T1 – Engenharia de Software III

Camilla Damasceno
Gabriel Souza Gomes
Luiza Ávila

Requisitos:

- O sistema deve dar um desconto de $1 caso CPF seja fornecido
- O sistema deve guardar receita total gerada
- O sistema não deve permitir venda de itens a mais do que existem no estoque
- O sistema deve avisar estoque vazio de um item

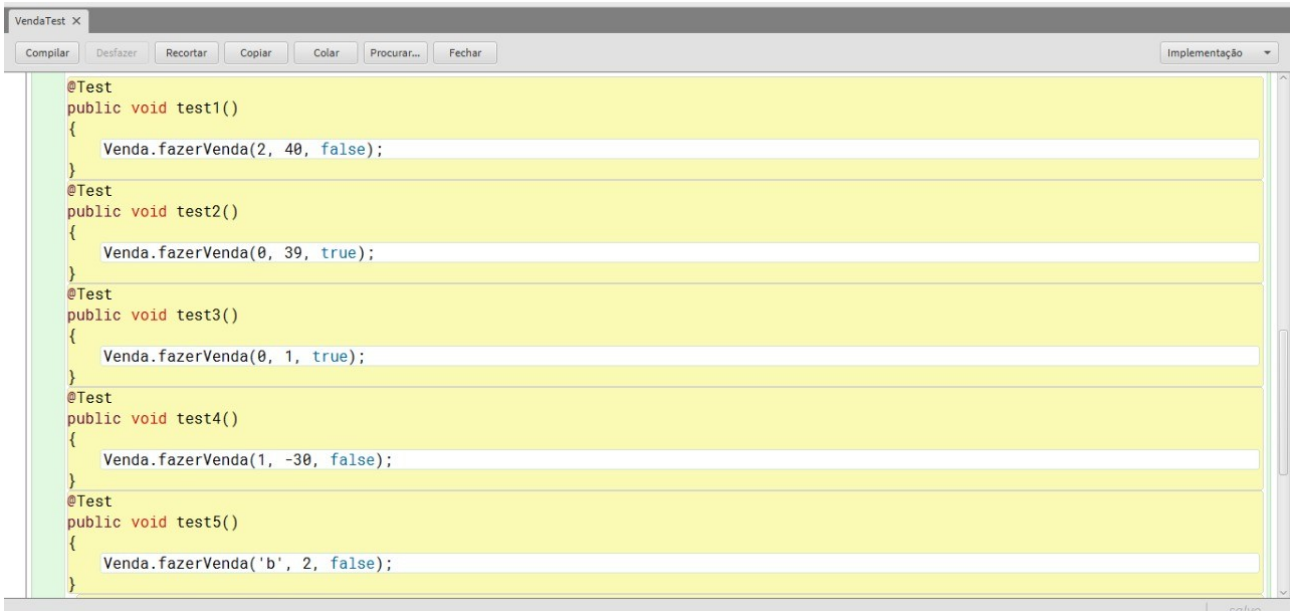
Classe em Java:

```java
class Venda {
    // Matriz que guarda, nesta ordem: codigo do produto, quantidade disponivel no estoque, e preço
    static int[][] products = {{0, 40, 5},
                               {1, 10, 30},
                               {2, 250, 2}};
    public static int receitaTotal = 0;

    public static String fazerVenda(int codProduto, int quant, boolean cpf) {
        int precoVenda = products[codProduto][2];
        if (cpf)
            precoVenda--;

        // Defeito 1: o if abaixo deveria ter >= ao invés de >
        if (products[codProduto][1] > quant) {
            products[codProduto][1] -= quant;
            receitaTotal += products[codProduto][2] * quant; // Defeito 2: total adicionado deveria ser precoVenda, e não o preço original

            if (products[codProduto][1] == 0)
                System.out.println("Estoque do protudo " + codProduto + " vazio");

            return "Registrada venda de " + quant + " #" + codProduto + " por $" + precoVenda;
        }
        else
            return "Nao ha estoque suficiente para realizar esta venda";
    }
}

// Classe driver
public class MyClass {
    public static void main(String args[]) {
      Venda venda = new Venda();
      System.out.println("Venda 1: ");
      System.out.println(venda.fazerVenda(2, 40, false));
      System.out.println("Venda 2: ");
      System.out.println(venda.fazerVenda(0, 39, true));
      System.out.println("Receita total: " + venda.receitaTotal); // O defeito 2 fará com que o valor incorreto seja mostrado
      System.out.println("Venda 3: ");
      System.out.println(venda.fazerVenda(0, 1, true)); // O defeito 1 invalidará esta venda incorretamente
      System.out.println("Venda 4: ");
      // Uma vulnerabilidade na classe fará com que o estoque de itens aumente com esta venda, além de diminuir a receita total
      System.out.println(venda.fazerVenda(1, -30, false));
      System.out.println("\nReceita total: " + venda.receitaTotal);
    }
}
```

Partições de equivalência:

Entrada	Classes de equivalência válidas	Classes de equivalência inválidas
Código	Inteiro maior ou igual a 0 menor que 3	Inteiro menor que 0, maior ou igual a 3; não é inteiro
Quantidade	Inteiro maior que 0, menor ou igual do estoque	Não é inteiro, menor que zero
CPF	Boolean false ou true	Não é boolean

Testes executados via Unit:




```
@Test
public void test1()
{
    Venda.fazerVenda(2, 40, false);
}
@Test
public void test2()
{
    Venda.fazerVenda(0, 39, true);
}
@Test
public void test3()
{
    Venda.fazerVenda(0, 1, true);
}
@Test
public void test4()
{
    Venda.fazerVenda(1, -30, false);
}
@Test
public void test5()
{
    Venda.fazerVenda('b', 2, false);
}
```

```
@Test
public void test6()
{
    Venda.fazerVenda(3, 4, true);
}
@Test
public void test7()
{
    Venda.fazerVenda(0, 'b', false);
}
@Test
public void test8()
{
    Venda.fazerVenda(-1, 30, false);
}
```

Casos de teste:

Teste	Dados	Resultado Esperado	Resultado Obtido
01	Código 2, quantidade 40, sem CPF	Funcione	Funcionou
02	Código 0, quantidade 39, com CPF	Funcione	Funcionou
03	Código 0, quantidade 1, com CPF	Funcione	Funcionou
04	Código 1, quantidade -30, sem CPF	Erro	Funcionou
05	Código ‘b’, quantidade 2, sem CPF	Erro	Erro
06	Código 3, quantidade 4, com CPF	Erro	Erro
07	Código 0, quantidade ‘b’, sem CPF	Erro	Funcionou
08	Código -1, quantidade 30, sem CPF	Erro	Erro

Avaliação dos resultados:

Os testes 01, 02, 03 tiveram obtiveram sucesso no funcionamento.

No Teste 04, o RE era de erro, pois a quantidade foi negativa, mas o RO foi sucesso no funcionamento, pois o erro foi tratado com uma mensagem de alerta no próprio código.

No Teste 05, o RE e o RO foram erro, pois, ao pegar o código ascii de ‘b’, o parâmetro código recebe um valor que faz com que tente acessar uma posição inexistente na matriz products. No caso do RE ser erro, se deve ao fato de que o parâmetro código espera um int, mas recebeu um char.

No teste 06, o RE e o RO foram erro, pois, como a quantidade foi 4, tentou-se acessar uma posição inexistente na matriz products.

No teste 07, o RE era erro, pois o parâmetro quantidade espera um int, mas recebeu um char, mas o RO foi sucesso no funcionamento, pois ao converter automaticamente para o valor inteiro equivalente em ascii, quantidade recebe um valor maior que products[codProduto][1], mas que é tratado no código por mensagem de alerta.

No Teste 08, RE era erro, pois código não pode ser um valor negativo, e RO também foi erro, pois tentou-se acessar uma posição inexistente (“negativa”) no vetor products.