Unit Testing

Upon completion of this module, a student will be able to

- understand the iterative workflow with test driven development
- understand and explain the purpose of unit testing
- implement simple unit tests using JUnit
- perform additional setup to test more complicated code
- use Roboelectric to test activity functionality

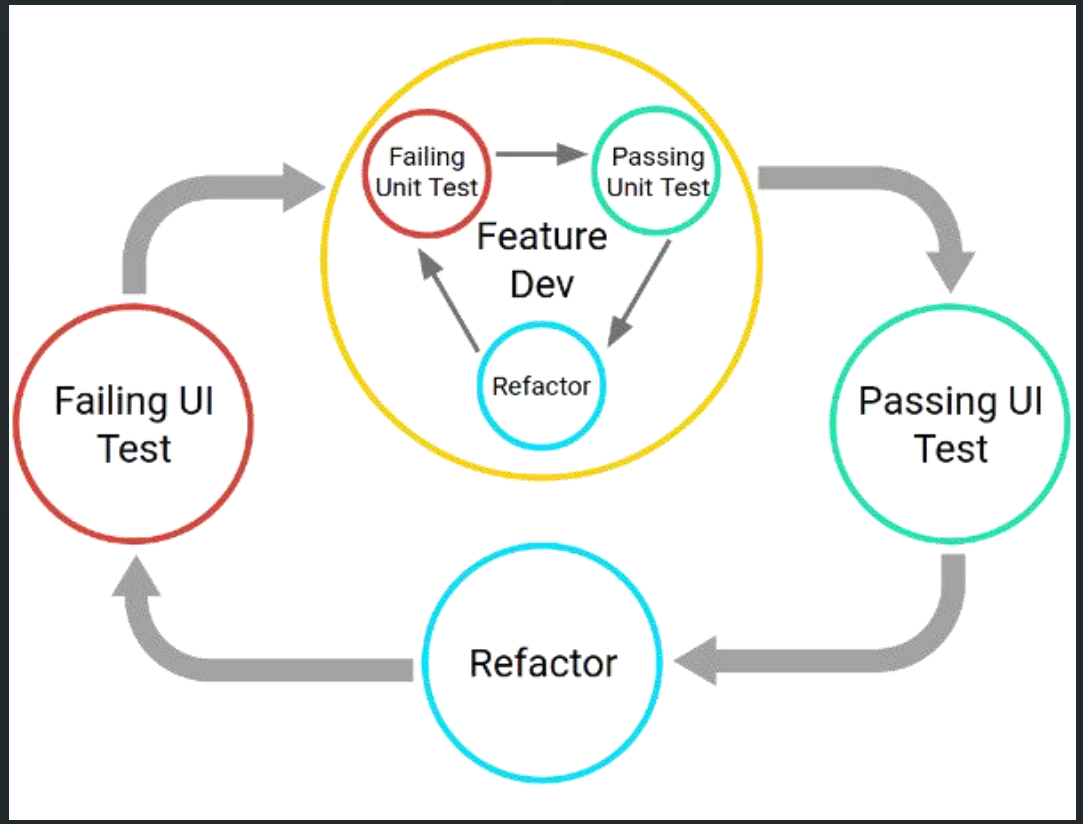
Assignment

- Task
  - Write the backend for a calculator app using test driven design.
- Repo
  - https://github.com/LambdaSchool/Android_UnitTest_Calculator
- Submission
  - Fork on github and submit pull request

A Student Can

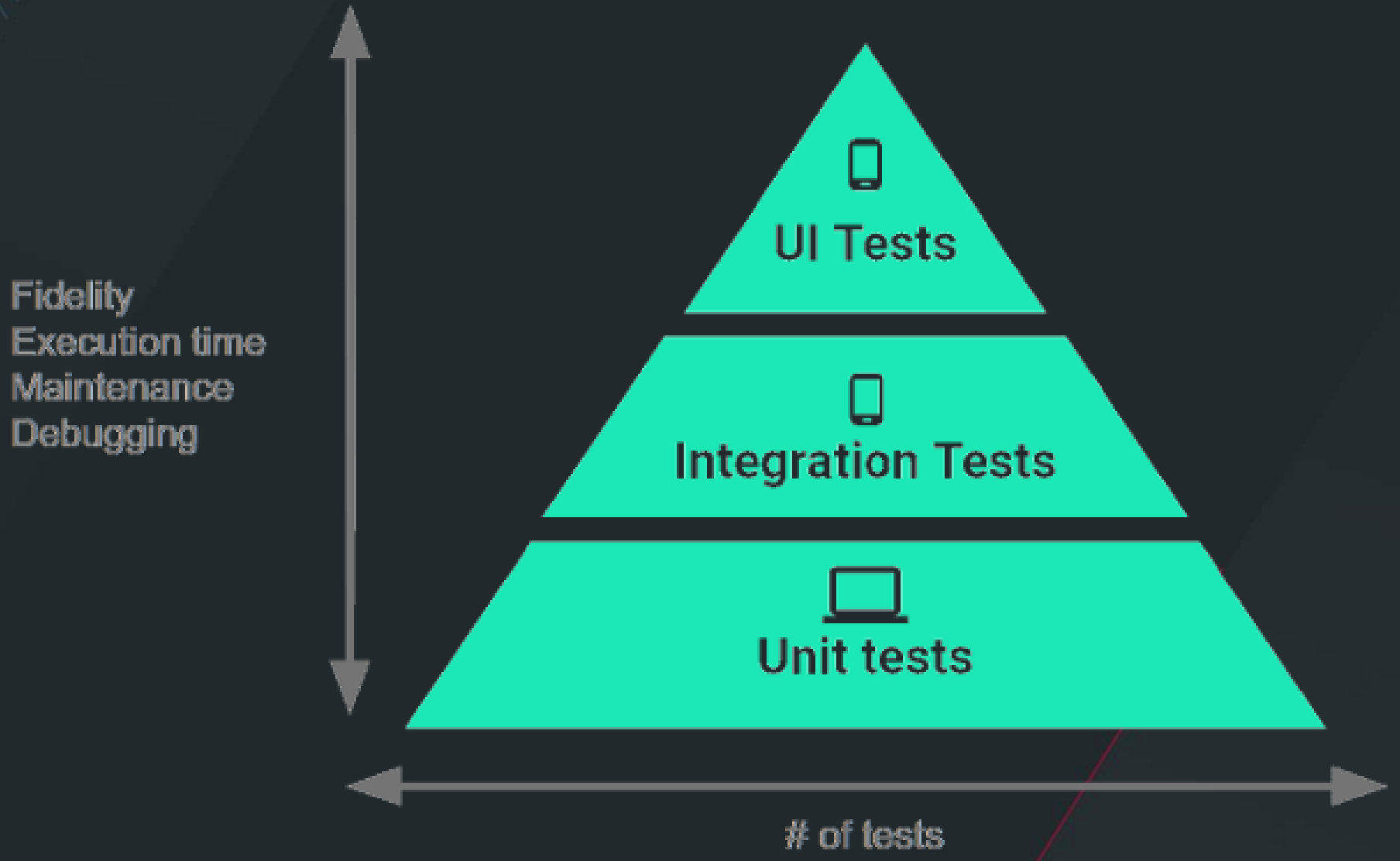
understand the iterative workflow with test driven development

Test Driven Development



- Rapid Feedback on failures
- Safety net

Testing Pyramid



A Student Can

understand and explain the purpose of unit testing

Unit Tests

- Thorough
- Repeatable
- Focused
- Verifies Behavior
- Fast
- Concise

A Student Can

implement unit tests using JUnit

Unit Tests

- Assert method
- https://developer.android.com/reference/junit/framework/Assert

```
@Test public void shouldGenerateCorrectRomanNumeralsWithLargeInput() {
    assertEquals("MMXXII", Main.romanNumeralize(2022));
}

@Test public void shouldGenerateCorrectRomanNumeralsWithLowInput() {
    assertEquals("XIV", Main.romanNumeralize(14));
}

@Test public void shouldReturnEmptyStringWithNullInput() {
    assertEquals("", Main.romanNumeralize(null));
}
```

A Student Can

implement an effective unit test using the three part system

Three Part Tests

```
@Test public void stackShouldBeLifo() {
    // Set up conditions of the test
    Stack<Integer> stack = new Stack<>();
    stack.push(1);
    stack.push(2);
    stack.push(3);

    // Execute the code under test
    int last   = stack.pop();
    int second = stack.pop();
    int first  = stack.pop();

    // Make assertions on the result
    assertEquals(3, last);
    assertEquals(2, second);
    assertEquals(1, first);
}
```

- Set up
- Execution
- Check

A Student Can

use roboelectric to test activity functionality

Roboelectric

```java
@Test
public void shouldStartEditActivity() {
    // Set up conditions of the test
    MainActivity activity = Robolectric.setupActivity(MainActivity.class);
    activity.findViewById(R.id.add_button).performClick();

    // Execute the code under test
    Intent expected = new Intent(activity, EditActivity.class);
    Intent actual = Shadows.shadowOf(activity).getNextStartedActivity();

    // Make assertions on the result
    assertTrue(actual.filterEquals(expected));
}
```

- Test Activity methods
- Open Source but adopted by google