Plano de testes – Validar funcionamento da classe ContaBancaria			
Caso	Descrição	Entrada	Saída esperada
1	Conferir se o método getSaldo pega o saldo da conta.	Criar uma conta.	getSaldo() deve retornar um valor double 0.
2	Verificar se o método Listar retorna a lista de MovimentosFinaceiros.	Criar uma conta.	Listar() deve retornar uma lista vazia.
3	Verificar se Incluir adiciona na lista.	Criar uma conta e um movimento.	A lista da conta deve possuir um movimento financeiro.
4	Verificar se o método ConsultarSaldoAtual() só pega os valores até o dia de hoje.	Criar uma conta e dois movimentos, um com data até no máximo hoje e outro para o futuro.	Deve retornar com o valor do movimento até a data atual.
5	Verificar ConsultarSaldoPeriodo() retorna o valor independente da data.	Criar uma conta e dois movimentos, um com data até no máximo hoje e outro para o futuro.	Deve retornar com o valor somado de ambas.
6	Verificar ConsultarSaldo() retorna um valor até a data informada.	Criar uma conta e dois movimentos, um com data até no máximo hoje e outro para o futuro. Informar uma data para filtrar até qual momento deseja verificar o saldo.	Retorna o valor filtrado até o período especificado.
7	Verificar se CriarArquivo() cria o arquivo “arquivoContaBancaria.csv” no repositório.	Criar conta e chamar o método.	O arquivo só deve ter o cabeçalho.
8	Verificar se SalvarArquivo() salva as informações da lista no arquivo especificado.	Criar conta, movimento e arquivo.	O arquivo especificado deve conter as informações do movimento.
9	Verificar se LerArquivo() importa as informações do csv	Criar conta e utilizar arquivo com informações de movimentos e importar.	A lista de movimentos da conta deve conter todos os movimentos no arquivo.

Plano de testes – Validar funcionamento da classe MovimentoFinanceiro			
Caso	Descrição	Entrada	Saída esperada
1	Conferir se o método getNome pega o nome do movimento.	Criar um movimento.	getNome() deve retornar o nome.
2	Verificar se setNome() seta o nome.	Criar movimento e utilizar setter para mudar nome.	O nome deve ser o mudado.
3	Verificar se setNome() não aceita string vazia.	Criar movimento sendo o nome uma string vazia.	Deve retornar uma exceção.
4	Verificar se getData() pega a data do movimento.	Criar movimentos com data.	Deve retornar data especificada.
5	Verificar se setData() seta a data.	Criar movimento e utilizar setter para mudar a data.	A data deve ser pela mudada.
6	Verificar se setData() não aceita data vazia.	Criar movimento sendo a data vazia.	Deve retornar uma exceção.
7	Verificar se getCategoria() pega a categoria do movimento.	Criar movimentos com categoria.	Deve retornar a categoria especificada.
8	Verificar se setCategoria() seta a categoria.	Criar movimento e utilizar setter para mudar a categoria.	A categoria deve ser pela mudada.
9	Verificar se setCategoria() não aceita data vazia.	Criar movimento sendo a data vazia.	Deve retornar uma exceção.
7	Verificar se getValor() pega o valor do movimento.	Criar movimentos com valor.	Deve retornar o valor especificado.
8	Verificar se setValor() seta o valor.	Criar movimento e utilizar setter para mudar o valor.	O valor deve ser pelo mudado.
9	Verificar se setValor() não aceita valor negativo.	Criar movimento sendo o valor negativo.	Deve retornar uma exceção.
10	Verificar Equals() entre objetos.	Criar um movimento e comparar com outro movimento.	Se forem iguais retorna verdadeiro.
11	Verificar hashCode().	Criar movimento e criar seu hash.	Deve retornar um número inteiro.
12	Verificar se toString() retorna uma string do objeto.	Criar movimento e chamar seu toString.	Deve retornar uma string modificada do objeto.