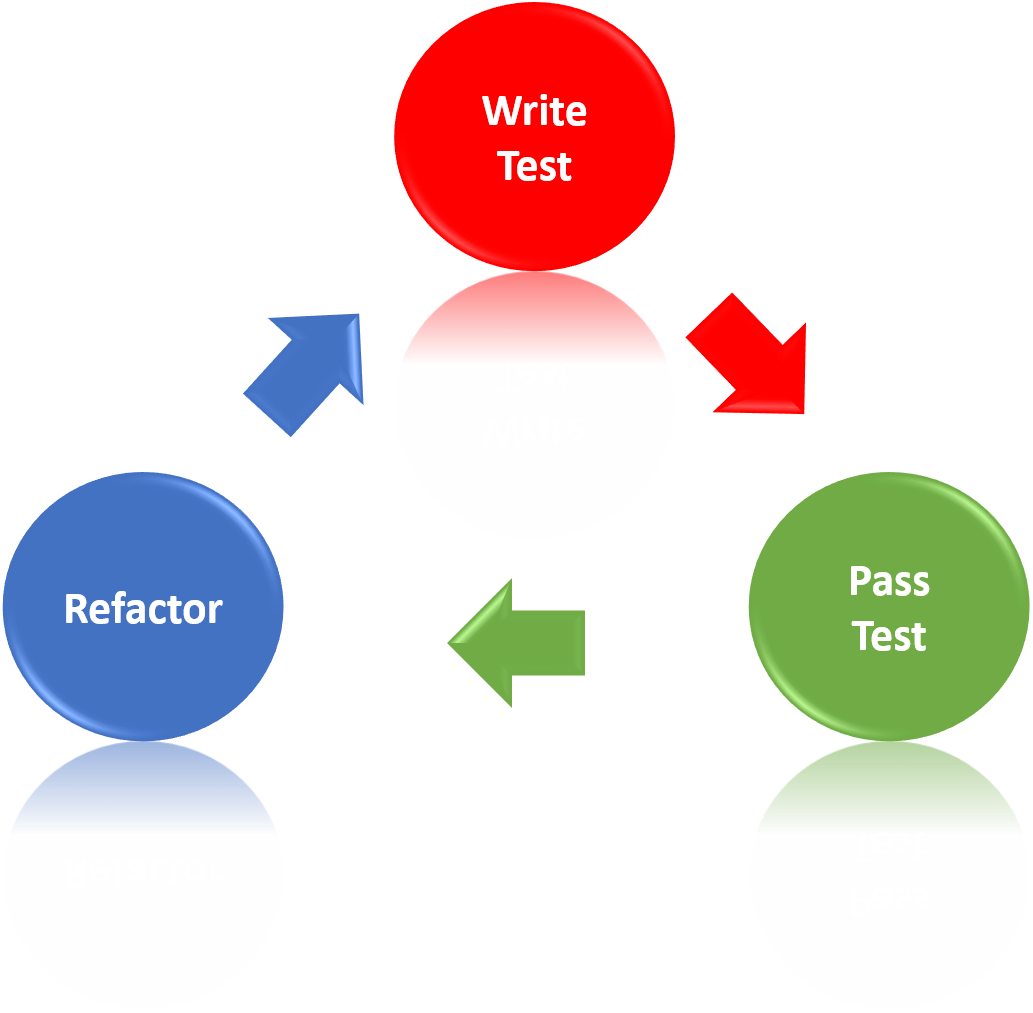
**Test Driven Development (TDD)**

Def: Testing Drives Development (you’re welcome)

What does that mean? All development is only done when you need it, or Just In Time (JIT) and only to pass a specific unit test.

More formally, **TDD** is a development methodology where developers write a feature test that initially fails before writing production code that implements the feature.

1. Write a test for the functionality you want. It should fail.
2. Make the test pass with as little code as possible
3. Refactor



Refactoring: Modifications to the code intended to improve the structure or design without changing functionality.

Cons of writing code first before unit tests:

* Biased results (you look at code there and then write tests for those cases)
* If you run out of time unit tests are skipped
* Likely to ignore or miss some of the tests

TDD Benefits

* Program for specific conditions
* Baby steps toward solving a problem, incrementally adding code
* Refactor with new patterns
* Previous tests hold value with new modifications

[Further reading on TDD](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))

[Further reading on Refactoring](https://sourcemaking.com/refactoring)

[Further reading on JIT development](https://devmethodologies.blogspot.com/2014/04/jit.html)