Lista de Exercícios 06

Questão 1

Implemente o seguinte plano de testes usando JUnit:

Plano de testes PL01 - Objetivo é validar se a classe `Pessoa` do exercício 1 da lista 2 atende aos requisitos propostos.			
Caso	Descrição	Entrada	Saída esperada
1	Verificar se é recusada altura negativa via setter	Criar uma instância com o construtor padrão e definir –1.75 como altura	Deve ser lançada a exceção `IllegalArgumentException`
2	Verificar se é recusado peso negativo via setter	Criar uma instância com o construtor padrão e definir –50 com peso	Deve ser lançada a exceção `IllegalArgumentException`
3	Verificar se é recusado nome vazio via setter	Criar uma instância com o construtor padrão e definir “” como o nome	Deve ser lançada a exceção `IllegalArgumentException`
4	Verificar se o construtor não padrão recusa altura negativa	Criar uma instância com o construtor que aceita argumentos, definindo um valor de altura negativo, o valor de nome como “Maria” e o valor de peso como 58	Deve ser lançada a exceção `IllegalArgumentException`
5	Verificar se o construtor não padrão recusa peso negativo	Criar uma instância com o construtor não padrão, definindo um peso como negativo, o valor de nome como “João” e o de altura como 1.90	Deve ser lançada a exceção `IllegalArgumentException`
6	Verificar se o construtor não padrão recusa nome composto de caracteres em branco	Criar uma instância com o construtor não padrão, definindo o nome como “ ”, o valor de peso como 50 e altura como 1.70	Deve ser lançada a exceção `IllegalArgumentException`
7	Verificar se o calculo de IMC está correto	Criar uma instância de pessoa, definindo o peso dela como 58 e a altura como 1.65	O método `calcularImc()` deve retornar 21,30 (pode-se aceitar 0,01 de diferença)
8	Verificar se o calculo de IMC está correto	Criar uma instância de pessoa, definindo o peso dela como 90 e a altura como 1.80	O método `calcularImc()` deve retornar 27,78 (pode-se aceitar 0,01 de diferença)
9	Verificar se o calculo de IMC está correto	Criar uma instância de pessoa, definindo o peso dela como 75 e a altura como 1.75	O método `calcularImc()` deve retornar 24,49 (pode-se aceitar 0,01 de diferença)

Questão 2

Implemente o seguinte plano de testes usando JUnit para a classe `ContaBancaria` da lista de exercícios 2.

Plano de testes PL01 - Objetivo é validar se a classe `ContaBancaria` atende aos requisitos propostos.			
Caso	Descrição	Entrada	Saída esperada
1	Verificar se a operação de saque barra saques sem saldo	Criar uma conta bancária e tentar sacar 100.00	Deve ser lançada a exceção `IllegalArgumentException`
2	Verificar se a operação de saque barra valores negativos	Criar uma conta bancária e tentar sacar -100.00	Deve ser lançada a exceção `IllegalArgumentException`

3	Verificar se a operação de depósito barra valores negativos	Criar uma conta bancária e tentar depositar -100.00	Deve ser lançada a exceção `IllegalArgumentException`
4	Verificar se o saldo está sendo contabilizado corretamente	Criar uma conta bancária, depositar 100.00, depois depositar 2500.00, e depois sacar 1200.00	getSaldo() = 1400.00
5	Verificar se o saldo está sendo contabilizado corretamente	Criar uma conta bancária, depositar 850.00, sacar 500.00, depois depositar 1200.00, depois sacar 859.92	getSaldo() = 690,08
6	Verificar as validações de saldo da transferência	Criar uma conta bancária c1 e c2, depositar 100.00, em c1 e tentar transferir 200.00 para c2	Deve ser lançada a exceção `IllegalArgumentException` (conferir se a operação de sacar é o primeiro método invocado depois das validações)
7	Verificar se o saldo fica correto depois da transferência	Criar uma conta bancária c1 e c2, depositar 100.00 em c1, depositar 550.00 em c2 e transferir 100.00 para c2	c2.getSaldo() = 650.00
8	Verificar se o saldo fica correto depois da transferência	Criar uma conta bancária c1 e c2, depositar 250.00 em c1, depositar 550.00 em c2 e transferir 100.00 para c2	c1.getSaldo() = 150.00

Questão 3

Modifique a classe `Fracao` da lista de exercícios 2, para incluir um método imprimir(): String que deve retornar uma String representando a Fração, o formator dessa string deve ser: numerador"/"denominador. Depois disso implemente o seguinte plano de testes usando JUnit.

Plano de testes PL01 - Objetivo é validar se a classe `Fracao` atende aos requisitos propostos.			
Caso	Descrição	Entrada	Saída esperada
1	Verificar se o construtor recusa denominador 0	Criar uma fração informando o numerador como 1 e o denominador como 0	Deve ser lançada a exceção `IllegalArgumentException`
2	Verificar se os sinais são ajustados	Criar uma fração informando o numerador como 1 e o denominador como -2	imprimir() = -1/2
3	Verificar se os sinais são ajustados	Criar uma fração informando o numerador como -1 e o denominador como -2	imprimir() = 1/2
4	Verificar se a operação de somar frações funciona corretamente	Criar uma fração f1 com numerador 1 e denominador 3 e uma	f3.imprimir() = 10/12

		fração f2 com numerador 2 e denominador 4 e invocar o método de somar passando como parâmetro f2, gerando a fração f3	
5	Verificar se a operação de somar frações e números funciona corretamente	Criar uma fração f1 com numerador 1 e denominador 3 e invocar o método de somar passando o número 2, gerando a fração f3	f3.imprimir() = 7/3
6	Verificar se a operação de somar frações negativas funciona corretamente	Criar uma fração f1 com numerador -1 e denominador 2 e uma fração f2 com numerador 1 e denominador 4 e invocar o método de somar passando como parâmetro f2, gerando a fração f3	f3.imprimir() = -2/8
7	Verificar se a operação de somar frações e números negativos funciona corretamente	Criar uma fração f1 com numerador 1 e denominador 3 e invocar o método de somar passando o número -2, gerando a fração f3	f3.imprimir() = -5/3