A **test** is the object that is returned by your typical **assertion statements**.

A **test method** is a public instance method that takes no arguments and returns a *test*.

A **fixture** is a public class that contains test methods.

A **named constructor** is a static no-parameter method that returns an instance of its containing class.

A **fixture factory** is one of:

* A public **default constructor** on a *fixture* class.
* A public **named constructor** on a *fixture* class.

A **test case** is a *fixture factory* and a *test method* within that *fixture*.

A *test case* can be **executed**. To execute a *test case*,

1. The test case’s *fixture factory* is executed, producing a fresh *fixture* for the *test case*.
2. The test case’s *test method* is executed, returning a *test*.

A *fixture* is **unreachable** if it has no valid, accessible *fixture factory* defined.

A *test method* is *unreachable* if it is defined on an *unreachable fixture*.

**Discovering Tests**

1. The test finder is passed an assembly.
2. Find all public classes in the assembly.
3. Find all the classes that contain *test methods*. These classes become *fixtures*.
4. Find all *fixture factories* in each *fixture*.

**Brainstorming**

* Inheritance feature: subclasses should be instantiated with base-class tests.
  + Subclass tests should be grouped with base-class fixtures.
  + Should this happen when base class has no tests?
* Nested class composition
  + Nested fixtures should be displayed as groups under containing class
  + Nested fixtures with public ctors that take instance of outer class should be instantiated.
* Visual Studio Test Adapter
  + The part I’m looking forward to least, but also the most important part.
  + We need to get these tests displaying in Visual Studio, and not just the console.
* Visual Studio intellisense upgrades
  + Highlight valid fixture factories.
  + Highlight valid test methods.
  + Warn/error on unreachable test methods.
* Support test methods that return common monads of Test
  + IEnumerable<Test> methods should act like NUnit theories.
  + Task<Test> methods should be automatically async.
  + IObservable<Test> methods should also be async.
  + Func<Test> methods should be lazily-evaluated.
* More assertions
  + Assertions for collections.
  + More equality assertions.
  + Assertions for numbers.
    - IsNan
    - IsPositive
    - IsNegative
    - Floating-point equality with tolerance
  + Assertions for reflection.