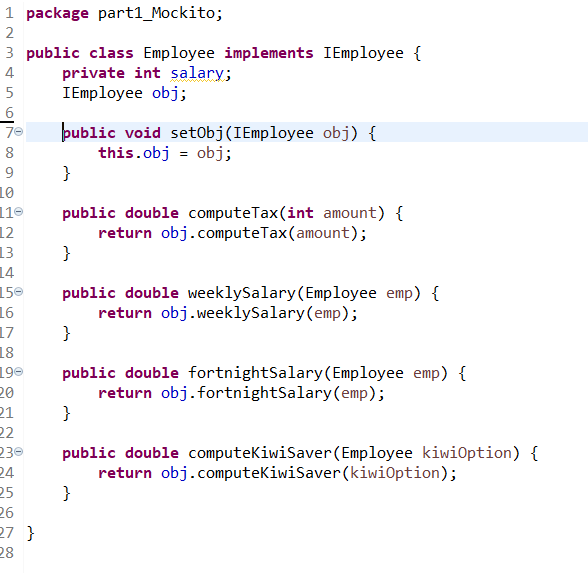
# JUnit – Mockito

The exercise starts with the creation of the class Employee and interface IEmployee, and the appropriate implementation of the interface by the use of the keyword ‘implements’.

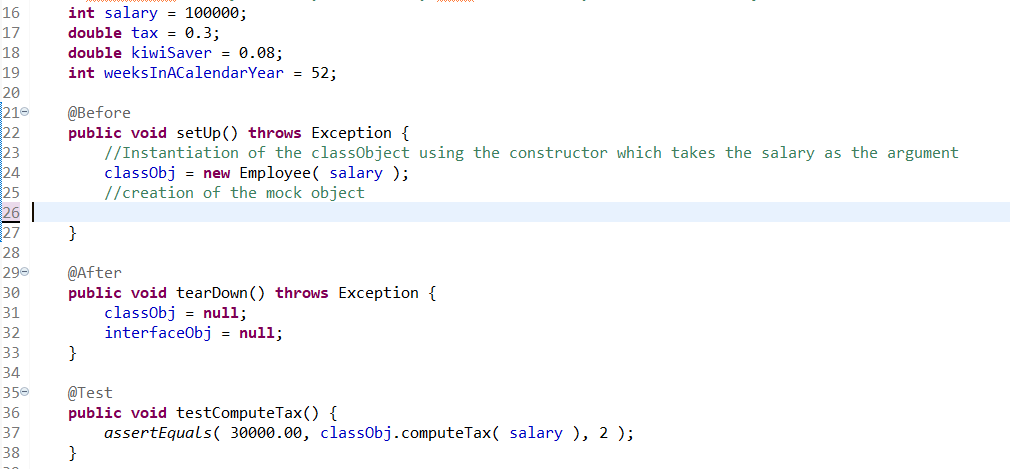
Unimplemented methods were added, an object of IEmployee created which was used to call its own methods:



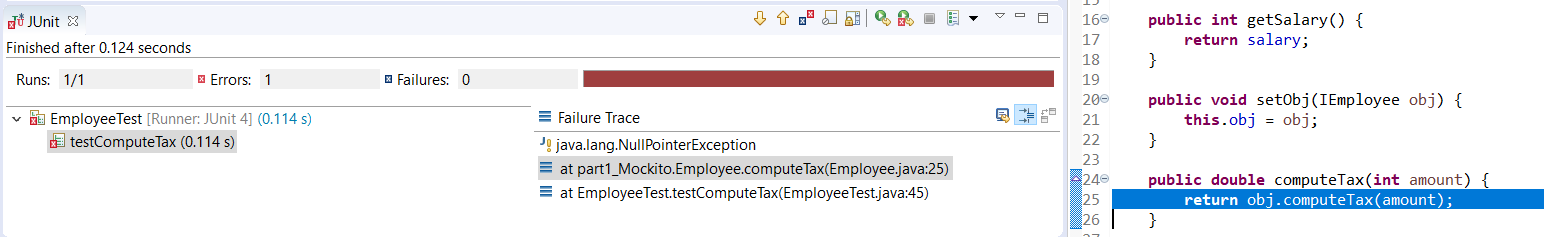
A new JUnit test case for class Employee was then created. Objects of class Employee and interface IEmployee were created, and assumed test variables were initialised: salary of type int as 100.000, tax of type double as 0.3, kiwiSaver of type double as 0.8, weeksInACalendarYear of type int as 52.

In the setUp method the class object is instantiated using variable salary as the parameter for the constructor.

A test is created where the expected value of compute tax is 30.000 based on our assumptions: salary = 100000, and tax = 30%.



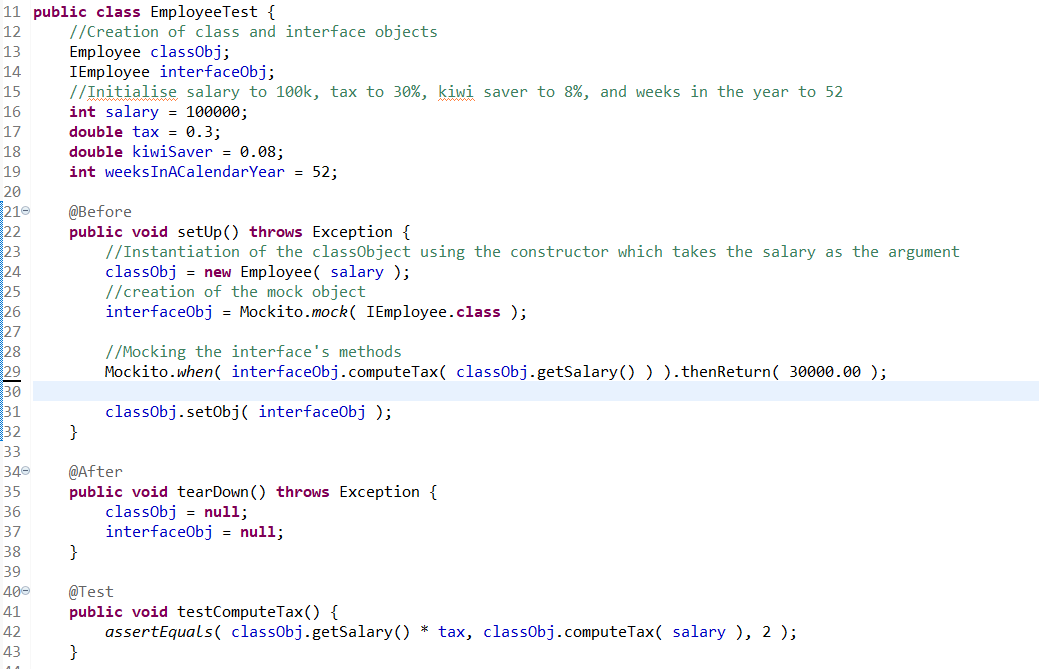
Running this test results in a NullPointerException, given that the method called executes the method inside the interface IEmployee which is a skeleton code.

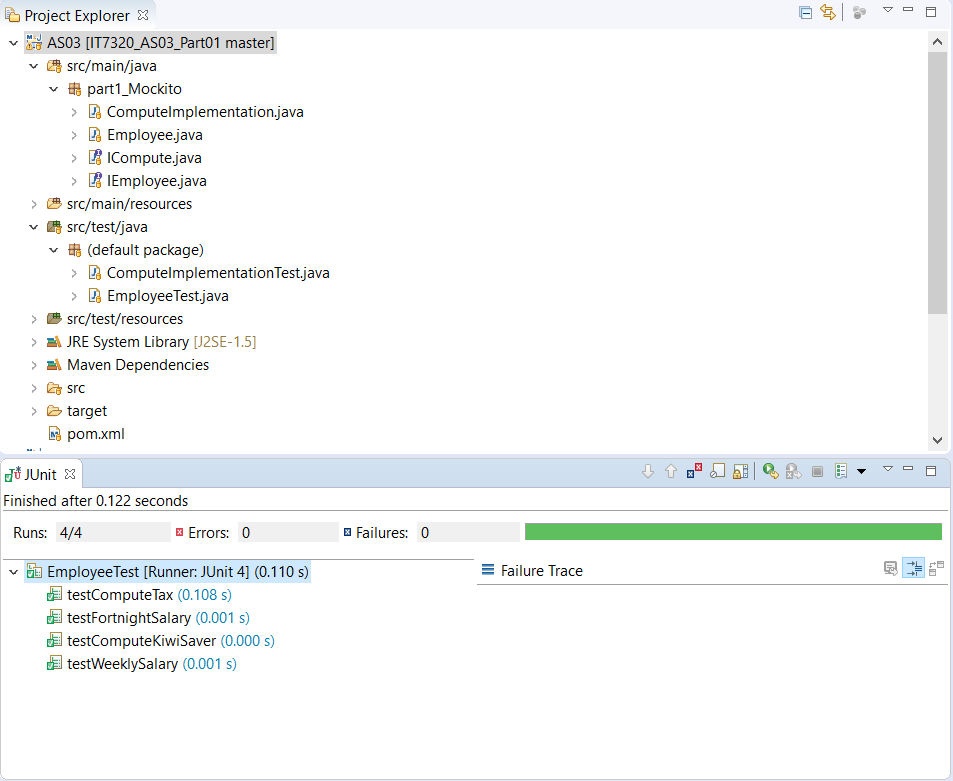


## Why Mock?

“The purpose of mocking types is to sever dependencies in order to isolate the test to a specific unit.” [1]

An interface was selected because it gives enough a degree of abstraction that is necessary when mocking. Classes are concrete and can be tested. What an interface does and how it works is often considered beyond the knowledge of a developer.





[1] <https://stackoverflow.com/questions/2665812/what-is-mocking> retrieved on 19/09/18 at 3:51PM