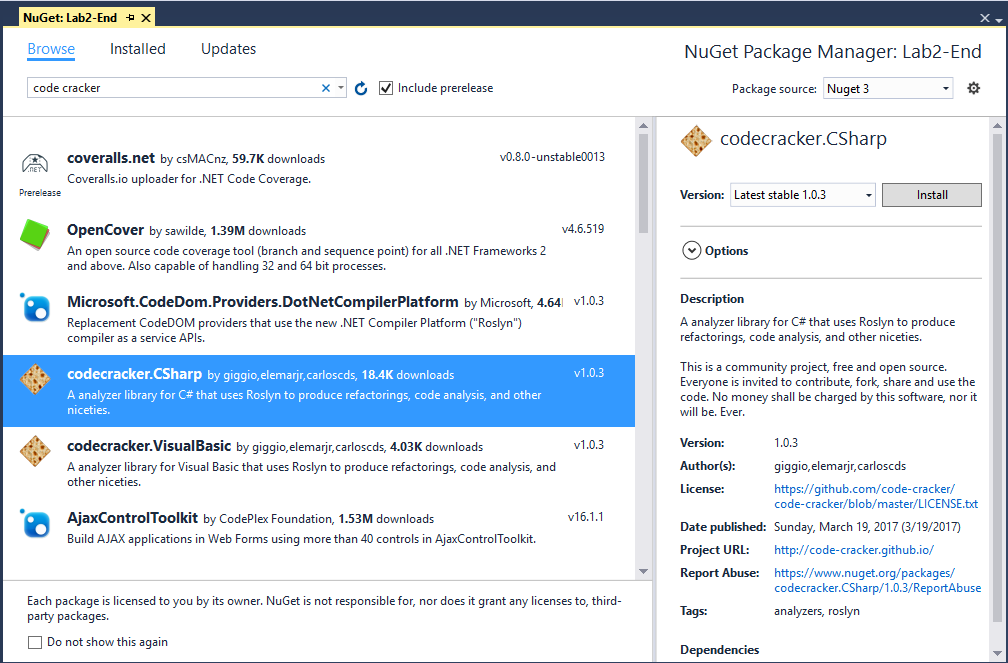
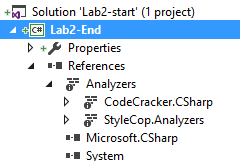
# Lab 2 – Adding external analyzers

In this lab, we’re going to continue improving the code file that we started in Lab1. This time, we’re going to add a couple open source analyzer packages and see what additional improvements they can recommend for our code. Begin by opening the Lab2-Start.project file.

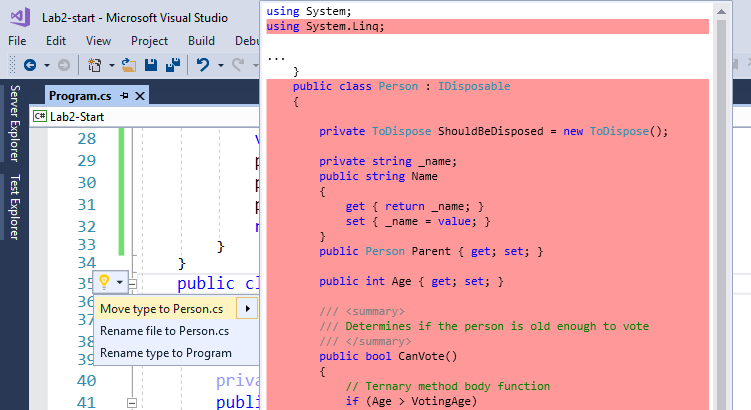
Right click on the project and select “Manage NuGet Packages”. Click on the Browse tab and search for “Code-cracker”. Locate the codecracker.CSharp package and install it. Do the same for the StyleCop.Analyzers package.



If you expand the references tab of your project, you should see a folder for “Analyzers” and under that entries for CodeCracker.CSharp and StyleCop.Analyzers. Analyzers need to be installed on each project that you wish to use them on. This allows for easy deployment via the NuGet package manager. It also allows library authors to ship analyzers with their libraries when domain specific or best-practice issues are known.

Build your project and notice that Visual Studio is now reporting quite a number of new errors, warnings, and info messages. Use the table below to guide you through locating the various issues and applying the appropriate code fixes for this lab. In many cases, the option you choose may be subject to your preferred coding standards, but if you select different fixes than recommended here, the subsequent fixes may not light up as some fixes are dependant on applying the previous fix.

To illustrate the first row of the table to find the file Class1 on the “Person” string of the “Person class” to apply the fix “Move type to new file”, you should see the following in Visual Studio 2017:



Here are the remainder of the code fixes for this lab. See how many of them you can apply.

|  |  |  |  |
| --- | --- | --- | --- |
| File | Code block | Click on | Apply fix |
| Class1 | Person class | Person | Move type to new file |
| ToDispose class | ToDispose | Move type to new file |
| using block | using System | Remove unnecessary usings |
| Reorder usings |
| Namespace block | namespace | Add file header |
| Class1 declaration | Class1 | Order Class1’s members following StyleCop patterns |
| GetJim method | var Person | Use object initializer |
| Inline temporary variable |
| Convert to expression bodied member |
| GetJim | Replace “GetJim” with property |
| ShouldUseVar method – Throw new argumentException | “input” | use nameof() |
| Person person | Use var  (Note: The above action will cause a conflict with the stylecop rule. Disable the stylecop rule in the ruleset properties.) |
| His parent is {0} | Change to string interpolation |
| Catch | Remove wrapping Try Block |
| Person | \_name field declaration | \_name | Use auto property |
| CanVote method | If | Change to ternary |
| Convert to expression bodied member |
| Blog field declaration | “thinqlinq” | Manually change the text to <http://www.thinqlinq.com> (a great site if I could be so modest). Note here, there is no fix. This is just an analyzer error message because it detects a runtime exception for a domain specific issue. |
| IsPrime method | Where | Remove ‘Where’ moving predicate to ‘Any’ |
| IsFibber method | First “if” | Convert to switch |
| SayHello method | hello += | Use StringBuilder to create a value for ‘hello’ |

## On your own

Try working through the rest of the errors, warnings and info messages and see if you can either clean up all of the remaining issues, or suppress/disable those that you don’t feel are necessary for your application/agency.