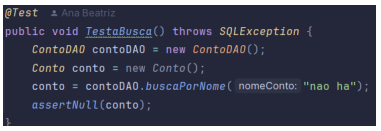
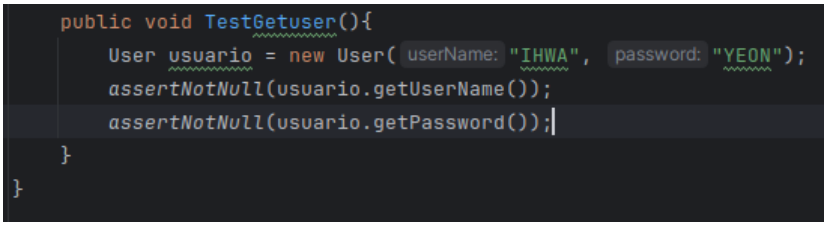
Trabalho 3 POO

Giovana Fernandes Nascimento, Ana Beatriz Sureke Buese Henrique e Diego Camargo

Testes Ana Beatriz:

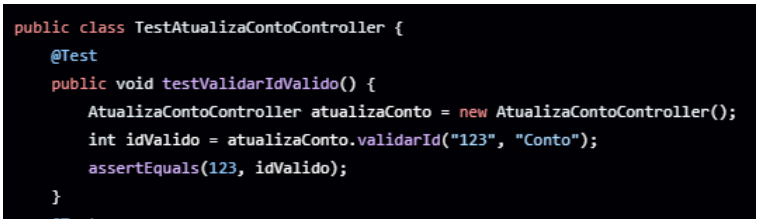
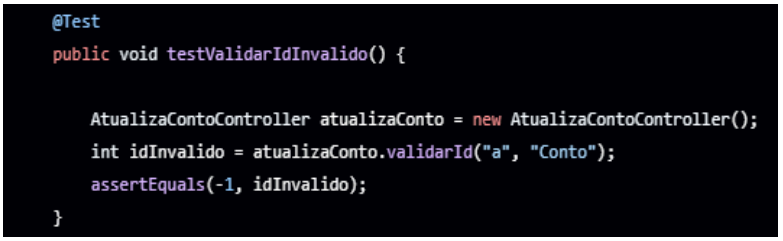
ID do Teste	Objetivo do Teste	Código
1	Verifica se a inserção está funcionando correta da seguinte forma: insere um conto e um herói, e então, procura o herói relacionado ao conto. O objetivo é verificar se este herói não está vazio.	
2	Verifica se o resultado de uma busca por um conto não existente possui resultado nulo, como é esperado.	
3	Verifica se um id de um herói mais antigo é menor que o id de um herói mais novo	
4	Verifica se a atualização está funcionando corretamente, verificando se um id continua o mesmo após atualização.	

Testes Diego:

ID do Teste	Objetivo do Teste	Código
1	Testa a criação do usuário passando como parâmetro o “userName” e o “password” verificando se realmente foi criado.	
2	Testa a alteração do nome e da senha de um usuário já criado e com nome e senha anteriormente definidos	
3	Testa a inserção de um userName no banco de dados, depois consulta o banco buscando o nome inserido e atribui esse nome a um novo usuário o qual deve ser o mesmo que foi inserido	
4	verificar se inicialmente o Session Cookie é nulo, depois ver se está recebendo o userName	

Método	Descrição	Teste passa se
assertEquals(a,b)	Compara dois valores	a.equals(b)
assertFalse(a)	Avalia uma expressão booleana	a == false
assertTrue(a)		a == true
assertNotNull(a)	Compara uma variável com nulo	a != null
assertNull(a)		a == null
assertNotSame(a,b)	Compara dois objetos	a == b
assertSame(a,b)		a != b
fail()	Causa uma falha no teste atual	

Testes Giovana:

ID do Teste	Objetivo do Teste	Código
1	Verificar se o retorno do ID está correto, então a validação é feita e, se o valor retornado for igual ao de entrada, será declarado como válido.	
2	Nesse caso, decidi fazer dois testes da mesma coisa. Em alguns testes o resultado retornado não estava sendo '-1' quando existia um problema. Por isso da decisão, para poder averiguar esse problema.	

```
public class TestAtualizaContoController {
    @Test
    public void testValidarIdValido() {
        AtualizaContoController atualizaConto = new AtualizaContoController();
        int idValido = atualizaConto.validarId("123", "Conto");
        assertEquals(123, idValido);
    }
```

```
    @Test
    public void testValidarIdInvalido() {

        AtualizaContoController atualizaConto = new AtualizaContoController();
        int idInvalido = atualizaConto.validarId("a", "Conto");
        assertEquals(-1, idInvalido);
    }
```

3	Nesse caso, verificamos a inclusão de heroi. Se o dado entrada for retornado exatamente igual. O resultado será válido. Depois de alguns testes, ele está válido. Tive alguns problemas em relação ao inserir por isso a verificação é feita diretamente em Heroi	 ```java package projeto1.appmitologia.controller; import org.junit.Test; import projeto1.appmitologia.model.Heroi; import static org.junit.Assert.assertEquals; public class TestIncluiHeroiController { @Test public void testIncluiHeroi() { IncluiHeroiController atualizaHeroi = new IncluiHeroiController(); Heroi heroi = new Heroi(); heroi.setNome("Novo Nome"); heroi.setDescricao("Nova Descrição"); heroi.setUrl("nova-imagem.com"); assertEquals("Novo Nome", heroi.getNome()); assertEquals("Nova Descrição", heroi.getDescricao()); assertEquals("nova-imagem.com", heroi.getUrl()); } } ```
4	Nesse caso, verificamos a inclusão de Conto. Se o dado entrada for retornado exatamente igual. O resultado será válido. Depois de alguns testes, ele está válido. Tive os mesmos problemas que em inserir Herói, por isso o uso direto de Conto	 ```java package projeto1.appmitologia.controller; import org.junit.Test; import projeto1.appmitologia.model.Conto; import static org.junit.Assert.assertEquals; public class TestIncluiContoController { @Test public void testIncluiConto() { IncluiContoController atualizaHeroi = new IncluiContoController(); Conto conto = new Conto(); conto.setNome("Novo Nome"); conto.setDescricao("Nova Descrição"); conto.setNomeHeroi("Novo NomeHeroi"); conto.setLocalizacao("Novo Localizacao"); assertEquals("Novo Nome", conto.getNome()); assertEquals("Nova Descrição", conto.getDescricao()); assertEquals("Novo NomeHeroi", conto.getNomeHeroi()); assertEquals("Novo Localizacao", conto.getLocalizacao()); } } ```