

Atividade de Eng. Software - Testes Unitários

Aluno: Luiz Gustavo Klitzke

Parte I - Realizar manualmente um teste de funcionalidade do sistema

Ao iniciar a aplicação, o menu apresenta 5 funcionalidades, selecionáveis ao digitar seu respectivo código numérico, sendo elas:

1. Consultar por um cliente

Ao selecionar essa opção, é requisitado o ID do cliente para ser consultado.

Caso o ID pertença a um cliente cadastrado na base, suas informações são apresentadas, contendo ID, Nome, Email, Idade e Status.

Caso nenhum cliente seja encontrado com esse ID, a mensagem "Nenhum cliente encontrado!" é apresentada.

2. Consultar por uma conta corrente

Ao selecionar essa opção, é requisitado o ID de uma conta para ser consultado.

Caso o ID pertença a uma conta cadastrada na base, suas informações são apresentadas, contendo ID, Saldo e Status.

Caso nenhuma conta corrente seja encontrada com esse ID, a mensagem "Conta não encontrado!" é apresentada.

3. Ativar um cliente

Ao selecionar essa opção, é requisitado o ID do cliente que deseja ativar.

Caso seja encontrado um cliente com esse ID, o mesmo é ativado, apresentando a mensagem "Cliente ativado com sucesso!".

Caso nenhum cliente seja encontrado para esse ID, é apresentada a mensagem "Cliente não encontrado!".

4. Desativar um cliente

Ao selecionar essa opção, é requisitado o ID do cliente que deseja desativar.

Caso seja encontrado um cliente com esse ID, o mesmo é desativado, apresentando a mensagem "Cliente desativado com sucesso!".

Caso nenhum cliente seja encontrado para esse ID, é apresentada a mensagem "Cliente não encontrado!".

5. Sair

Ao selecionar essa opção, é apresentada a mensagem "##### Sistema encerrado #####" e a aplicação é finalizada.

Parte II - Implementar os testes de unidade do sistema, utilizando a biblioteca JUnit.

Os testes para a classe gerenciadoraClientes e gerenciadoraContas foram implementados dentro do package "testes", nas classes GerenciadoraClientesTeste e GerenciadoraContasTeste, respectivamente, utilizando o JUnit5 e Jupiter.

Ambos os testes foram agrupados em um `Suite` na classe chamada `TesteRegressao`, presente no mesmo package. Segue abaixo um print de sua execução através da IDE Eclipse:

The screenshot displays the Eclipse IDE interface. The top toolbar shows standard development tools. The left sidebar contains the 'Project Explorer' and 'JUnit' views. The 'JUnit' view shows a test suite 'TesteRegressao' with a total duration of 0.220s. It lists various tests, including 'removeConta Existente', 'clienteAtivo Ativo', 'pesquisaCliente Inexistentes', 'adicionaCliente', 'clienteAtivo Inexistente', 'limpa', 'clienteAtivo Inativo', 'validade Válidos', 'getClientesDoBanco', 'pesquisaCliente Existentes', 'validade Inválidos', 'removeCliente Inexistente', 'adicionaConta', 'removeConta Existente' (with sub-tests for IDs 1, 2, 3, 4, 5), 'transfereValor', 'pesquisaConta Existentes', 'contaAtiva Inexistente', 'getContasDoBanco', 'pesquisaConta Inexistentes', 'contaAtiva Ativo', 'removeConta Inexistente', and 'contaAtiva Inativo'. The 'Failure Trace' at the bottom shows an 'AssertionFailedError' with the message 'expected: <false> but was: <true>'.

```
88 {
89     Assertions.assertTrue(this.gerenciadoraContas.removeConta(id));
90     Assertions.assertEquals(5, this.gerenciadoraContas.getContasDoBanco().size());
91 }
92
93 @DisplayName("removeConta Inexistente")
94 @ParameterizedTest(name = "id: {0}")
95 @ValueSource(ints = { 99, 0, 23 })
96 public void removeContaInexistente_Test(int id)
97 {
98     Assertions.assertFalse(this.gerenciadoraContas.removeConta(id));
99     Assertions.assertEquals(5, this.gerenciadoraContas.getContasDoBanco().size());
100 }
101
102 @DisplayName("contaAtiva Ativo")
103 @ParameterizedTest(name = "id: {0}")
104 @ValueSource(ints = { 1, 3, 5 })
105 public void contaAtivaAtivo_Test(int id)
106 {
107     Assertions.assertTrue(this.gerenciadoraContas.contaAtiva(id));
108 }
109
110 @DisplayName("contaAtiva Inativo")
111 @ParameterizedTest(name = "id: {0}")
112 @ValueSource(ints = { 2, 4 })
113 public void contaAtivaInativo_Test(int id)
114 {
115     Assertions.assertFalse(this.gerenciadoraContas.contaAtiva(id));
116 }
117
118 @DisplayName("contaAtiva Inexistente")
119 @ParameterizedTest(name = "id: {0}")
120 @ValueSource(ints = { 99, 82, 22, 0 })
121 public void contaAtivaInexistente_Test(int id)
122 {
123     Assertions.assertFalse(this.gerenciadoraContas.contaAtiva(id));
124 }
125
126 private static Stream<Arguments> transfereValor_Test() {
127     return Stream.of(
128         Arguments.arguments(true, 1, 2, 1.00, 9.22, 34.99),
129         Arguments.arguments(false, 1, 2, 100.00, 10.22, 33.99),
130         Arguments.arguments(true, 2, 3, 10.00, 23.99, 1031.1),
131         Arguments.arguments(true, 5, 1, 4.20, 0.00, 14.42)
132     );
133 }
134
135 @DisplayName("transfereValor")
136 @ParameterizedTest(name = "resultado: {0}, idOrigem: {1}, idDestino: {2}, valorTransf: {3}, esperadoOrigem: {4}, esperadoDestino: {5}")
137 @MethodSource()
138 public void transfereValor_Test(boolean resultado, int idOrigem, int idDestino, double valorTransf, int esperadoOrigem, int esperadoDestino)
139 {
140     Assertions.assertEquals(resultado, this.gerenciadoraContas.transfereValor(idOrigem, valorTransf));
141     Assertions.assertEquals(esperadoOrigem, this.gerenciadoraContas.pesquisaConta(idOrigem));
142     Assertions.assertEquals(esperadoDestino, this.gerenciadoraContas.pesquisaConta(idDestino));
143 }
144
145 }
```