* Challenges/Costs of developing software today
  + analysis, design, development, testing, integration, maintenance
    - maintenance has the highest cost out of these phases
* Test Driven Development
  + test: a procedure intended to establish the quality, performance, or reliability of something, especially before it taken into widespread use
  + testing can help answer the following:
    - does it solve the requirements?
    - does it respond correctly to ALL input?
    - acceptable performance?
  + Test-Driven Development:
    - "a **software development process** that relies on the repetition of **a very short development cycle**: requirements are turned into very specific test cases, the then software is improved to pass the new tests, only." – Wiki
    - "**Red** – **Green** – **Refactor**"
      * a cycle that is continued while developing software
      * Red: write a test that fails
      * Green: minimal work necessary to make the test pass
      * Refactor: cleanup code while having code pass through each refactoring
* Different Types of Testing
  + Software Testing
    - Composed of *Units of Computation* (e.g., classes, functions, dependencies, …)
    - Unit Testing: verifies compositions of units are behaving properly together
    - Integration Testing: aimed to catch issues not caught in isolated unit testing
    - Acceptance Testing: verifies the software from the user's point of view
      * does the software behave the way it's supposed to?
  + Testing styles
    - Blackbox: aim to verify the behavior of the software by giving it different and only observing the output
      * views the component being tested as an opaque box
      * test has no insight into inner working of the component
    - Whitebox: see how the components are getting the work done to give the output
      * powerful technique when testing systems with lots of dependencies and interconnected components
* Testing Frameworks and Tools
  + xUnit Frameworks
    - SUnit (Smalltalk)
    - Junit (Java)
  + UI Frameworks
  + System Frameworks
    - Simian Army
      * Chaos Monkey, Latency Monkey, Janitor Monkey
* Testing Concepts
  + BeforeEach
    - setup before each test is ran
  + AfterEach
    - ran after each test in the test suite is ran (used for cleanup)
  + Verification concepts
    - **Assert**: allows the test to tell what values we expect to determine what values passes and fails the test
  + Test Execution
    - synchronous vs asynchronous
* Dependency Injection
  + in software engineering, **dependency injection** is a technique whereby one object supplies the dependencies of another object
    - a *dependency* is an object that can be used
    - an *injection* is the passing of a dependency to a dependent object that would use it

OBJECT

*Injection*

Dependency

Dependency

* + Constructor Injection
    - providing dependencies though a class constructor
  + Property/Setter Injection
    - using a property or setter method to inject a dependency
  + Interface Injection
    - client defines interface that describes how dependencies are injected into it
* Test Double
  + a "generic term for any kind of pretend object used in place of a real object for testing purposes"
  + Stubs
    - a test double that provides canned answers to calls made during the test
  + Mocks
    - a test double that is pre-programmed with **expectations** which form a specification to be verified
* Best Practices
  + treat test code like production code
  + focus only on necessary values and results
* Testing Anti-Patterns
  + dependencies between tests
    - execution order of tests should not matter
    - interdependent tests cause cascading failures and false positives
    - serial execution versus parallel execution
  + testing implementation details
    - tests should focus on the "what" NOT the "how"
    - testing implementation details leads to brittle tests that break when refactoring
  + slow-running tests
    - prevents rapid red/ green/refactor cycles
    - warning-sign that code may be too coupled
    - warning-sign that code may NOT be very testable
* Limitations of TDD
  + possible *holes* in tests
  + TDD is NOT sufficient by itself
    - deployment verification
    - network changes
    - integration testing