# Unit 3 - Exercises

The following exercises will requires additional files. You can download the files [here](https://thethirstycoder.gitlab.io/adev-2005-learn/unit_3/downloads/adev-2005_unit_3_exercise_files.zip).

## Setup

1. Download the exercise files using the link above.
2. Extract the .zip file downloaded in step 1.
3. Examine the extracted file(s).

### Download Files

1. ADEVPayroll.dll - Contains the HourlyEmployee class you will be unit testing in the following exercises.
2. ADEVPayroll.xml - Contains the XML documentation for the .dll.
3. HourlyEmployeeClassDocumentation.pdf - Conains a formatted version of the documentation for the HourlyEmployee class.

Create a new Visual Studio solution called ADEV2005Unit3 containing a Test Project project called Unit3Exercises.

After you create a Test Project, a code file is included in the project template. This file is typically called “UnitTest1.cs” and contains a class for coding unit tests. Before proceeding, rename the “UnitTest1.cs” file to “HourlyEmployeeTest.cs”.

**EX1**. Add a reference to an assembly.

1. Right-click the “Unit3Exercises” project within Solution Explorer and choose Add Reference… from the context menu.
2. Select the “Browse” tab.
3. Locate the “ADEVPayroll.dll” file extracted from the download .zip.
4. Select the file.
5. Click the “Ok” button.
6. Verify the reference was created by expanding the “References” folder under the “Unit3Exercises” project in the Solution Explorer.

Before coding any unit tests, you must open the unit test class file and add any using statements required to access the code you are testing. The documentation for the HourlyEmployee class is included in the download file. You will need to read the documentation of the HourlyEmployee class to understand how the methods/properties of the class are implemented in order to know how many unit tests to create and how to code them.

### Additional Information

The HourlyEmployee class has 4 fields:

1. \_employeeID
2. \_name
3. \_hoursWorked
4. \_rateOfPay

**EX2**. Create a unit test method for the constructor of the HoulyEmployee class.

**Note**: The HourlyEmployee class is compiled within the ADEVPayroll.dll assembly.

1. Delete the auto-generated unit test method called “TestMethod1”.
2. Code a test method named “Constructor\_Test”.
3. Code the unit test. You are testing that after constructing the object, the object’s state is what it should be based on the arguments you used.

**EX3**. Run the test.

1. Right-click anywhere within the unit test method.
2. Choose “Run Tests” from the context menu.

If the test failed:

1. Analyze the unit test code.
2. If unit test code looks good, analyze method code.
3. Fix the problem.
4. Re-run the test.

### Questions

1. When a unit test evaluates to “Pass”, does that mean the method’s code is correct? Why or why not?

**EX4**. Testing for an Exception.

When testing any piece of code, all possible outcomes must be tested. When a method can throw an Exception, this outcome must be tested.

1. Add a new unit test method to the class called “ConstructorEmployeeIdBlank\_Test”.
2. Write the unit test to generate the exception and evaluate the result.

Repeat these two steps for the other three exceptions in the method. Name the unit test methods appropriately.

**EX5**. Unit testing a property.

Properties are method constructs and require unit testing. No matter how trivial the implementation, a unit test is required for each outcome.

1. Code unit tests for the properties of the HourlyEmployee class.

**Note**: Some properties throw exceptions.

**EX6**. Unit test methods.

1. Code unit tests for the methods of the HourlyEmployee class.