SafeWay

Teste Dev./Operacional

Erros encontrados

Experiência do usuário:

- Na linha 127 da Classe Main, ao efetuar login com cliente, cliente2 ou admin a palavra “Realizar” está escrito de maneira errada.

```
System.out.println("1 - Relizar Compras");
```

- A numeração das empresas está ao contrário, isso pode confundir na hora de consultar alguma delas (Empresa está com número 2, e empresa2 está com número 1).

```
            Empresa empresa = new Empresa(2, "SafeWay Padaria",
"30021423000159", 0.15, 0.0);
```

```
            Empresa empresa2 = new Empresa(1, "Level Varejo",
"53239160000154", 0.05, 0.0);
```

- Linha 29, “Chá” escrito de maneira errada.

```
Produto produto10 = new Produto(10, "Ché Gelado", 4, 5.50, empresa);
```

- Linha 20, “Francês” escrito de maneira errada. (UTF-8 Já configurado, acentuação pode ser realizada sem problemas)

```
Produto produto = new Produto(1, "Pão Frances", 5, 3.50, empresa);
```

- Na criação de produtos, poderia ter seguido uma ordem numérica na hora de criar produtos da mesma empresa, ficar com números espalhados na hora de escolher um pedido dificulta para o usuário, deveriam ficar agrupados por empresa.

```
Produto produto = new Produto(1, "Pão Francês", 5, 3.50, empresa);
```

```
Produto produto2 = new Produto(2, "Coturno", 10, 400.0, empresa2);
```

```
Produto produto3 = new Produto(3, "Jaqueta Jeans", 15, 150.0, empresa2);
```

No caso o **produto2** deveria ser da **empresa** também, e não da empresa2, ao fazer experiência como usuário senti um pouco de dificuldade com os números ordenados de qualquer maneira.

Boas Práticas:

- O Scanner da classe Main não foi fechado.

- A organização de classes está muito solta, melhor criar pacotes e agrupar classes com funcionalidades semelhantes.

- O código da classe Main está repleto de “ifs”, “cases”, gerando uma complexidade desnecessária, melhor separar em classes afim de diminuir a complexidade.

- Se um dia a empresa decidir mudar sua taxa, vai ter que procurar no código onde foi colocado o antigo valor da taxa, com a aplicação do *Design Pattern Strategy*, é possível corrigir isso, e deixar de uma maneira mais legível e fácil de se dar manutenção.

- Para se trabalhar com dinheiro e números decimais é aconselhável utilizar a classe *BigDecimal* invés de Double, ela consegue fazer arredondamentos e lidar com decimais de melhor forma.

- Eu acho importante colocar os métodos Equals e Hashcode nas classe: Empresa, Produto, Cliente, para garantir registros de objetos individuais, já que o banco está sendo simulado por listas.

- Creio que o ID de empresas e CÓDIGO em vendas, devem ser LONG invés de INTEGER, afim de ter mais capacidade de bytes.

- A classe Main está extensa demais, quebrar em outras classes seria uma melhor prática, tanto para manutenção quanto para legibilidade do código.

- Regras de negócio: ************

- A empresa consegue vender produtos que não está relacionada a ela. Ao selecionar produtos de uma determinada empresa, aparece a listagem numérica, se por acidente digitarmos um número que não está ali na lista e esse número fazer parte da lista de outra empresa, esse item será adicionado ao carrinho, isso é um erro, onde uma digitação errada do usuário ocasiona em uma compra de um produto não desejável.

- O admin é tratado como um “Não empresa”, porém, ele também não é um usuário.

- O limite de mercadoria também não está sendo respeitado, é possível comprar mais produtos do que existem em estoque.

- **Código:**

Com os relatos anteriores, algumas partes do código foram alteradas afim de satisfazer a regra de negócio. E outras mudanças foram feitas para um ajuste de melhor legibilidade.

- Criei uma interface chamada Taxa, afim de forçar que cada empresa realmente tenha sua própria taxa, e para facilitar futuras mudanças de valor.

- O admin tem o poder de ver todos os dados de todas empresas já cadastradas.

- Para evitar exceptions que são geradas ao digitar username e senha errados, adicionei o “`executar(usuarios, clientes, empresas, produtos, carrinho, vendas);`”

Após as mensagens de erro, afim do programa continuar rodando sem parar.