**TestNG:**

* Open Demos>Java>TestNG in IntelliJ
* Navigate to src>main>java and show students the Calculator files allows us to perform simple calculations on 2 operands
* Open pom.xml and add a tag for dependencies and paste in the dependency from https://mvnrepository.com/artifact/org.testng/testng/7.5
* Open src>test>java>CalculatorTest.java and add a new test for the add method
  + Add comments for arrange, act, and assert and write each section of the test under its comment
  + Write the assert first, which will take the variables expected and actual
  + Declare actual and define expected in arrange
  + Set actual to the value returned by calling the add method
* Write another test for the subtract method that only uses the assert statement by passing in the actual expected value and the call to the subtract method
* Write a similar test for the multiply method, which doesn’t exist
  + Show that the test fails, for the right reason (multiply doesn’t exist)
  + Open Calculator.java and write a method named multiply that always returns 4 (minimum to get pass), re-run the test
  + Test should still fail b/c there are no arguments, add arguments and re-run the test
  + The test passes, but still doesn’t technically do what we want, write a test to ensure it can handle negatives
  + The test will fail until we fix the method to return num1 \* num2, re-run and the test passes
* Write a happy path test for the divide method
* Write a failure path test to check that we receive an error class
  + The test will fail because we receive positive infinity (one of the special numeric values that floating point numbers return)
  + Add an if statement that throws an error if the denominator is 0

**Implementing Constructor Injection:**

* Open Demos>Java>dependency-injection in IntelliJ and navigate to src>main>java
* Create a class called Client
  + Declare and assign a field of type ServiceA
  + Create a method called doSomething that returns void
  + In doSomething, call service.getInfo()
* Create a class called ServiceA
  + Create the getInfo method and make it print “Service A info”
* In Main, create an instance of Client and call doSomething on it – run the program
* In Client remove the assignment to service
* Create a constructor which takes a Service
* Create a Service interface with the abstract method getInfo
* Make ServiceA implement the Service interface
* In Main, create an instance of ServiceA and pass it into the Client constructor – run the program
* Create a class called ServiceB that implements Service and getInfo prints “Service B info”
* In Main, create an instance of ServiceB and pass it into the Client constructor – run the program
* Add TestNG to the project and create a test class for Client
* In Client, make doSomething return a String
* In Service, ServiceA, and ServiceB make getInfo return a String
* In ClientTest, write a test to check that client returns mock data given a mock service
  + Expected should be “Mock Service info”
  + Create an instance of MockService
  + Create an instance of Client and pass mockservice in
  + Set actual to client.doSomething()
  + Assert that actual equals expected
  + Create the Mock service class – run the tests

**Implementing Setter Injection:**

* Open Demos>Java>dependency-injection in IntelliJ and navigate to src>main>java
* In Client, remove the constructor and create a setter for service instead
* In Main, remove the instance of service passed into the client constructor
* In ClientTest, comment out the test
* In Main, before the call to client.doSomething, call client.setService and set it to one of the service instances
* Be sure to print client.doSomething – run the program
* In ClientTest, write a new test for checking that doSomething returns mock data from a mock service
  + Copy and paste the previous test and remove the service passed to the client constructor
  + Add a call to setService and pass in the mockService – run the tests

**Implementing Method Injection:**

* Open Demos>Java>dependency-injection in IntelliJ and navigate to src>main>java
* In Client, remove the setService method
  + Remove the service field
  + Pass a Service into the doSomething method directly
* In Main, remove the call to setService
* Pass one of the instances of Service into doSomething
* Comment out the test in ClientTest– run the program
* In ClientTest, copy and paste the test
  + Remove the call to setService
  + Pass the mock service into the call to doSomething – run the test
* In Client, create a new method called doSomethingElse that takes an IntService and returns an int
  + Return service.getInfo
* Create an IntService interface and a class that implements it (ServiceC)
* In Main, create an instance of ServiceC
* Print a call to doSomethingElse and pass in service – run the program
* In ClientTest, add a test for doSomethingElse to return 10 given a mock IntService
  + Expected = 10
  + Declare actual
  + Instantiate MockIntService
  + Instantiate Client
  + Assign actual to client.doSomethingElse(mockService)
  + Assert actual = expected
* Create an MockIntService class that implements IntService
  + getInfo() should return 10
* Run the tests