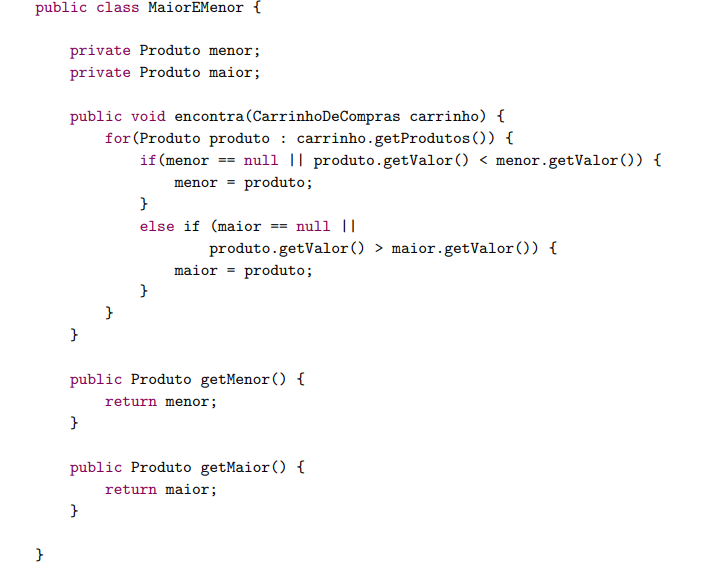
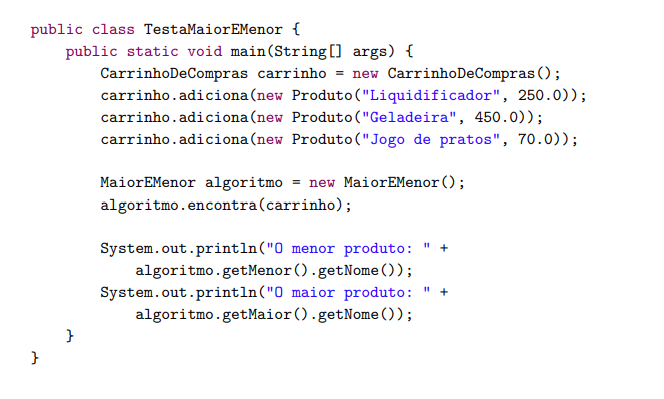
**TDD**

**Testes Unitários:**

Um teste de unidade testa uma única unidade do nosso sistema. Geralmente, em sistemas orientados a objetos, essa unidade é a classe. Em nosso sistema de exemplo, muito provavelmente existem classes como “CarrinhoDeCompras”, “Pedido”, e assim por diante. A ideia é termos baterias de testes de unidade separadas para cada uma dessas classes; cada bateria preocupada apenas com a sua classe.





Veja bem essa é uma implementação simples de um teste unitário, a partir dessa classe de Teste podemos mapear as funcionalidades e verificar erros de forma mais rápida e eficiente.

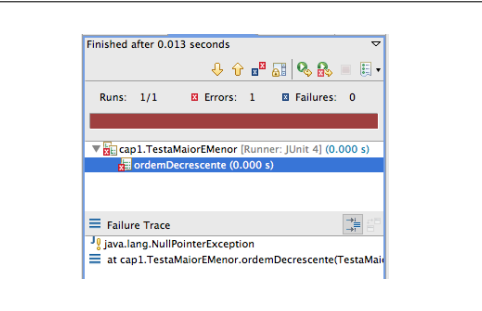
**Testes Automatizados:**

Veja bem se analisarmos a classe de teste acima percebemos que mesmo que ela automatize algumas tarefas e teste a unidade ainda é necessário que a intervenção de um ser Humano para analisar as saídas e registrar os erros, nesse sentido é importante também adicionar automatização nesse registro, isso pode ser feito através de códigos que façam isso ou podemos utilizar de **frameworks** de automatização de testes, no Java o mais conhecido é o jUnit.

O JUnit possui um plugin para Eclipse, que mostra uma lista de todos os testes executados, pintando-os de verde em caso de sucesso, ou vermelho em caso de falha. Ao clicar em um teste vermelho, a ferramenta ainda apresenta a linha que falhou, o resultado que era esperado e o resultado devolvido pelo método.

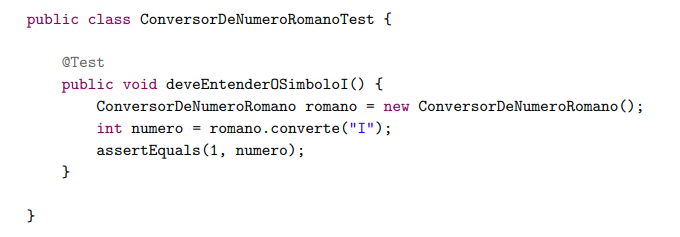
Para converter uma classe em um teste jUnit devemos declarar ela como uma classe de teste com a anotação **@Test** e devemos declarar quais são os resultados esperados através do método **Assert.assertEquals();**

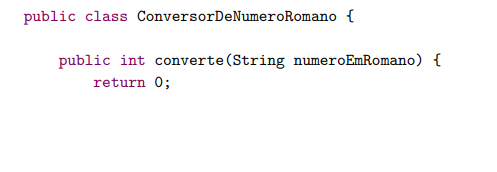


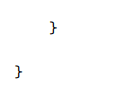


**Test-Driven Development:**

Quando falamos de TDD estamos falando em testes feitos anteriormente a criação da própria classe, de forma simplificada pensamos primeiro no resultado esperado e montamos a classe de teste, em seguida montamos a classe a ser testado com o mínimo de recursos possíveis para o teste ter sucesso:







Perceba um Detalhe aqui, utilizamos diretamente o asserEquals graças ao import estático da classe: import static org.junit.Assert.\*

Após montarmos a estrutura básica, cabe ao desenvolvedor desenvolver as duas classes de forma simultânea.

De maneira mais abstrata, o ciclo que foi repetido ao longo do processo de desenvolvimento da classe acima foi:

• Escrevemos um teste de unidade para uma nova funcionalidade;

• Vimos o teste falhar;

• Implementamos o código mais simples para resolver o problema;

• Vimos o teste passar;

• Melhoramos (refatoramos) nosso código quando necessário

