# Exercises: Unit Testing

This document defines the **in-class exercises** assignments for the ["High-Quality Code" course @ Software University](https://softuni.bg/courses/high-quality-code).

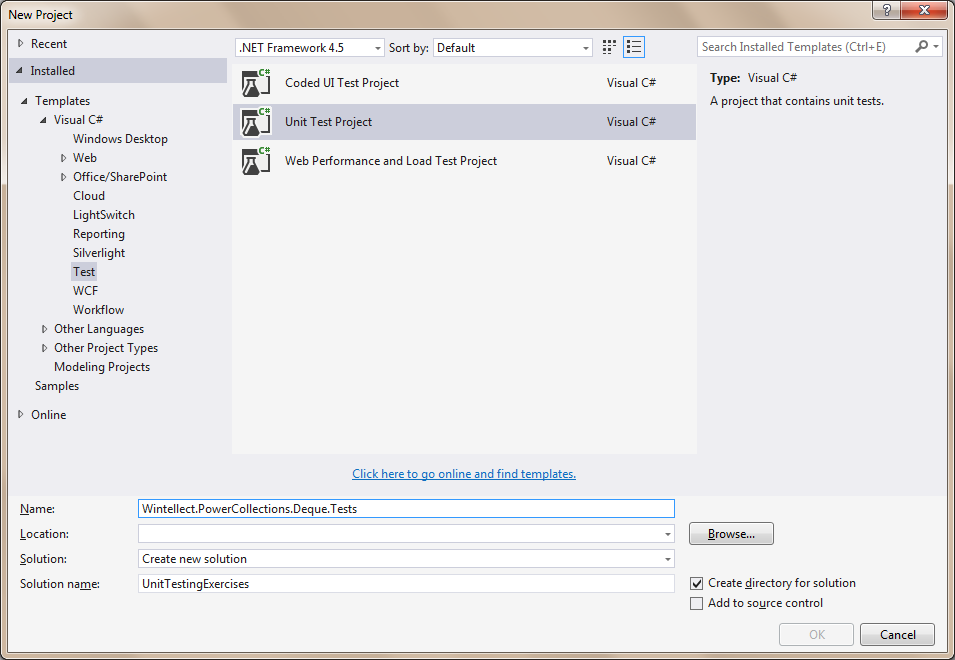
## Get to Know Wintellect PowerCollections

In these exercises, you will be testing Wintellect's PowerCollections library. You can have a look at its source code at CodePlex: <https://powercollections.codeplex.com/>. Here you can view the documentation and full source code of the library. It provides implementations of various useful data structures (collections) in C#, such as multi dictionaries, deques (double-ended queues) and big lists.

Select one of the classes (the **Deque** and **BigList** classes are a good starting point). This is the class you will be testing.

## Create a New Test Project

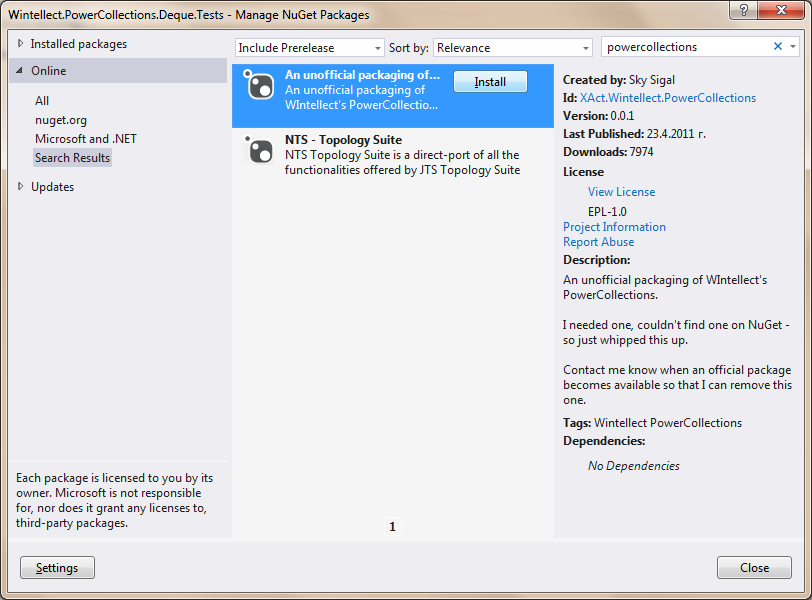
In Visual Studio, select **File > New > Project…** In the left part of the dialog box, select **Visual C# > Test** and then select **Unit Test Project**. Name your new project and solution accordingly.



## Install the PowerCollections Library via NuGet

Right-click your unit test project and select **Manage NuGet Packages…** After that, select **Online** in the left menu and search for "powercollections" in the top right search bar. Select the package and install it. You may need to accept a license agreement.

You will now have a reference to the **PowerCollections** library.



## Select the Methods to Test

Since the classes are too big, you won't need to test them all. You can only select some methods to test.

Refer to the source code to learn what tests you should write. Test the normal and the unusual cases. For example, if a method expects an argument to be initialized (i. e. not **null**), be sure you test it both **with and without** null.

Example: **Deque.AddToFront(T item)**

Cases:

* Add to an empty deque, ensure element is added
* Add to a full deque, ensure element is added and is first
* Remove some elements from a full deque, and then add one. Ensure removing before adding elements does not break the data structure
* Add many random values, ensure they are in the correct order

## Test the Selected Methods

Write unit tests for the methods you have chosen. Follow the unit testing guidelines.

If a test doesn't pass, make sure your test is right first. Then, if you think you have found a bug in the library, you may submit it to the developers.

## \* Analyze the Code Coverage

Try to analyze the code coverage for the methods you have selected. The code you're testing is in an external library, so you'll have to find a way to include it in the code coverage analysis.

Aim at 80% code coverage (or more) for the methods you are testing. Since you are probably not testing the whole class, the coverage for the class will be lower; don't worry about this.