
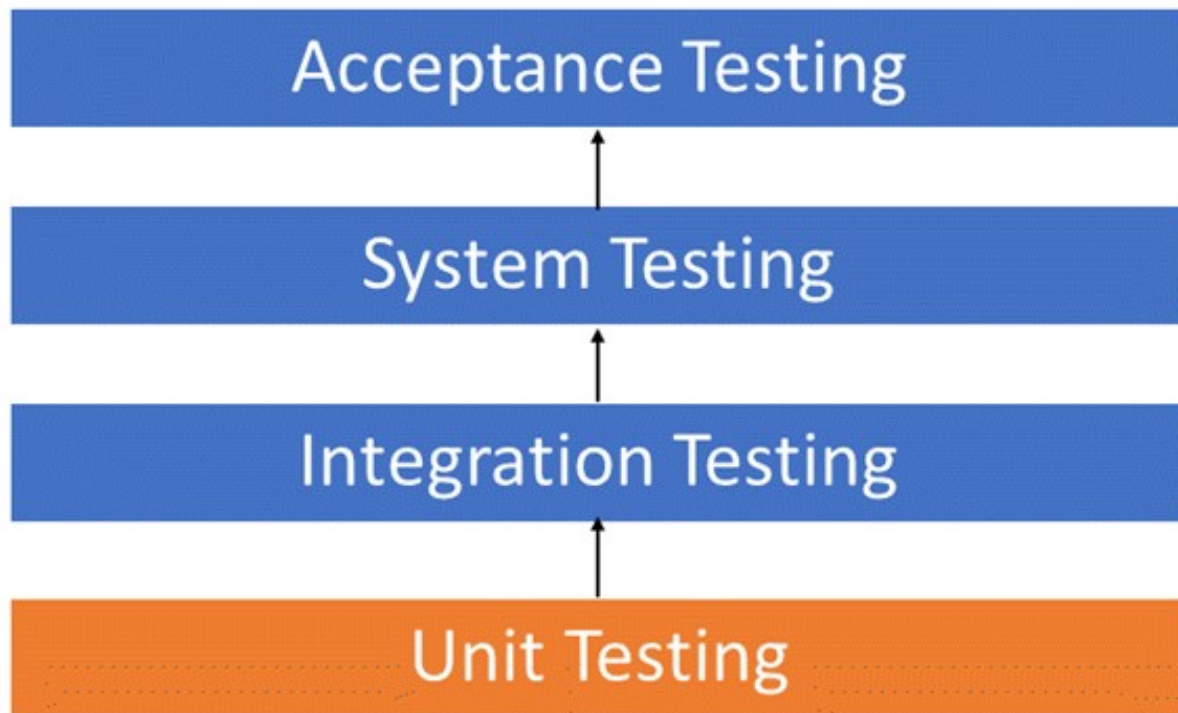


WHAT IS UNIT TESTING?

- UNIT TESTING is a type of software testing where individual units or components of a software are tested.
- The purpose is to validate that each unit of the software code performs as expected.
- Unit Testing is done during the development (coding phase) of an application by the developers.
- Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.
- Unit testing is a WhiteBox testing technique that is usually performed by the developer.

- 
- The background of the slide features several thin, light blue lines that intersect to form various geometric shapes, including triangles and quadrilaterals, creating a modern, abstract design.
- A unit is the smallest testable part of an application. It mainly has one or a few inputs and produces a single output. In procedural programming, a unit referred to as an individual program, while object-oriented programming languages include Base/Superclass, abstract class, Derived/Child class takes place.



*TYPES OF
TESTING*

WHY UNIT TESTING?

- Unit Testing is important because software developers sometimes try saving time doing minimal unit testing and this is myth because inappropriate unit testing leads to high cost Defect fixing during System Testing, Integration Testing. If proper unit testing is done in early development, then it saves time and money in the end.
 - Unit tests help to fix bugs early in the development cycle and save costs.
 - It helps the developers to understand the testing code base and enables them to make changes quickly
 - Good unit tests serve as project documentation
 - Unit tests help with code re-use. Migrate both your code and your tests to your new project. Tweak the code until the tests run again.

HOW DOES UNIT TESTING WORK?

- To do this, mocks are required. Is there a need for mocks to make testing on functions?
- Yes, without creating mocks functions cannot be unit tested. Testing works on the basis of mock objects. Mock objects work to fill in for missing parts of a program. For example, there might be a function that needs variables or objects that not created yet. To test function, mock objects created. In such conditions, mock objects fill missing parts. Techniques within Unit testing

TECHNIQUES WITHIN UNIT TESTING

- White-Box testing
 - It's referred to as a glass box testing/transparent testing. In this type of testing, the tester is aware of internal functionality. The internal structure of an item or function to be tested is unknown.
- Black-Box testing
 - It is a type of testing, tester not aware of the internal functionality of a system. The internal structure of the function to be tested is unknown.
- Gray-Box testing
 - It's referred to as semi-transparent testing. It is a combination of a Black Box and White Box testing. It is the type of testing in which tester aware with internal functionality of a method or unit but not in a more deep level like white box testing. In this, the user partially aware of the internal functionality of a system.

WHAT ARE THE BENEFITS OF UNIT TESTING?

- Unit Tests reveal a basic understanding of API units or functions to understand the functionality of the code. It is a way to write test cases for all functions and methods so that whenever a change causes a fault then in that case bug quickly identified and fixed. A modular approach followed in testing, single-single functionality or part of the code tested without waiting for other of code to be completed.

TOOLS FOR UNIT TESTING

- **Jtest**
- **NUnit**
- **JMockit**
- **EMMA**
- **GTest**

INTRODUCTION: WHY GOOGLETEST?

- googletest helps you write better C++ tests.
- googletest is a testing framework developed by the Testing Technology team with Google's specific requirements and constraints in mind. Whether you work on Linux, Windows, or a Mac, if you write C++ code, googletest can help you. And it supports any kind of tests, not just unit tests.

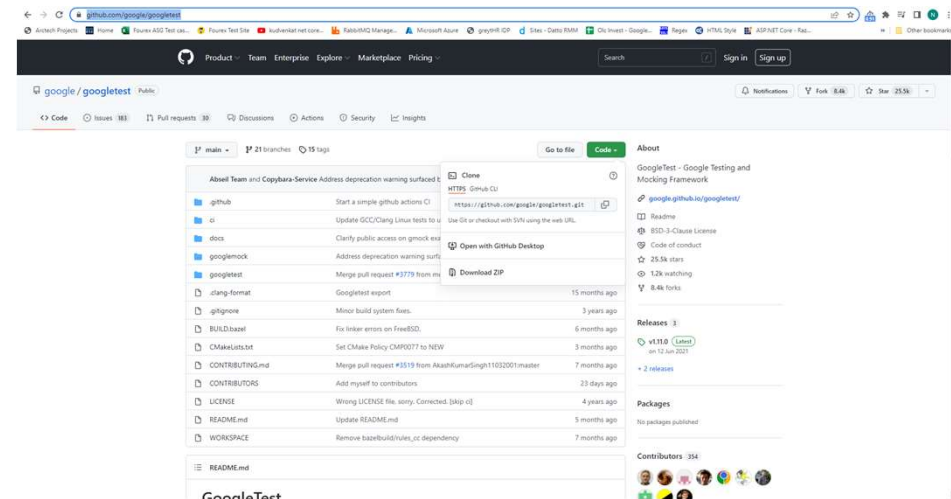
- So what makes a good test, and how does googletest fit in?
- **Tests should be independent and repeatable.** It's a pain to debug a test that succeeds or fails as a result of other tests. googletest isolates the tests by running each of them on a different object. When a test fails, googletest allows you to run it in isolation for quick debugging.
- **Tests should be well organized and reflect the structure of the tested code.** googletest groups related tests into test suites that can share data and subroutines. This common pattern is easy to recognize and makes tests easy to maintain. Such consistency is especially helpful when people switch projects and start to work on a new code base.
- **When tests fail, they should provide as much information about the problem as possible.** googletest doesn't stop at the first test failure. Instead, it only stops the current test and continues with the next. You can also set up tests that report non-fatal failures after which the current test continues. Thus, you can detect and fix multiple bugs in a single run-edit-compile cycle.

BASIC CONCEPTS

- When using googletest, you start by writing assertions, which are statements that check whether a condition is true. An assertion's result can be success, nonfatal failure, or fatal failure. If a fatal failure occurs, it aborts the current function; otherwise the program continues normally.
- Tests use assertions to verify the tested code's behavior. If a test crashes or has a failed assertion, then it fails; otherwise it succeeds.
- A test suite contains one or many tests. You should group your tests into test suites that reflect the structure of the tested code. When multiple tests in a test suite need to share common objects and subroutines, you can put them into a test fixture class.
- A test program can contain multiple test suites.

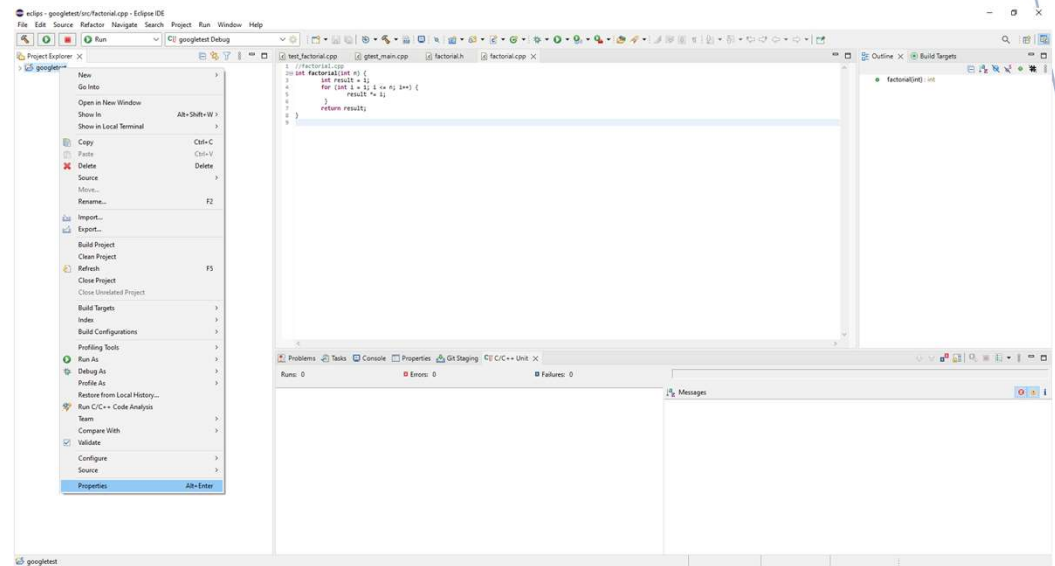
HOW TO INSTALL GTEST IN ECLIPSE

Go to the url
<https://github.com/google/googletest>
and download google test library on
your computer.



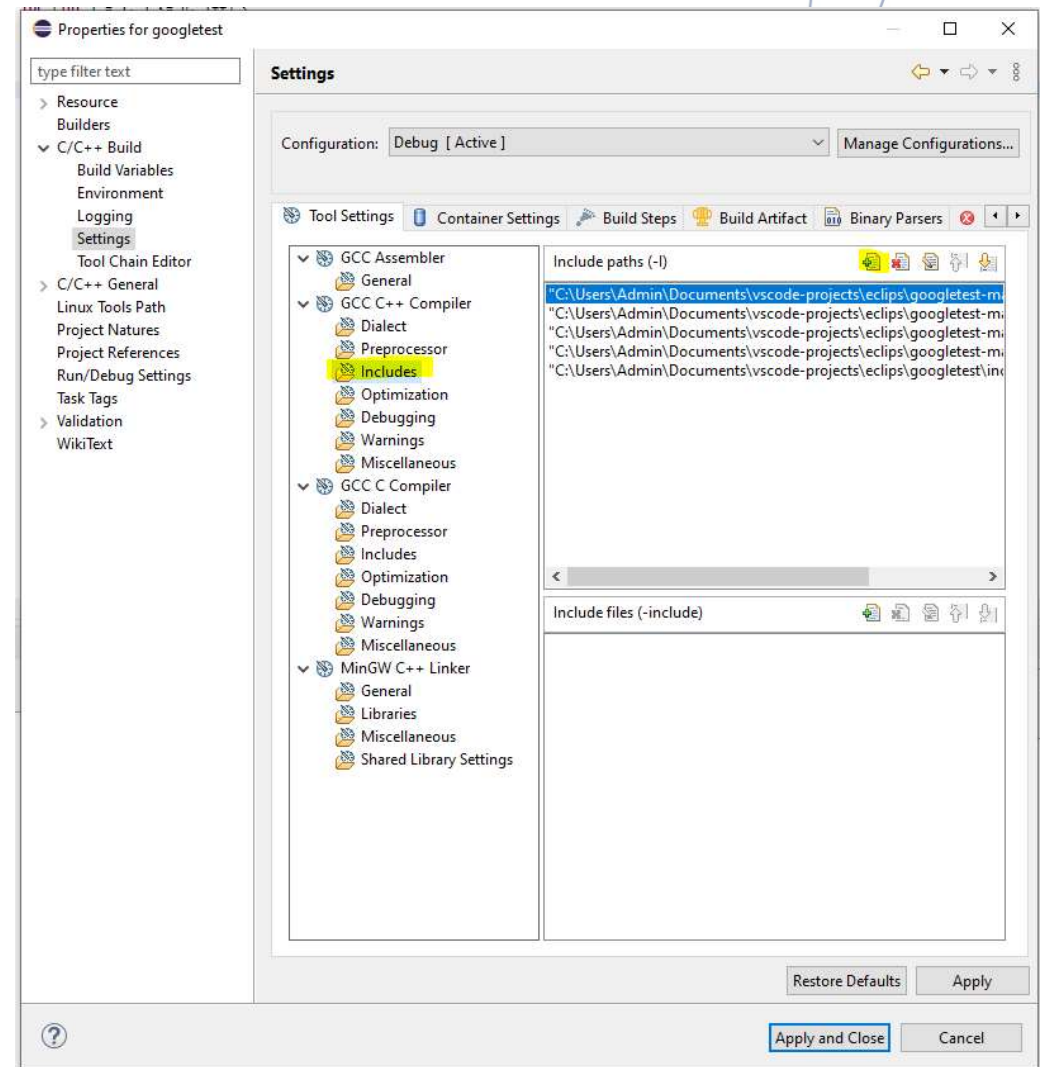
HOW TO INSTALL GTEST IN ECLIPSE

- After downloading googletest, open eclipse application and create new project in your workspace.
- Right click on project and click on properties option



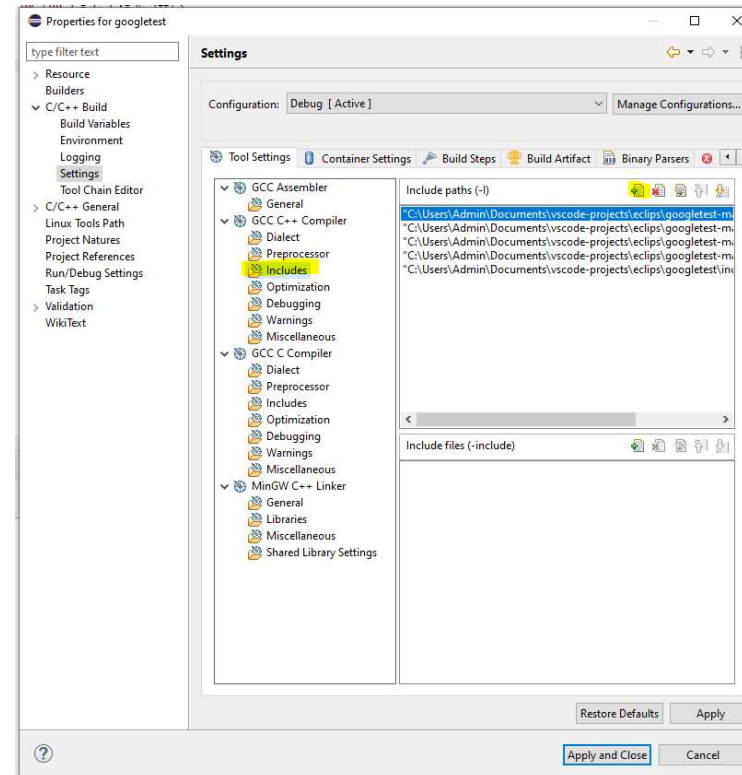
HOW TO INSTALL GTEST IN ECLIPSE

Click on C/C++ Build option from left menu and select settings option.



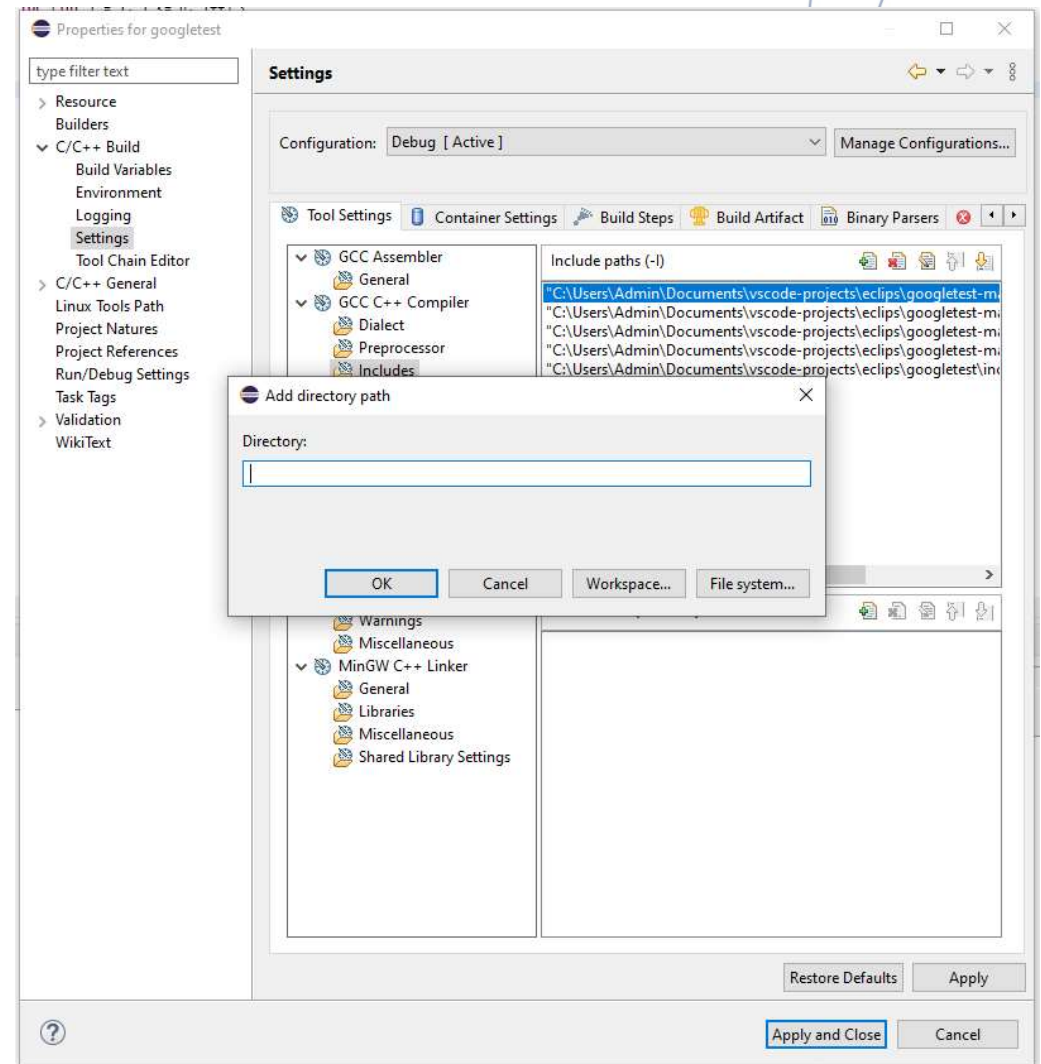
HOW TO INSTALL GTEST IN ECLIPSE

- Click on Includes from GCC C++ Compiler option



HOW TO INSTALL GTEST IN ECLIPSE

- Click on add icon from right pane (Include paths(-I)) and add the gtest include path in Directory and click on OK
- Please add below four Gtest paths in Include paths one by one
 - C:\Users\Admin\Documents\vscode-projects\eclips\googletest-main\googletest
 - C:\Users\Admin\Documents\vscode-projects\eclips\googletest-main\googletest\include
 - C:\Users\Admin\Documents\vscode-projects\eclips\googletest-main\googlemock
 - C:\Users\Admin\Documents\vscode-projects\eclips\googletest-main\googlemock\include

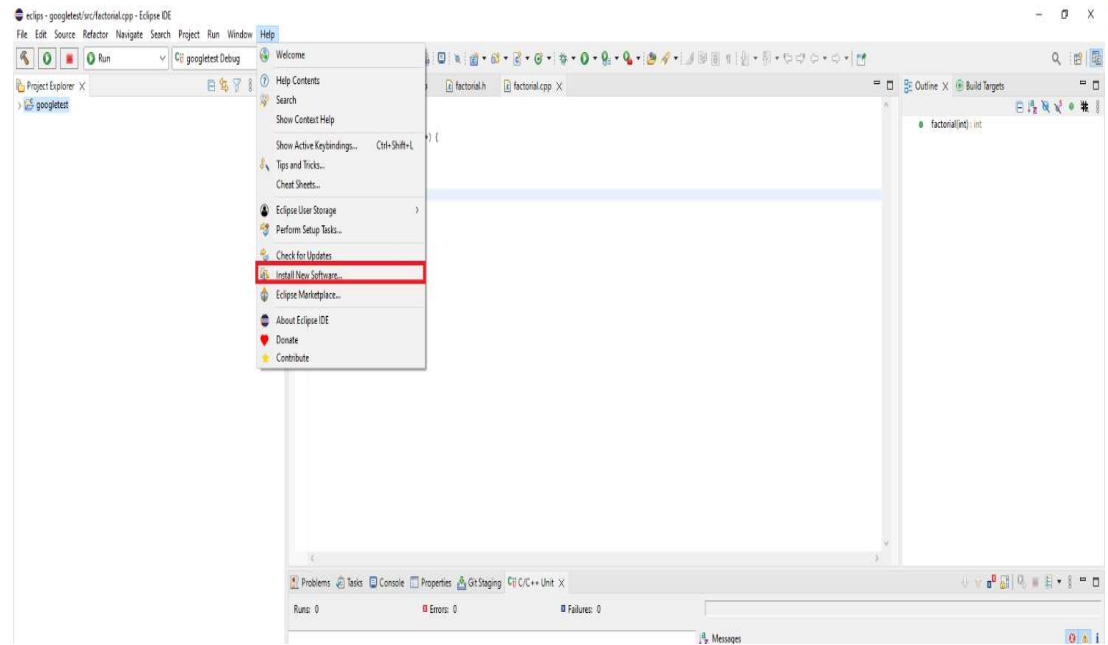


HOW TO INSTALL GTEST IN ECLIPSE

- After added above four paths click on Apply or Apply and Close button in window.
- After these steps is done you can run the application directly with your normal steps
- Once this done you can also download additional test software in eclipse using below steps

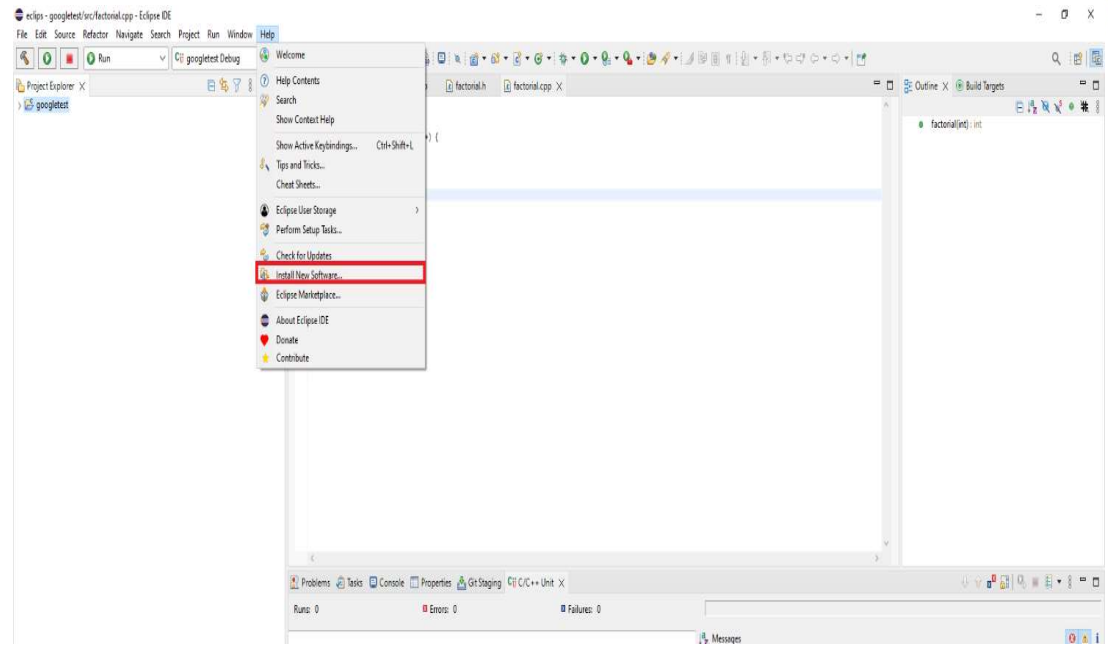
HOW TO INSTALL GTEST IN ECLIPSE

- Go to the Help menu and click on “Install new software”



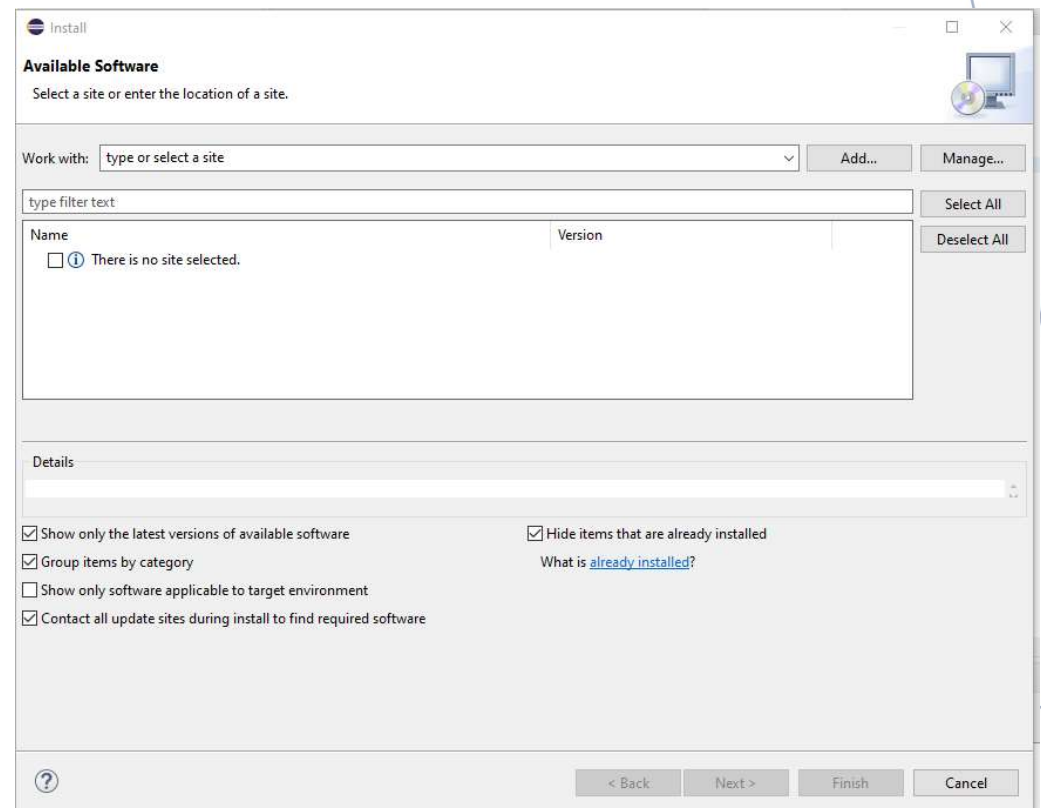
HOW TO INSTALL GTEST IN ECLIPSE

- Go to the Help menu and click on “Install new software”



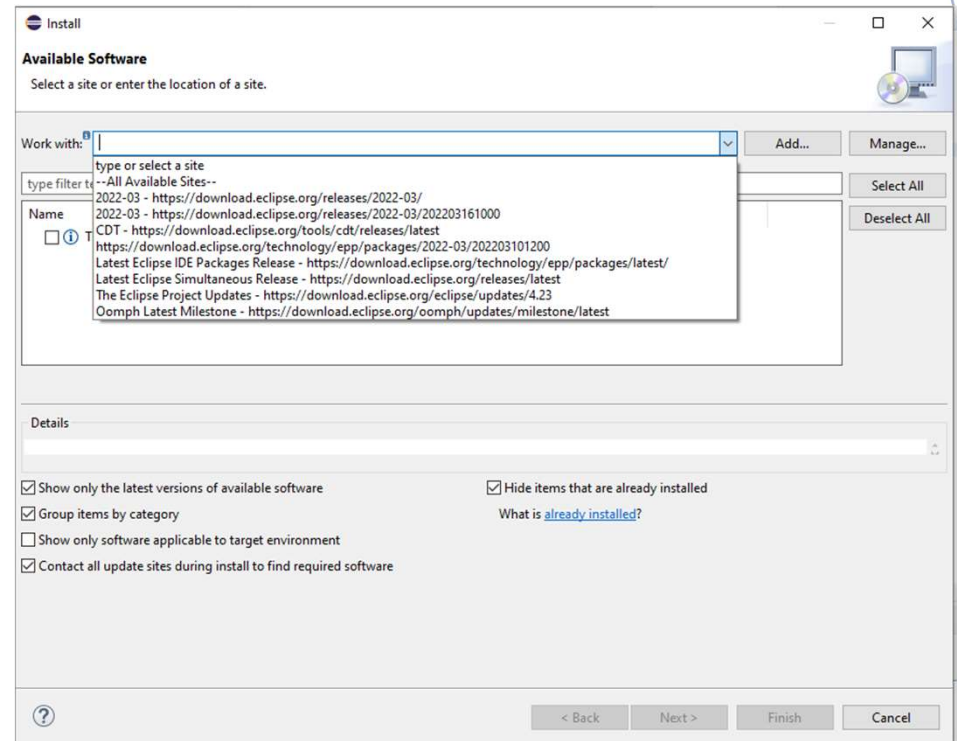
HOW TO INSTALL GTEST IN ECLIPSE

- After clicking install new software below window will be open on the screen



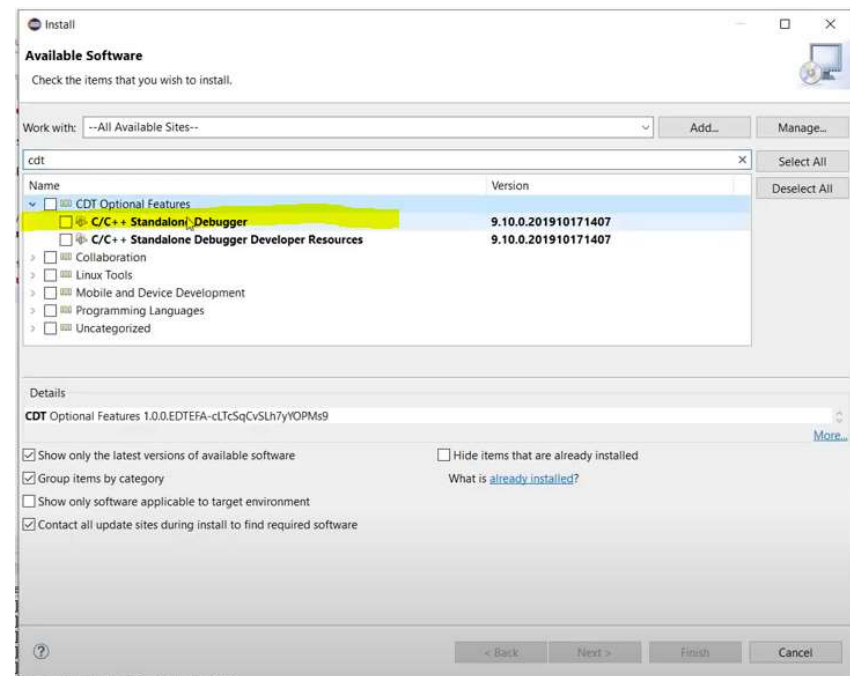
HOW TO INSTALL GTEST IN ECLIPSE

- From Work with dropdown select "-- All Available Sites --" and type filter text search text "cdt"



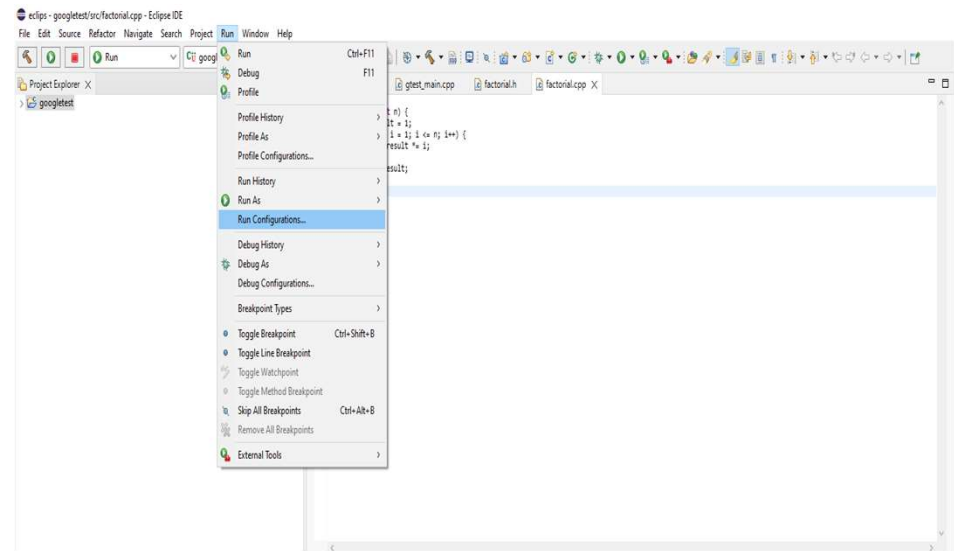
HOW TO INSTALL GTEST IN ECLIPSE

- Install the software from CDT optional Features by click Next button and close the window after installation.



HOW TO INSTALL GTEST IN ECLIPSE

- After installation done click on Run menu and select Run Configuration



HOW TO INSTALL GTEST IN ECLIPSE

- In run configuration window double click on C/C++ Unit from left menu and select C/C++ Testing tab. Then select "Google Test Runner" from Test Runner dropdown as show in below image and click on Apply button and close the window.

