|  |  |
| --- | --- |
| 1) | Find/make your "real" project folder, make sure you know what directory it is in. In this case, CircleProject is my "real" project. You can find the code that I am using in the UnitTestDemo project from the bitbucket repository. |
| 2) | In QTCreator, make a new Unit Test Project: |
| 3) | Make sure to create it in the same directory as your "real" project. My CircleProject folder is in the folder UnitTestDemo, so that is where I make my new project as well. Do not place it *inside* the real project folder.  Should end up with two folders in same location:  It will have code like this:     * catch.hpp runs our tests - you can take a peek if you want, but it is ugly… never modify it ☺ * mainTester.cpp is what provides the main function for our Test project. It has a few options you can modify to make it show passed tests, to write to a file, etc… See commented out code for details. You do not have to touch it if you don't want. * sampleTester.cpp : a file with a sample test. This is the file you will modify. |
| 4) | Add the .h and .cpp for the classes in your original project to the new project. We do NOT want to add main.cpp.  Make sure to navigate to your "real" project folder…  Then use Ctrl-Click to select the files we want to test:   They should appear listed in their own (real) folder: |
| 5) | Open the .pro file in the tester project: |
| 6) | We need to add instructions to the .pro file on where to find the .h file from your "real" project. Find this line:  *The .. says "go up one level". Adding that to the include path means that we should look for .h files (included files) there. This is a placeholder line the template adds.*  We need to change it to direct the compiler to where your "real" project's code is. Change it to something like:  Where **CircleProject** is the name of the folder for your "real" project.  **".."** says *"go up one directory level"*, so "../CircleProject/" says *"go up one folder and then look inside the CircleProject folder". Note the space between .. and ../CircleProject/; each location needs to be separated by a space.*  Make sure to save the .pro file. |
| 7) | Rename sampleTester.cpp to something like CircleTester.cpp (right click in QTCreator). This is the file your tests will go in. |
| 8) | Include your class's .h file in the tester file:  Try to compile - if you get an error about Circle.h (or whatever your .h is) not being found, go recheck step 6. |
| 9) | Add tests: Each Test should look like:  TEST\_CASE( "UniqueMachineName" ) {  //Do any setup we need to before we can test what we want  REQUIRE( something is true );  }  Replace the sample test case with this one to test the default constructor:  TEST\_CASE( "Circle/DefaultConstructor" ) {  Circle c1;  REQUIRE( c1.getX() == 0 );  REQUIRE( c1.getY() == 0 );  REQUIRE( c1.getRadius() == 2 );  }  Build and run, verify that it passes. You may notice a ton of warnings… comparing doubles with == is a bad idea (remember that they are always an approximation – two things that should be technically equal to each other may be off slightly). Fortunately, CATCH includes a tool to automate testing approximately equals. Rewrite the test to look like this:  And the warnings should go away.  Try adding a new test that exposes the bug in the single argument constructor. Make a new Circle with a radius of -5 and then REQUIRE that its radius is reported as 5. *(Our circle class should only ever use positive values for radius… this is done everywhere but that constructor).*  Run the test and make sure it fails. Then fix up the constructor so that the test passes (without breaking any other tests!). Note that you can do cout statements or use the debugger in unit tests just like any other code if you need to see why a particular test is failing.  You can my sample in UnitTestDemoCompleted folder for more test examples. |
| 10) | Some rules for test writing:   * Every class function that doesn't do Input/Ouput probably needs a test. * Each test tests only one function (with exception of having to use things like getX() to see if a constructor or other function worked). * A function may need to be tested more than once: certainly a function that returns a bool needs a test to see if it returns true when it should as well as a test to see if it returns false when it should. * A test can require more than one thing - I might try using the same function in multiple ways in one test and have a different REQUIRE for each of the variations. |
| 11) | Switch back to the main project. The **active project** in QTCreator is the one that will be compiled or run when you hit build or run. It is always shown in bold. To switch which project is Active right click a project and chose Set XXX as Active Project:    If CircleProject is active, it will build that project and run the regular main.cpp. If you switch back to the CircleTester project, that is the one that will be built and your tests will be run when you hit Run. Switch back and forth and make sure you can identify which project is active and what code is running. |