –

- Eclipse
- Ant
- Maven



EXPECTATION VS REALITY

- Create an instance of the class under test
- Set up inputs
- Execute the code you want to test
- Verify the result is what you expect

CODE COVERAGE

Code coverage is a measurement of how many lines/blocks/arcs of your code are executed while the automated tests are running.

- Jacoco
- Parasoft JTest

TASK

Write a simple code for palindrome detection using TDD with following conditions:

- Test negative scenarios and exception cases, in addition to positive scenarios
- Measure code coverage

- Any questions or suggestions?

REFERENCES

- http://tryqa.com/why-is-testing-necessary/
- https://www.tutorialspoint.com/junit/junit_test_framework.htm
- https://www.youtube.com/watch?v=2E3WqYupx7c&list=PLqq-6Pq4lTTa4ad5JISViSb2FVG8Vwa4o
- https://objectpartners.com/2010/12/21/how-to-write-junit-tests/
- https://www.guru99.com/junit-assert.html
- https://www.agilealliance.org/glossary/tdd/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'tdd))~searchTerm~'~sort~false~sortDirection~'asc~page~1)
- https://www.agilealliance.org/glossary/rules-of-simplicity/
- https://stormpath.com/blog/7-tips-writing-unit-tests-java
- http://www.agiledata.org/essays/tdd.html
- https://medium.com/capital-one-tech/improve-java-code-with-unit-tests-and-jacoco-b342643736ed
- https://www.youtube.com/watch?v=QDFI19lj4OM

Thank you