| **Plano de testes PL01 - Objetivo é validar se as classes dentro do pacote “model” atendem aos requisitos propostos.** | | | |
| --- | --- | --- | --- |
| Caso | Descrição | Entrada | Saída esperada |
| 1 | Validar que é recusado criar uma despesa com valor negativo. | Criar uma despesa com saldo negativo igual a R$ -400,00 | Deve ser lançada a exceção IllegalArgumentException |
| 2 | Validar que é recusado cria uma  receita com valor negativo. | Criar uma receita com saldo negativo igual a R$ -300,00 | Deve ser lançada a exceção IllegalArgumentException |
| 3 | Validar que é possível pegar a quantidade de movimentações. | Criar uma despesa e uma receita, incluir tanto a receita quanto a despesa e depois verificar se as duas foram incluídas. | O resultado deve ser 2. |
| 4 | Validar se a ordenação por data está funcionando. | Criar 3 movimentações e incluí-las no List<Movimentacao> e após isso verificar se foram ordenadas corretamente por data. | A ordem correta deve ser:  movimentação2  movimentação1  movimentação3 |
| 5 | Validar que é recusado criar uma despesa com o dia inválido. | Criar uma despesa com o dia igual a zero. | Deve ser lançada a exceção IllegalArgumentException |
| 6 | Validar que é recusado criar uma receita com o dia inválido. | Criar uma receita com o dia igual a zero. | Deve ser lançada a exceção IllegalArgumentException |
| 7 | Validar que é recusado criar um gerenciamento com o dia inválido. | Criar um gerenciamento com o dia igual a zero. | Deve ser lançada a exceção IllegalArgumentException |
| 8 | Validar se a despesa é realmente criada conforme proposto no código. | Criar uma despesa com os valores:10, 11, 2023, 50, TipoDespesa.ALIMENTACAO e assim verificar se foi realmente criada. | Espera-se: 10 para receita.getDia();  Espera-se: 11 para receita.getMes();  Espera-se: 2023 para receita.getAno();  Espera-se: 50 para receita.getValor();  Espera-se: TipoDespesa.ALIMENTACAO para despesa.getTipo(); |
| 9 | Validar se a receita é realmente criada conforme proposto no código. | Criar uma despesa com os valores:15, 5, 2023, 100, TipoReceita.SALARIO e assim verificar se foi realmente criada. | Espera-se: 15 para receita.getDia();  Espera-se: 5 para receita.getMes();  Espera-se: 2023 para receita.getAno();  Espera-se: 100 para receita.getValor();  Espera-se: TipoReceita.SALARIO para despesa.getTipo(); |
| 10 | Validar se é possível pegar a(s) receita(s) incluída(s). | Criar duas despesas e somente uma receita. Após isso incluir todas as movimentações em ArrayList<Receita> e verificar se o receitas.size() funciona. | O resultado deve ser 1 e também verifica se realmente foi pego a receita1; |
| 11 | Validar se é possível pegar a(s) despesa(s) incluída(s). | Criar duas despesas e somente uma receita. Após isso incluir todas as movimentações em ArrayList<Despesa> e verificar se o despesas.size() funciona. | O resultado deve ser 2 e também verifica se realmente foi pego a despesa1 e despesa2; |
| 12 | Validar que é recusado criar uma despesa com ano inválido. | Criar uma despesa com o ano -2023. | Deve ser lançada a exceção IllegalArgumentException |
| 13 | Validar que é recusado criar uma receita com ano inválido | Criar uma receita com o ano -2023. | Deve ser lançada a exceção IllegalArgumentException |
| 14 | Validar que é recusado criar uma despesa com o mês inválido. | Criar uma despesa com o mês 13. | Deve ser lançada a exceção IllegalArgumentException |
| 15 | Validar que é recusado criar uma receita com o mês inválido. | Criar uma receita com o mês 13. | Deve ser lançada a exceção IllegalArgumentException |
| 16 | Validar que é recusado criar uma despesa com o saldo inválido. | Criar uma despesa com o saldo igual a R$ 0,00. | Deve ser lançada a exceção IllegalArgumentException |
| 17 | Validar que é recusado criar uma receita com o saldo inválido. | Criar uma receita com o saldo igual a R$ 0,00. | Deve ser lançada a exceção IllegalArgumentException |
| 18 | Validar que é possível criar uma despesa. | Criar uma despesa com os seguintes valores: 19, 11, 2003, 2, TipoDespesa.EDUCACAO. | Espera-se a própria despesa. |
| 19 | Validar que é possível criar uma receita | Criar uma despesa com os seguintes valores: 19, 11, 2003, 3, TipoReceita.DECIMO\_TERCEIRO. | Espera-se a própria receita. |
| 20 | Validar que é recusado cria um gerenciamento com o caminho de arquivo errado. | Criar um gerenciamento e dar o nome do arquivo errado. | Deve ser lançada a exceção IllegalArgumentException |