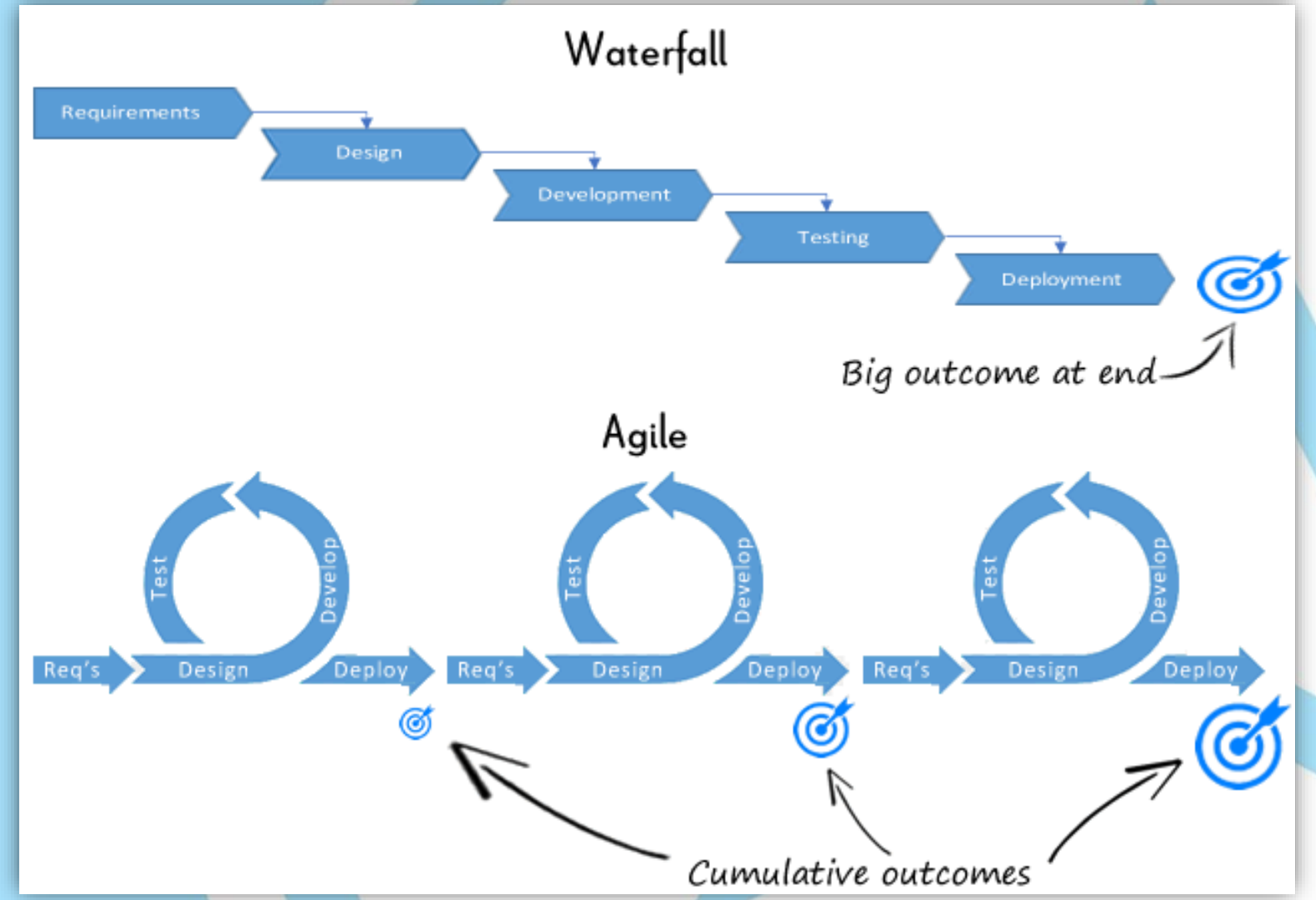
Module 1 Day 14

Unit Testing

Software Development Life Cycle (SDLC)



Testing Methods

- Manual
  - Easy to start
  - Can evaluate subjective qualities (user-friendliness)
  - Difficult to be execute consistently and frequently
  - Good for Exploratory Testing
    - exploring functionality, looking for defects, missing features, improvement opportunities
- Automated
  - Higher cost to get started
  - Fast, efficient and frequent
  - Consistent and accurate, repeatable
  - Good for Regression Testing
    - validating that existing functionality continues to operate as expected

Types of Testing

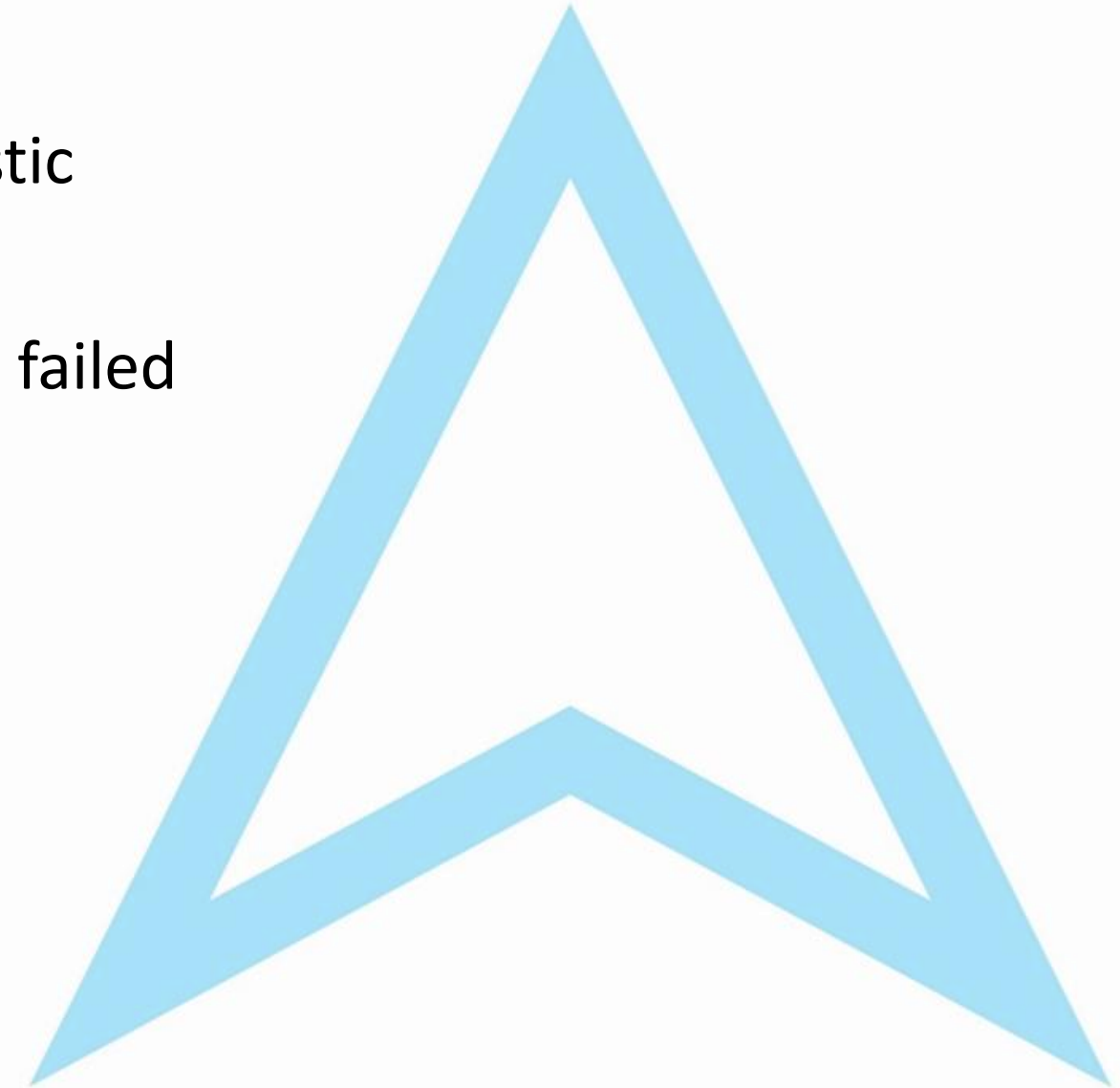
- Unit Testing
  - Verifies parts of an application independently from other parts (isolated)
  - Narrow scope (focused)
  - Tests many cases (deep)
  - Fast!
- Integration Testing
  - Verifies interactions between multiple components
  - Broader scope
  - More difficult to test thoroughly
- End-to-end Testing (E2E)
  - Verifies interaction from end-user through a transaction and back to end-user
  - Broadest scope
  - Most difficult to test all possible combinations of scenarios
  - Slowest to run (for both setup and execution)

Other Types of Testing

- User Acceptance Testing (UAT)
- Performance / Scalability
- Security
- Usability
- Accessibility

Unit Testing should be…

- Fast (milliseconds each)
- Reliable & Repeatable; Deterministic
- Independent of other tests
- Obvious; Easy to determine why it failed



How and What to Test

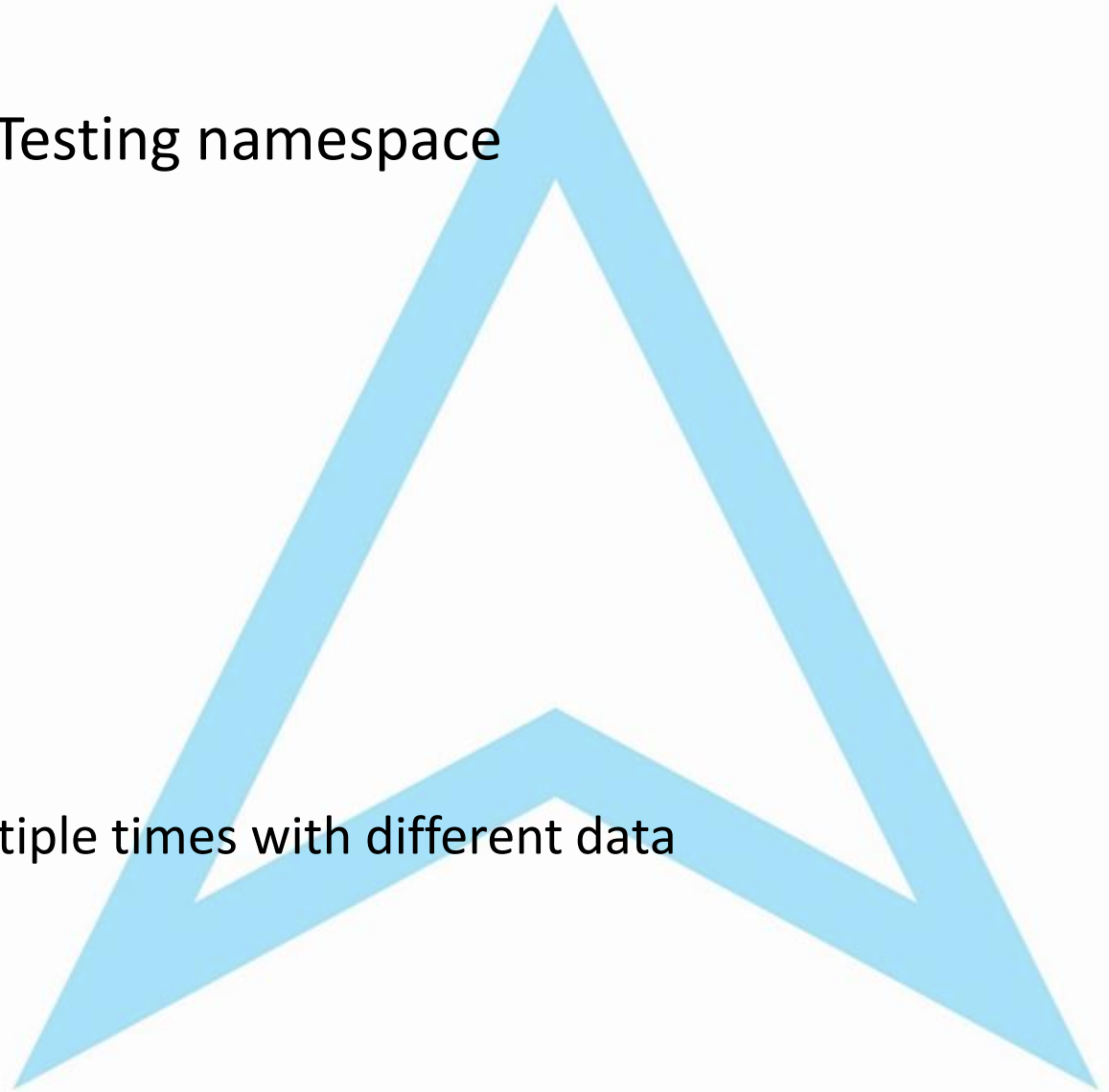
- “Happy Path”
  - Test with expected inputs for expected results
- Boundary cases (Edge cases)
  - Is there an if statement?
    - Test around the condition that the if statement tests (boundaries)
  - Is there a loop?
    - Test arrays in the loop that are empty, only one element, lots of elements
  - Is an object passed in? A string? A number?
    - Pass in null, an empty object, an object missing values that the method expects
    - Pass in an empty or null string
    - Pass in negative numbers or zero

Unit Test Structure – The 3 A's

- Arrange
  - begin by arranging the conditions of the test, such as setting up test data
- Act
  - perform the action of interest, i.e. the thing we're testing
- Assert
  - validate that the expected outcome occurred by means of an assertion

xUnit Testing Framework

- Microsoft.VisualStudio.TestTools.UnitTesting namespace
- Creating a Test Project
- Test Classes
  - Generally one for each class under test
  - [TestClass] attribute
- Test Methods
  - Represent a test case or scenario
  - [TestMethod]
- Data Test Methods
  - Allow you to execute a test method multiple times with different data
  - [DataTestMethod]
  - [DataRow]



Testing the Bank Application - 1

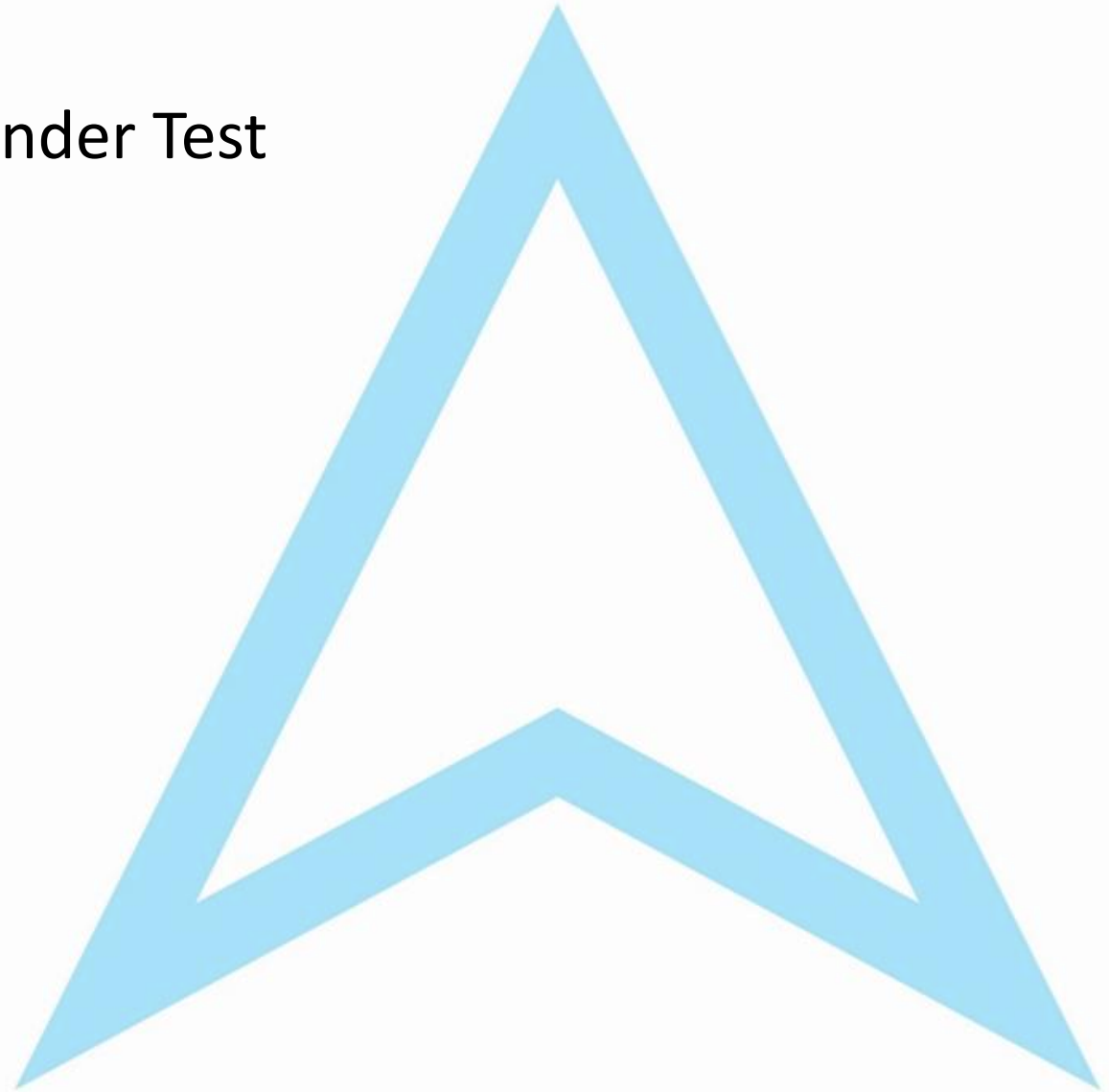
- First, should we ever be able to create a BankAccount?
- Create a Test Project
- Setup the assembly dependency
- Run the test!

Assert Class

- https://docs.microsoft.com/en-us/visualstudio/test/using-the-assert-classes?view=vs-2015
- AreEqual / AreNotEqual
- IsTrue / IsFalse
- IsNull / IsNotNull
- ThrowsException
- And many more

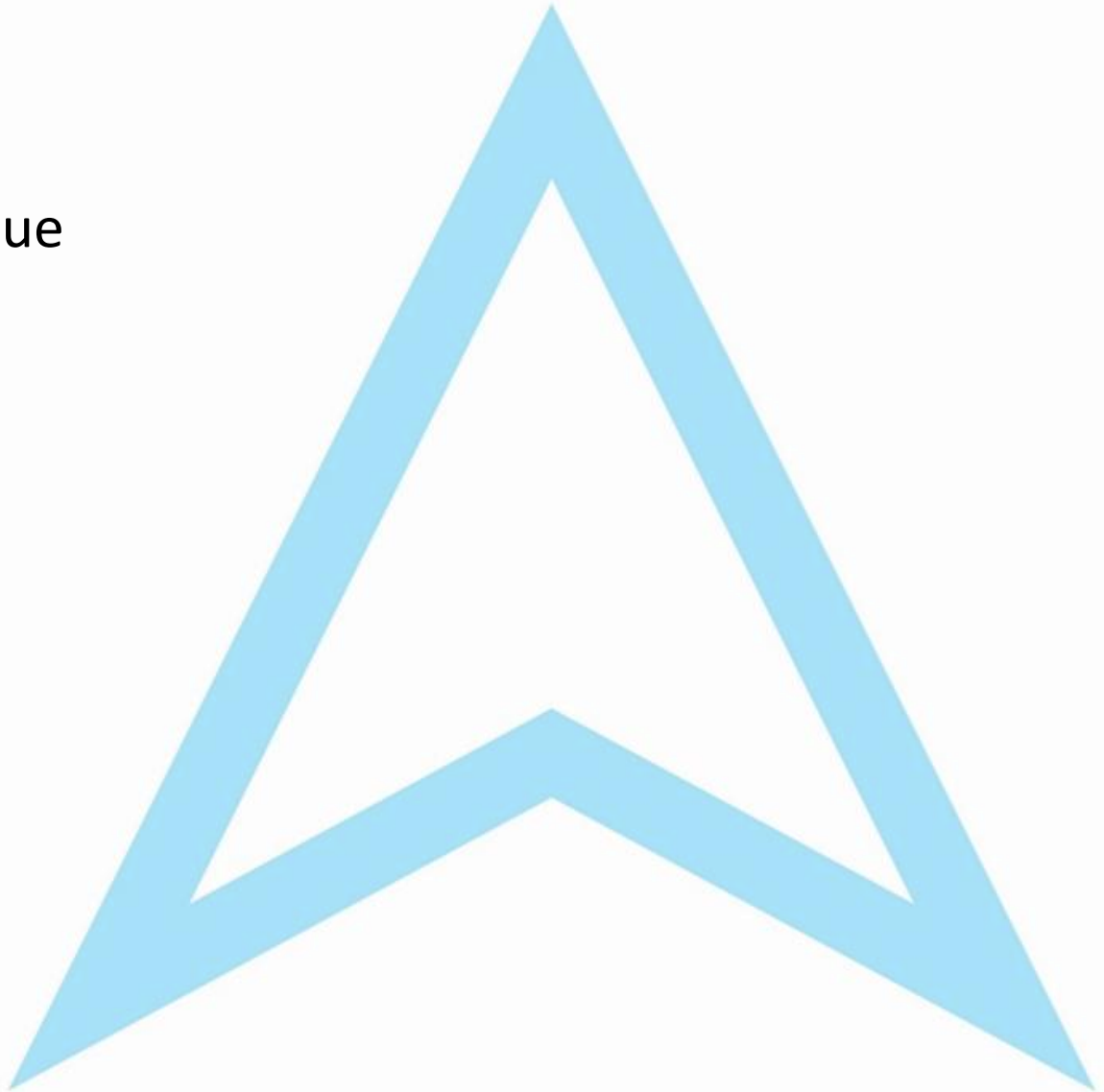
Testing the Bank Application - 2

- Create a Test class for each Class Under Test
  - Constructors and default values
  - Derived properties
  - Methods
    - Business rules
    - Edge cases
- BankAccount, IAccountable
- Savings Account
- Checking Account
- Credit Card Account



Other Assert Classes

- CollectionAssert
  - AllItemsAreNotNull, AllItemsAreUnique
  - AreEqual, AreEquivalent
  - Contains, IsSubsetOf
- StringAssert
  - Contains
  - Matches, DoesNotMatch
  - StartsWith, EndsWith



Testing the Bank Application - 3

- BankCustomer
  - VIP rules
  - Collection of accounts