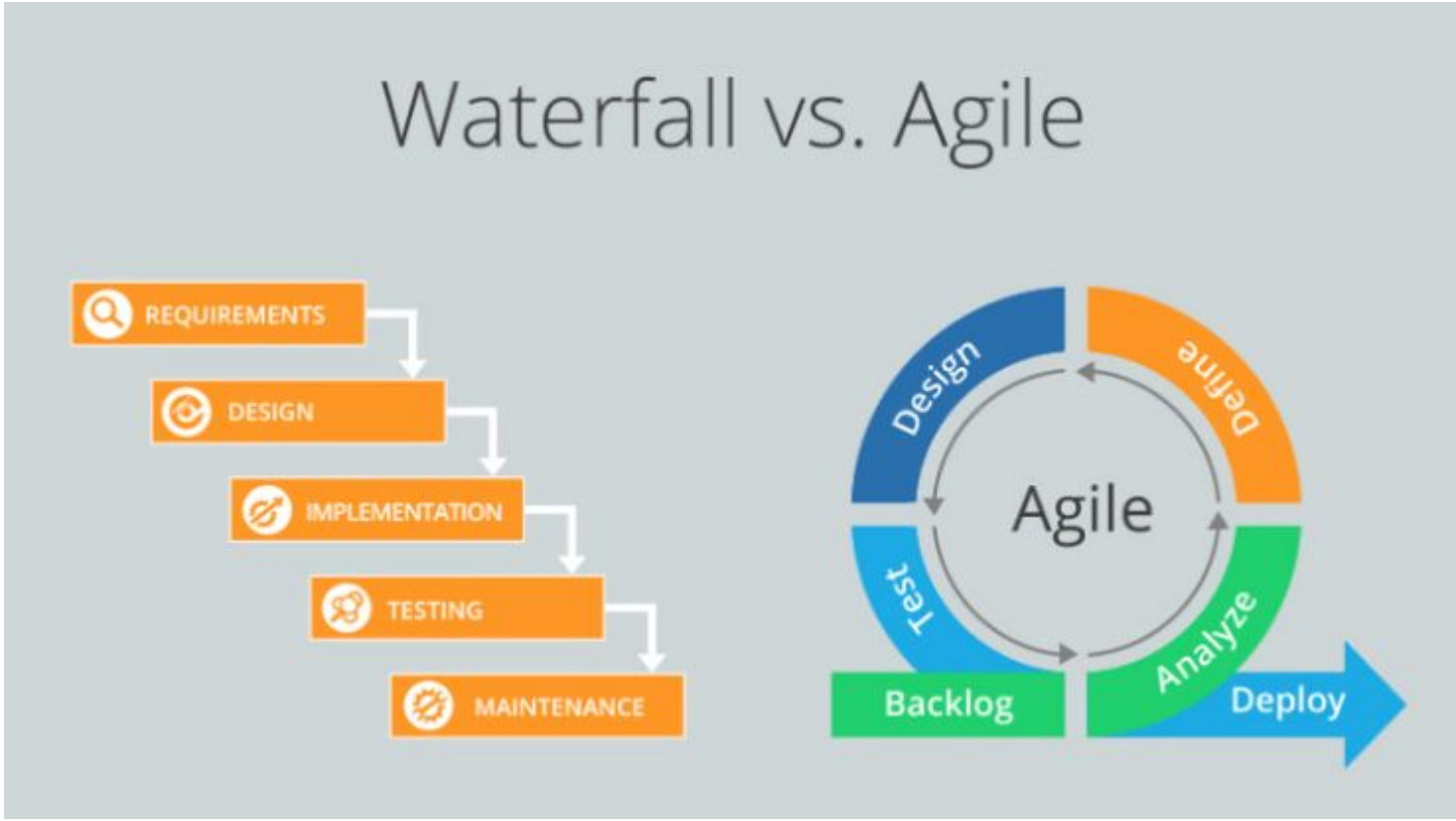
Module 1 - Lecture 14

Unit Testing

Review

- What is an abstract class?
- What is an abstract method?
- What are the differences between an abstract class and an interface?

Software Development Life Cycle



Testing overview

- **Manual Testing** - a tester using the program as an end user would to determine if the program acts appropriately.

- **Automated Testing** - software that performs predefined actions and compares expected outcomes against actual outcomes.

Manual testing

Pros

- short term cost is lower
- more likely to find real user issues
- flexible

Cons

- higher long term cost
- cannot reuse tests
- certain tasks are hard to do manually
- can be repetitive and not very stimulating

Automated testing

Pros

- less expensive
- faster results
- more predictable

Cons

- higher short term cost
- can’t think for itself

Testing overview

- **Exploratory Testing** - explores the functionality of the system looking for defects, missing features, or other opportunities for improvement.

- **Regression Testing** - validates the functionality of the system continues to operate as expected. Typically run after changes have been made to the system.

Testing overview

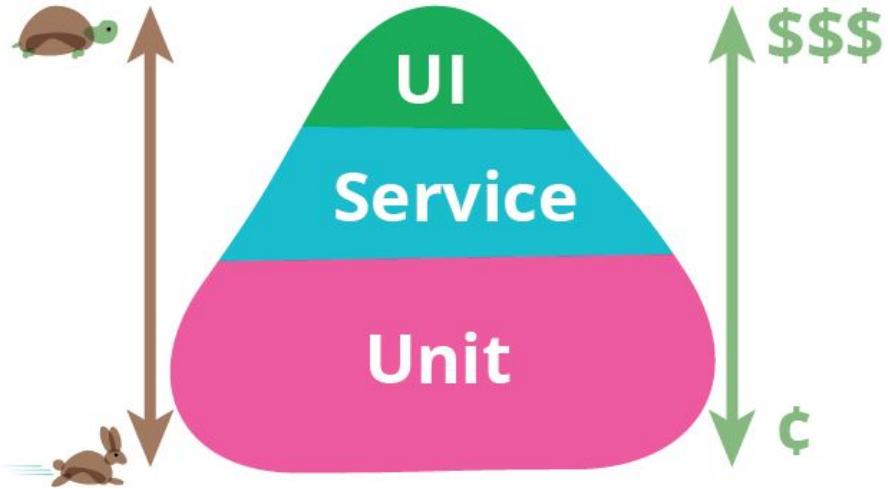
- **Unit Testing** - low level testing performed by programmers that validates individual units of code function as the programmer intended.

- **Integration Testing** - validates the integration between units of code and outside dependencies such as a database or network resources.

- **Acceptance Testing** - validation performed from the perspective of a user of the system in order to verify that the functionality of the system satisfies user needs.

Testing overview

- **Unit** -> **Integration** -> **Acceptance**
- Runtime increases from left to right
- Maintenance and troubleshooting increases from left to right



What are some other types?

Who does the testing?

- Dedicated quality assurance team (exploratory, integration, end-to-end)
- Software developers (unit, integration, performance)
- Business team members (acceptance, accessibility, usability)
- Security team

Properties of a unit test

- **Fast** - the elapsed time of a unit test should be measured in milliseconds.
- **Reliable** / **Repeatable** - if a test passes/fails once, it should pass or fail every time, assuming the code hasn't changed.
- **Independent** - one test should not have an impact on another. A test should not require another test to run in order to succeed.
- **Obvious** - it should be easy to determine why a test failed.

Three part test

- **Arrange** the conditions of the test, such as setting up data.
- **Act** upon the action of interest i.e. the thing that we are testing.
- **Assert** that the expected outcome(s) occurred i.e. a certain value returned, a file exists, etc.

Unit Test Best Practices

- No external dependencies
- One *logical* assertion per test
- Test code is of the same quality as production code
- Test boundary cases
  - Empty arrays/lists, nulls, negative numbers
- At most one test class per class file

Let's Code!

Code Coverage

Code coverage is the percentage of code which is covered by automated tests.

Code coverage measurement simply determines which statements in a body of code have been executed through a test run, and which statements have not.

We measure code coverage for the following reasons:

- To know how well our tests actually test our code
- To know whether we have enough testing in place
- To maintain the test quality over the lifecycle of a project

QUESTIONS?