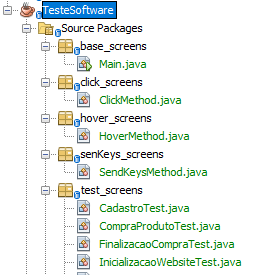
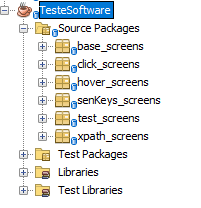
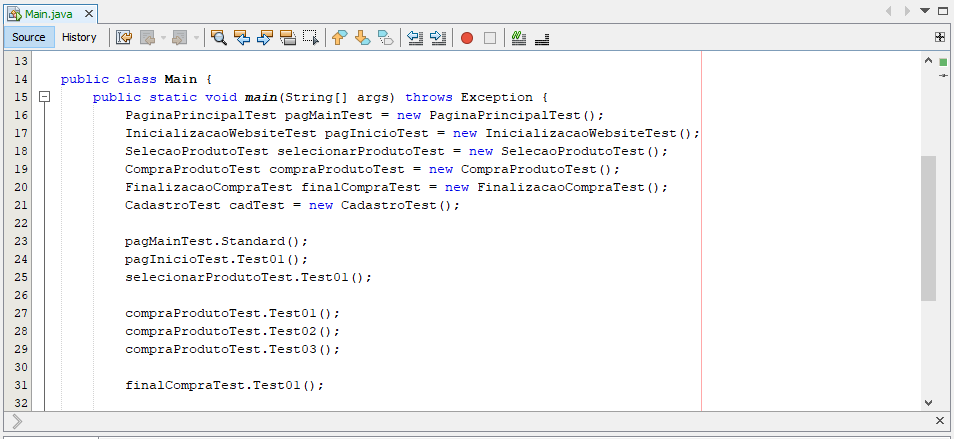
**Testes Apresentados Na Reunião**

1. Projeto: **Automação de um site de compras**

****

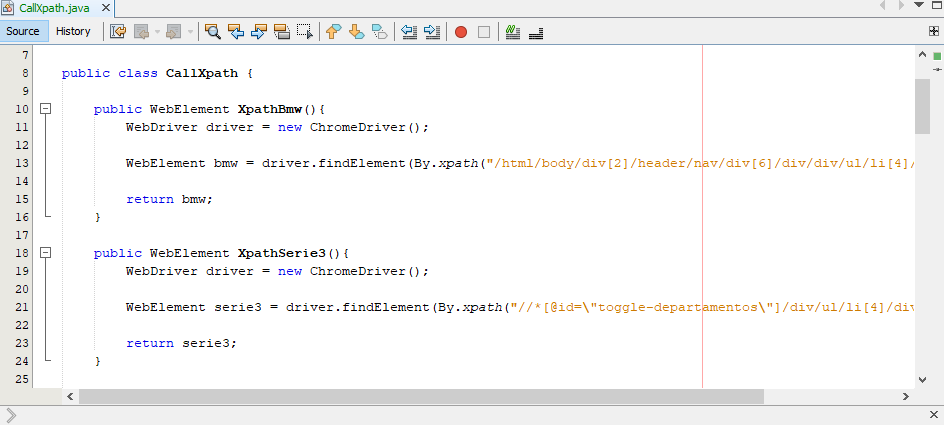
**Padrão *Page Objects.***

Nesse primeiro projeto os testes são separados pelo padrão de *design Page Objects* utilizado muito em testes *selenium*. Nesse caso o projeto é separado por ações como: a *main* onde é chamado todo o processo do projeto, instancias e métodos/testes.



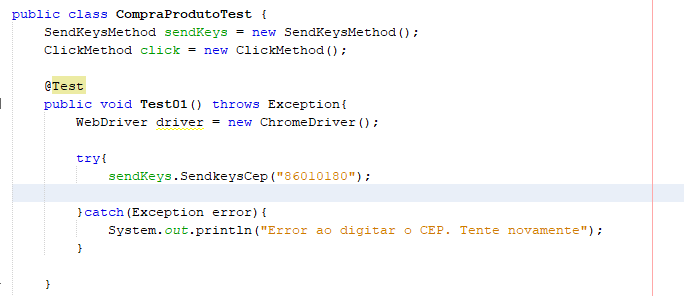
***Main* do Projeto.**

Para esse projeto foi utilizado a estrutura *selenium chrome webdriver* no qual, como mostrado na imagem abaixo, é feito pelo padrão *page objects*. Feito em um único método as tratativas do comando *xpath*, como também no projeto é feito as tratativas de clicar no botão, passar o *mouse* em um determinado campo, isso tudo separado em *packages* como é utilizado pelo *page objects*.

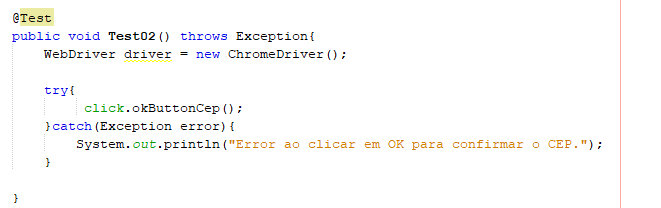
****

**Classe onde são guardados os comandos pelo caminho Xpath.**

E por fim, alguns testes que foram feitos conforme o exigido/necessitado. Como mostrado abaixado, como a primeira imagem mostrando um teste utilizando *o try-catch*, onde é testado via *sendkeys* se o cep inserido é o correto, se sim o teste passa caso o contrário o teste é falho.

****

**Teste de Compra de Produto. Teste 1 – Verificar se o CEP inserido existe.**

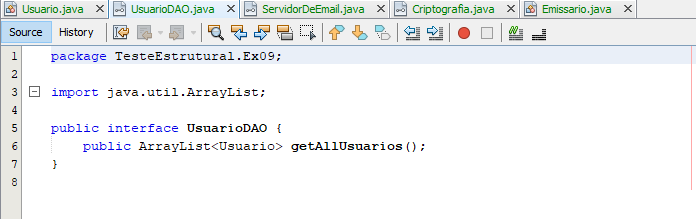
****

**Teste de Compra de Produto. Teste 2 – Verificar se o botão CEP é acionado.**

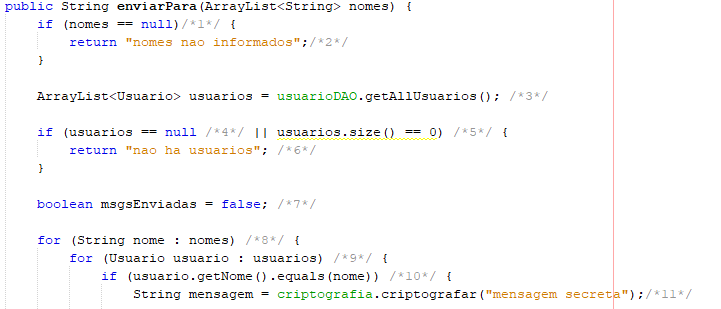
**Obs.:** Esse projeto não tem validação de *inputs*, ou seja, nos testes apresentados todos deveriam passar, já que não foi estipulado/concordado que dados prévios seriam utilizados para a verificação de dados que fossem inseridos nos testes.

1. Projeto: **Caixa Branca**

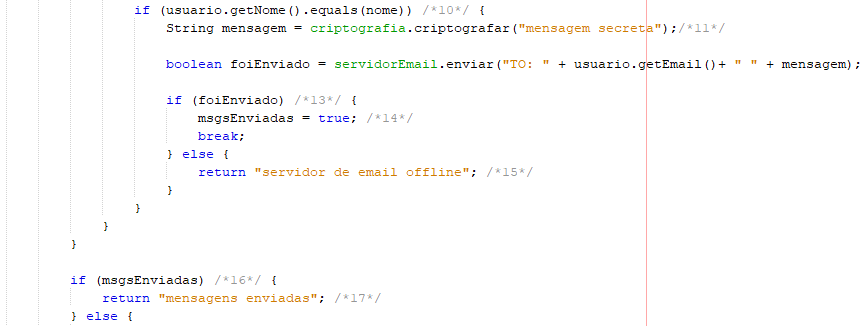
Nesse projeto é apresentado um código para envio de mensagem para um usuário em uma lista de *array*.

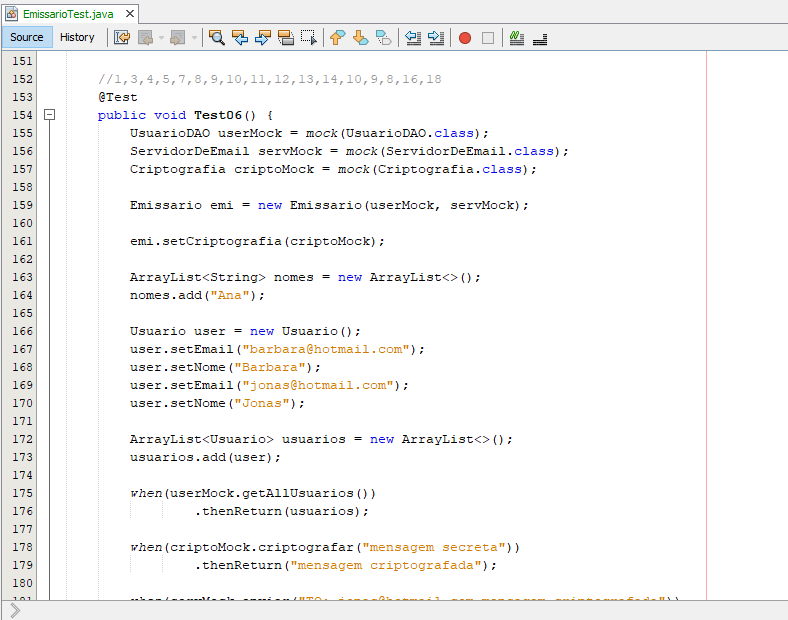


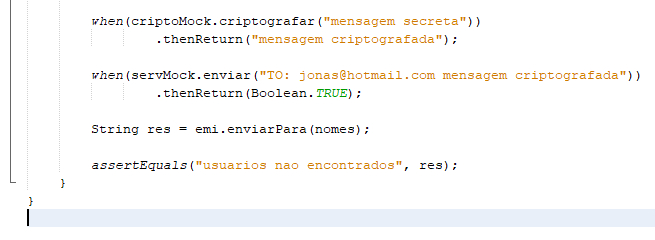
**Nesse projeto há três classes de interface junto a classe de *gets/sets***

****

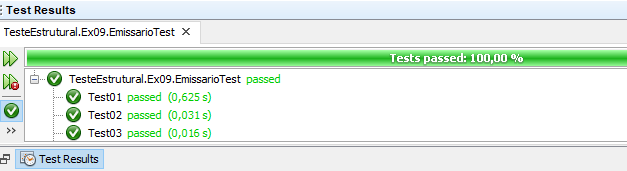
Classe de “Emissario” no qual a classe é criada para enviar uma mensagem de acordo com os usuários inseridos no *array*.

****



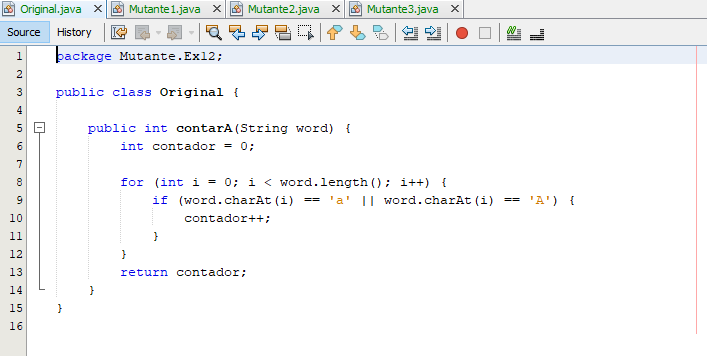


Como apresentado nas imagens do teste acima, é utilizado o *mockito/mock* para estruturar as interfaces apresentadas. Após isso, é adicionado um nome na lista de *array* no qual será enviado a mensagem para. É instanciado a classe “Usuário” no qual é adicionado no *array list* de usuário o nome e email do remetente a ser enviado a mensagem secreta. Após isso, a mensagem secreta é criptografada para ser enviada e por fim, é *mockado* o email da pessoa que vai enviar a mensagem, retornando um *boolean* verificando se o email está no *array* de usuários.

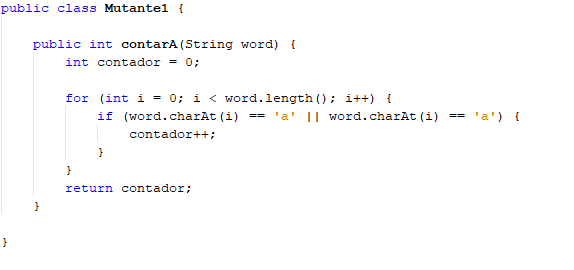


Como mostrado pela cobertura utilizando o *jacoco*, todos os testes passam. Sendo, feitos realmente para passarem com seus *inputs* inseridos para passarem em todos os testes gerados.

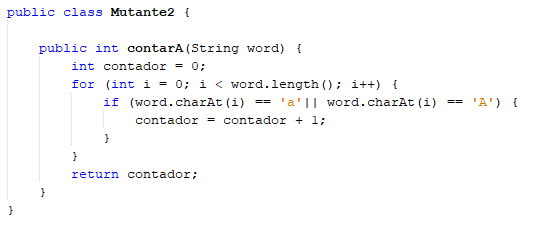
1. Projeto: **Mutante**



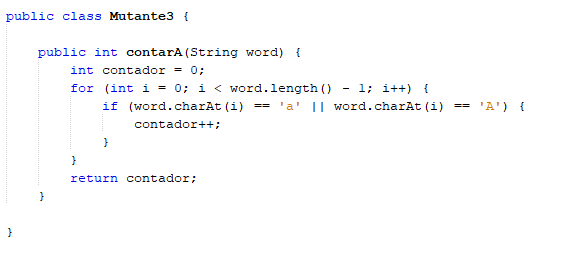
Nesse caso temos o código original e três mutações do mesmo. O código se baseia em contar as letras “a” minúsculo e “A” maiúsculo em uma determinada *string*.



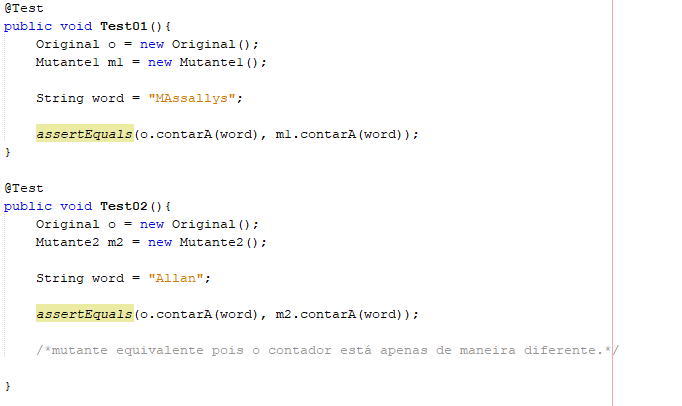
**Mutante 1** – São aceitos nesse mutante só letra “a” minúsculo.

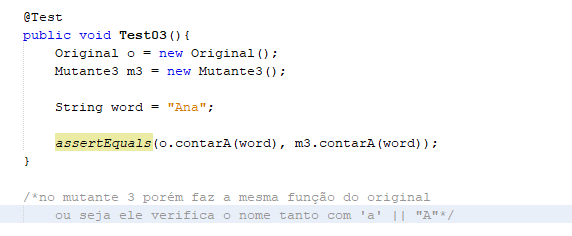


**Mutante 2** – São aceitos nesse mutante letras “a” minúsculo e “A” maiúsculo.



**Mutante 3** – A mesma situação do **Mutante 2** com o diferencial que o incremento da letra está começando com -1.



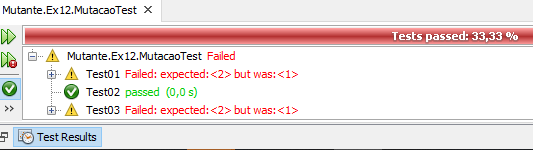


Nos testes dos mutantes é verificado se as inserções de *string* podem ser validas tanto para o mutante quanto para o código original.

**Teste 1** – Nesse teste é comparado o **mutante 1** e o código principal, o teste nesse caso não irá passar pois as comparações das letras no código original e no **mutante 1** são diferentes.

**Teste 2** – Já nesse caso o teste 2 irá passar, pois as comparações são iguais, sendo assim, o código **mutante 2** não é uma mutação.

**Teste 3** – No caso do código 3 ele é considerado uma mutação, pois ao contar o tamanho da letra, o mesmo começa a partir da posição -1, o que não é apresentado no código original que começa na posição 0.



**Como mostrado pelo *jacoco*, o único teste que passa é o “Teste02”.**