UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
BOAS PRÁTICAS DE PROGRAMAÇÃO
2024.2

Estudo dirigido 04 – Testes e Depuração
Unidade 04

Thaynan Paulo Fernandes Bezerra de Mendonça

1. TESTES E DEPURAÇÃO

Tanto Testes quando a depuração é utilizada para verificar erros no código.

Os testes verificam a entrada e a saída de valores em um sistema. Podem ser unitário (quando são realizados nas suas unidades – JUnit) ou de integração (quando se verifica se as unidades do sistema funcionam juntas - Mockito).

Já a depuração é o processo de procura e identificação de erros no sistema. No código abaixo ocorreu através da criação de logs a partir da biblioteca java.util.logging e da criação de breakpoints através da IDE (netbeans)

2. APLICAÇÃO DOS PRINCÍPIOS DA DEPURAÇÃO E TESTES

Nesse tópico será apresentado os trechos do código que tiverem ajustes em relação ao teste e depuração

2.1 Testes Unitários

Os testes unitários foram realizados através do JUnit 5. Após a inserção da dependência no arquivo pom.xml.

Wxemplo de um codigo com teste unitário é o de testCalcularTotal() que o usa o assertEquals(esperado, total) para verificar se o valor está certo (para o caso acima de 100 que tem desconto)

```
double esperado = ((prod1.getPreco() + prod2.getPreco() + prod3.getPreco()) * 0.9);
System.out.println("valor total da compra é: " + esperado);
assertEquals(esperado, total);
```

2.2 Testes Integração

O teste de integração foi realizado através do Mockito visando à simulação da criação de um servidor. O Discente conseguiu colocar no código, entretanto teve dificuldade de ajeitar o teste que incluísse ele.

```java
public class Banco_ServidorTest {

    @Mock
    private Banco_Repositorio repositorio;

    @InjectMocks
    private Banco_Servidor servidor;

    public Banco_ServidorTest() {
        MockitoAnnotations.openMocks(this);
```

2.3 Logs

A criação de logs ocorreu através da biblioteca import java.util.logging.Logger apresentado na saída (output) do Netbeans.

Após a inicialização da biblioteca mensagens de confirmação de algumas ações como adicionar novo item ao catalogo ou desconto realizado foram lançados para acompanhamento.

```java
if (total > 100) {
    total = 0.9 * total;
    logger.info("Desconto de 10% realizado!" );
```

2.4 Breakpoints

O uso de breakpoints ocorreu em momentos importantes do código como por exemplo no momento da aplicação do desconto. No código abaixo os breakpoints foram na classe teste do pedido, mas há em alguns outros pontos do código.

```java
        Pedido Pedido01 = new Pedido("Thaynan");
        Pedido01.adicionarProduto(prod1);
        Pedido01.adicionarProduto(prod2);
        Pedido01.adicionarProduto(prod3);

        total = Pedido01.calcularTotal();

        double esperado = ((prod1.getPreco() + prod2.getPreco() + prod3.getPreco()) * 0.9);
        System.out.println("valor total da compra é: " + esperado);
        assertEquals(esperado, total);
```

3. Referências

https://medium.com/rpedroni/tudo-sobre-testes-testes-unit%C3%A1rios-vs-testes-de-integra%C3%A7%C3%A3o-vs-testes-e2e-6a7cc955779 acessado em 12-12-2024 às 04:10

https://www.techtarget.com/searchsoftwarequality/definition/debugging acessado às 12-12-2024 as 05:04