Test Driven Development

Rob Pearse

Classic TDD

Created by Kent Beck

1. Add a test.
2. Run all tests. The new test should fail for expected reasons.
3. Write the simplest code that passes the new test.
4. All tests should now pass.
5. Refactor as needed, using tests after each refactor to ensure that functionality is preserved.

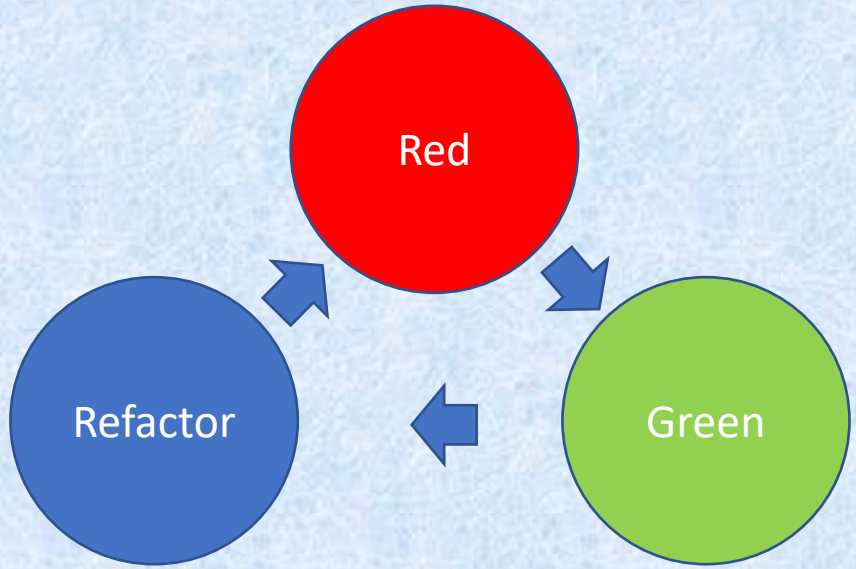
Repeat!

The Three Laws of TDD

Rules formalised by Robert C. Martin (Uncle Bob)

1. You are not allowed to write any production code unless it is to make a failing unit test pass.
2. You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failures.
3. You are not allowed to write any more production code than is sufficient to pass the one failing unit test.

These rules define the cycle of Red, Green, Refactor.



Red

- Keep tests and production code separate.
- Write a single test against a behaviour.
- Structure your test: Arrange, Act then Assert.
- Only one assert per test.
- Make sure the test is failing for the right reason.
- Make sure the test class and name form a full sentence describe the behaviour under test. “CatShould.PurrWhenStroked()”

Green

- Write the simplest code to make the test pass.
  - Fake implementation
  - Obvious Implementation
  - Triangulation

Refactor

- Remove duplicates - rule of three.
- Apply the Object Calisthenics rules.
- Use the IDE to refactor.
- Keep tests green whilst refactoring.

Conclusion

Using Classic Test Driven Development

- Tests will show that the code works.
- Provide a safety net.
- Document as you go.
- Tests lead design decisions.

Thank You