

Installation

Option 1: Install from Eclipse Marketplace (Easier, straightforward)

1. Help -> Eclipse Marketplace
2. Type "EclEmma" in the search box
3. If you didn't install before, then press install button to install EclEmma, and restart Eclipse afterward. Else, you are all set and ready for testing.

Option 2: Manually install

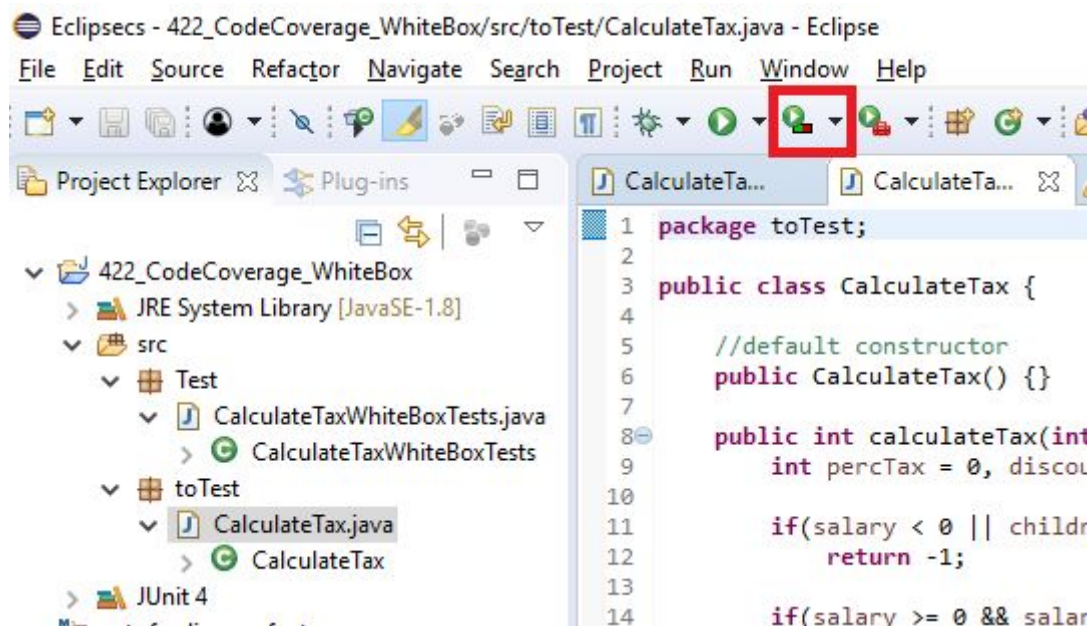
1. Download a version of EclEmma zip file from the following website:
<http://www.eclEmma.org/download.html>
2. Unzip the archive into dropins folder of your Eclipse installation, then restart Eclipse

User Guide

After one of the above installation, you should be able to see the EclEmma coverage button in the toolbar as following:



The most important button out of the toolbar is the one mark in the red box as below:




Example

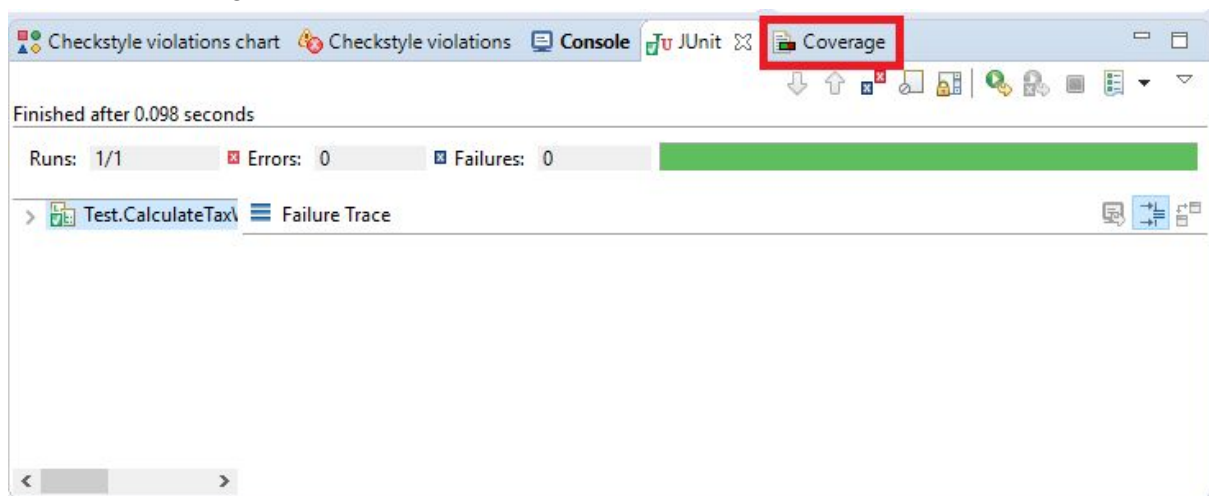
Here is an example class CalculateTax. Inside the class there is a function/method having salary and children pass in as parameters.

```
1 package toTest;
2
3 public class CalculateTax {
4
5     //default constructor
6     public CalculateTax() {}
7
8     public int calculateTax(int salary, int children) {
9         int percTax = 0, discount = 0;
10
11         if(salary < 0 || children < 0)
12             return -1;
13
14         if(salary >= 0 && salary <= 15000)
15             percTax = 0;
16         else if(salary > 15000 && salary <= 50000)
17             percTax = 25;
18         else if(salary > 50000)
19             percTax = 45;
20
21         if (children == 0)
22             discount = 0;
23         else if(children == 1)
24             discount = 2;
25         else if(children == 2)
26             discount = 5;
27         else if(children == 3)
28             discount = 7;
29         else if (children >= 4)
30             discount = 10;
31
32         if(salary > 80000)
33             discount = 0;
34
35         if(percTax - discount < 0)
36             return 0;
37         return (percTax - discount);
38     }
39 }
```

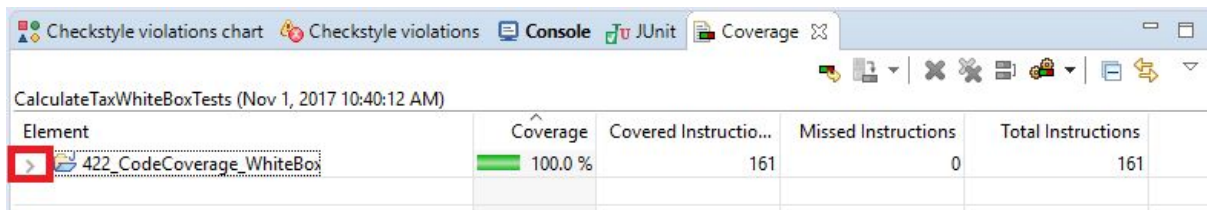
And the following is the test cases

```
9 public class CalculateTaxWhiteBoxTests {
10
11     @Test
12     public void testCalculateTax() {
13         System.out.println("inside testCalculateTax");
14
15         CalculateTax app = new CalculateTax();
16
17         // no children case green
18         assertEquals("salary 40000, children 0", app.calculateTax(40000, 0), 25);
19
20         // makes the > 80000 branch green
21         assertEquals("salary > 80000", app.calculateTax(80001, 0), 45);
22
23         // nodes 1-9 green
24         assertEquals("salary < 0", app.calculateTax(-1, 0), -1);
25         assertEquals("children < 0", app.calculateTax(40000, -1), -1);
26
27         // nodes 2-12 green
28         assertEquals("salary 15000, children 1", app.calculateTax(15000, 1), 0);
29         assertEquals("salary 30000, children 2", app.calculateTax(30000, 2), 20);
30         assertEquals("salary 40000, children 3", app.calculateTax(40000, 3), 18);
31         assertEquals("salary 60000, children 4", app.calculateTax(60000, 4), 35);
32         assertEquals("salary 60000, children 5", app.calculateTax(60000, 5), 35);
33     }
34 }
```

While you have your files open up, just simply press the button  to test for the coverage. After finish loading, an additional block should appear on the bottom of Eclipse like below:

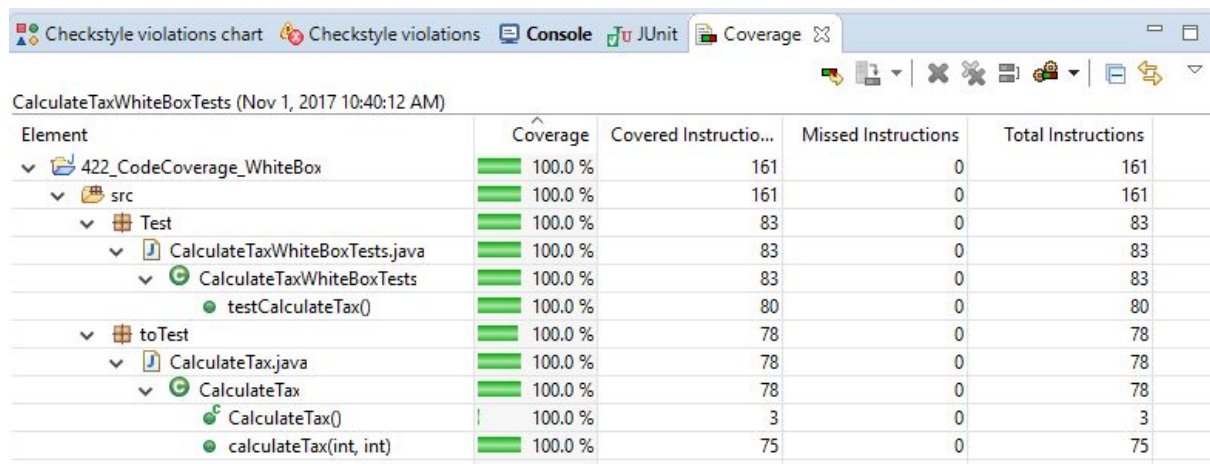


Tap on "Coverage". The Coverage column is the total coverage out of the entire package, and if you want to check on each individual class/function/method's coverage, click on the arrow mark as below, until you get to the part you want to review:



Element	Coverage	Covered Instruction...	Missed Instructions	Total Instructions
> 422_CodeCoverage_WhiteBox	100.0 %	161	0	161

Now you know all the coverage over the entire package with Eclemma.



Element	Coverage	Covered Instruction...	Missed Instructions	Total Instructions
422_CodeCoverage_WhiteBox	100.0 %	161	0	161
src	100.0 %	161	0	161
Test	100.0 %	83	0	83
CalculateTaxWhiteBoxTests.java	100.0 %	83	0	83
CalculateTaxWhiteBoxTests	100.0 %	83	0	83
testCalculateTax()	100.0 %	80	0	80
toTest	100.0 %	78	0	78
CalculateTax.java	100.0 %	78	0	78
CalculateTax	100.0 %	78	0	78
CalculateTax()	100.0 %	3	0	3
calculateTax(int, int)	100.0 %	75	0	75

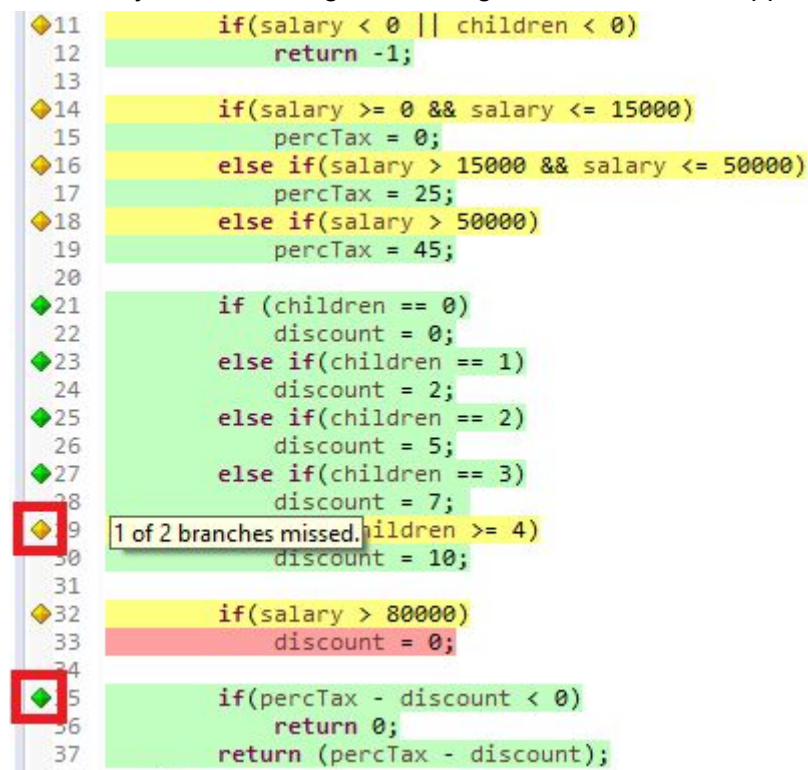
Q: What to do if the coverage is not 100%?

There are more information can be helpful on the java page itself. Now when you get to the java file page, you realize the lines are either red/ yellow/ green. The meaning of each coloring as below:

- red** - The line is not coverage base on the test you wrote
- yellow** - There are path you didn't not cover in you test
- green** - This path is being cover

When your mouse hover on the **green** diamond shape, it will notice you all the branches are cover and the line will appear in green.

When your mouse hover on the **yellow** diamond shape, it will notice you have how many branches you are missing on coverage, and the line will appear in yellow.



```

11  if(salary < 0 || children < 0)
12      return -1;
13
14  if(salary >= 0 && salary <= 15000)
15      percTax = 0;
16  else if(salary > 15000 && salary <= 50000)
17      percTax = 25;
18  else if(salary > 50000)
19      percTax = 45;
20
21  if (children == 0)
22      discount = 0;
23  else if(children == 1)
24      discount = 2;
25  else if(children == 2)
26      discount = 5;
27  else if(children == 3)
28      discount = 7;
29  1 of 2 branches missed: children >= 4
30      discount = 10;
31
32  if(salary > 80000)
33      discount = 0;
34
35  if(percTax - discount < 0)
36      return 0;
37      return (percTax - discount);

```

Tips

- Normally if statement can only cover 1 branch, for complete with all branch you will need extra else if statement and else statement to make the yellow lines turn into green.
- Fix and try to hit all cases. Fix it all one case by one case. Be patient and at the end you will be able to get 100% coverage.
- Power mock and EclEmma do not goes well together. Here is an discussion on github about this issue: <https://github.com/powermock/powermock/issues/422>