**ATIVIDADE TESTE UNITARIO**

**NOME: FÁBIO CARVALHO DE ALMEIDA RA: 172321018**

**ROGER OLIVEIRA RA: 172320937**

**BERNARDO GNOATTO RA: 172320338**

**CALCULADORA**

Esse código é um exemplo simples de testes unitários em Java usando o framework JUnit. Ele apresenta uma classe Calculadora com métodos para realizar operações matemáticas simples, como soma e subtração, e uma classe TesteCalculadora que contém métodos de teste para validar o comportamento desses métodos.

A classe Calculadora possui métodos para realizar operações matemáticas básicas:

soma(int x, int y): Retorna a soma dos dois números passados como argumento.

subtrai(int x, int y): Retorna a subtração do primeiro número pelo segundo.

A classe TesteCalculadora contém métodos de teste para testar a funcionalidade da classe Calculadora usando o framework JUnit. Os métodos de teste (testeSoma1 e testeSoma2) usam o método assertEquals para verificar se os resultados retornados pelos métodos da classe Calculadora correspondem aos resultados esperados.

**STACK**

Esse código contém duas classes: Main e Stack<T>. A classe Main contém um método main, enquanto a classe Stack<T> implementa uma estrutura de dados do tipo pilha (stack) usando um ArrayList para armazenar elementos de um tipo genérico T.

Aqui está uma descrição das funcionalidades principais:

Classe Main

A classe Main possui um método main(String[] args) que instancia um objeto TestRunner e chama o método main() desse objeto. No entanto, o código do TestRunner não está fornecido neste trecho.

Classe Stack<T>

Esta classe implementa uma pilha (stack) genérica, permitindo empilhar e desempilhar elementos de um tipo desconhecido <T>.

private ArrayList<T> elements: É uma lista interna usada para armazenar os elementos da pilha.

private int size: Mantém o controle do tamanho da pilha.

Essa implementação de pilha segue um padrão básico: os elementos são adicionados no topo e removidos do topo (princípio "last-in, first-out"). O ArrayList é utilizado para armazenar os elementos, e métodos são fornecidos para manipular a pilha de acordo com suas propriedades (tamanho, vazio, push e pop).

**SISTEMA DE BIBLIOTECA**

Esse código representa um exemplo simplificado de um sistema de biblioteca, incluindo classes como Biblioteca, Usuario, Livro, e um conjunto de testes unitários (BibliotecaTest). Vou explicar cada parte:

Este código ilustra boas práticas de organização de código, como separação de responsabilidades, injeção de dependência e testes de unidade.

Os testes ajudam a verificar se a classe Biblioteca está funcionando corretamente em diferentes cenários, como empréstimos, devoluções e tratamento de exceções.

Esses testes fornecem confiança de que a lógica de negócios implementada na classe Biblioteca está funcionando conforme o esperado, o que é crucial para garantir o correto funcionamento do sistema de uma biblioteca.

**MOCK**

Esse conjunto de classes parece estar relacionado à busca e à criação de objetos Book a partir de informações obtidas por meio de um serviço (BookService).

Resolução de busca de livros: Abstrai a lógica de busca de livros por ISBN, permitindo que a classe BookSearch utilize diferentes implementações de serviços (BookService) para buscar informações.

Flexibilidade e manutenibilidade: Ao separar a lógica de busca do restante do código, facilita-se a manutenção e a possibilidade de alterar a fonte dos dados de busca sem modificar a classe BookSearch.

Criação de objetos Book: A classe BookSearch é responsável por criar objetos Book a partir dos dados obtidos do serviço. Isso permite a criação de objetos Book de forma padronizada a partir das informações recebidas.

**TDD**

A classe ShoppingCart e seu teste associado (ShoppingCartTest) são parte de um sistema de carrinho de compras para livros, embora no estado atual eles não estejam funcionando corretamente. Ainda assim, vamos abordar o que essa estrutura busca resolver e sua utilidade quando estiver completa:

O que soluciona:

Gestão de Carrinho de Compras: A classe ShoppingCart é projetada para gerenciar a adição de livros ao carrinho e calcular o valor total da compra.

Cálculo do Valor Total: Uma vez corretamente implementada, a classe ShoppingCart solucionaria a lógica de calcular o valor total com base nos preços dos livros adicionados.

Atualmente, a classe ShoppingCart não está fazendo a lógica necessária para calcular o total corretamente. Ela precisa descomentar e implementar o método add(Book b) para somar os preços dos livros adicionados ao carrinho, atualizando o total conforme os livros são inseridos.

Assim, quando o teste ShoppingCartTest é executado, ele verifica se a lógica implementada no método getTotal() está correta e se está retornando o valor total esperado do carrinho após adicionar os livros.

Se for descomentada e implementada corretamente, a classe ShoppingCart será capaz de adicionar livros e calcular o total dos preços desses livros no carrinho de compras. O teste verifica se essa funcionalidade está correta.

**PRINCIPAIS DIFERENÇAS ENTRE OS TESTES**

Estrutura e Propósito:

Calculadora & Testes Unitários:

Apresenta uma calculadora simples com métodos de soma e subtração.

Testes unitários verificam se os métodos da calculadora retornam os resultados esperados para diferentes entradas.

Sistema de Bibliotecas & Testes Unitários:

Define classes para representar uma biblioteca e métodos para adicionar, emprestar e receber livros.

Os testes unitários verificam o comportamento desses métodos em diferentes situações.

Busca de Livros & Testes Unitários:

Estrutura voltada para buscar informações de livros usando um serviço (BookService) e converter essas informações em objetos Book.

Testes unitários validam se a busca e a criação dos objetos Book estão funcionando corretamente.

Livro, Carrinho de Compras & Testes Unitários:

Define a classe Book representando um livro com atributos como título, preço e ISBN.

Estrutura um carrinho de compras (ShoppingCart) para adicionar livros e calcular o valor total.

O teste unitário verifica se o carrinho (ShoppingCart) calcula corretamente o valor total dos livros adicionados.

Funcionalidades:

Calculadora:

Realiza operações matemáticas básicas (soma e subtração).

Testes validam a precisão das operações matemáticas.

Sistema de Bibliotecas:

Gerencia acervo, empréstimos e devoluções de livros.

Testes verificam se as funcionalidades de adição e empréstimo de livros funcionam corretamente.

Busca de Livros:

Foco na busca e conversão de informações de livros usando um serviço específico.

Testes validam a busca e a criação correta de objetos Book.

Carrinho de Compras:

Gerencia adições de livros e cálculo do valor total.

Teste garante que o cálculo do valor total do carrinho está correto.

Abordagens e Contextos:

Variedade de Funcionalidades:

Cada código aborda um domínio diferente: cálculos matemáticos, operações de biblioteca, busca de informações de livros e gerenciamento de carrinho de compras.

Testes Unitários:

Todos os códigos têm uma ênfase na verificação de comportamento correto através de testes unitários, garantindo a funcionalidade adequada das classes e métodos implementados.

Interação com Serviços Externos:

O código de busca de livros (BookService) interage com um serviço externo para buscar informações de livros, enquanto outros códigos lidam principalmente com lógica interna das classes.

Essas são algumas das principais diferenças entre os códigos analisados, destacando as estruturas, funcionalidades e abordagens de cada um em seus respectivos contextos e propósitos.