Técnico em Desenvolvimento de Sistemas

Disciplina: Desenvolvimento de Sistemas

Nome:_____ **Grupo:**_____

Nome:_____ **Grupo:**_____

RECUPERAÇÃO II

Exercício 1 – Sistema de Gestão de Pedidos

Desenvolve um sistema de gestão de pedidos utilizando os princípios da Programação Orientada a Objetos em Java, com foco em herança, polimorfismo, encapsulamento e tratamento de exceções personalizadas. O projeto será dividido em três pacotes principais: model, que conterá as classes Produto, Livro, Eletronico, Roupa e Pedido; o pacote exceptions, que será responsável pelas exceções personalizadas PrecoInvalidoException e QuantidadeInvalidaException; e o pacote main, que terá a classe TestePedidos, usada para executar e testar o sistema.

A estrutura começa com a criação da classe abstrata Produto, que representa qualquer item comercializado. Ela possui os atributos privados nome, preco e quantidadeEstoque, que devem ser acessados apenas por meio de métodos getters e setters. Os setters incluem validações: o preço deve ser maior que zero e a quantidade em estoque não pode ser negativa. A classe também possui um construtor com parâmetros e um método abstrato chamado exibirDetalhes, que será implementado de forma específica em cada subclasse.

Três subclasses concretas especializam o conceito de produto. A classe Livro adiciona o atributo autor e sobrescreve o método exibirDetalhes para mostrar o nome, o autor, o preço e a quantidade em estoque. A classe Eletronico, por sua vez, inclui o atributo garantiaMeses e exibe detalhes completos do produto, incluindo o tempo de garantia. Já a classe Roupa traz o atributo tamanho, com opções como P, M, G e GG, e seu método exibirDetalhes mostra o nome, tamanho, preço e quantidade. Todas as subclasses devem utilizar super() em seus construtores para reaproveitar a lógica da superclasse.

Para garantir a integridade dos dados, o sistema conta com exceções personalizadas. A PrecoInvalidoException é lançada quando se tenta atribuir um preço menor ou igual a zero, enquanto a QuantidadeInvalidaException é usada quando o estoque é definido como um valor negativo. Essas exceções são lançadas diretamente

nos métodos setPreco() e setQuantidadeEstoque(), promovendo um código mais seguro e robusto.

A classe Pedido, também no pacote model, representa uma solicitação de compra, composta por um produto e a quantidade solicitada. Durante sua criação, é feita uma verificação para garantir que a quantidade solicitada não ultrapasse o estoque disponível. Um método chamado gerarResumo() pode ser usado para exibir os detalhes do pedido, combinando informações do produto e da quantidade comprada.

Por fim, a classe TestePedidos, no pacote main, permite testar todo o sistema. Nela, são criadas instâncias de Livro, Eletronico e Roupa, algumas com dados válidos e outras propositalmente com erros para testar o tratamento de exceções. Essas instâncias são armazenadas em um vetor do tipo Produto[], o que permite o uso de polimorfismo ao chamar o método exibirDetalhes() para cada item, independentemente da sua classe concreta. O uso de try/catch garante que as exceções lançadas durante a criação ou modificação dos objetos sejam tratadas com mensagens amigáveis ao usuário, promovendo uma boa experiência e depuração eficaz.

Exercício 2 – Sistema de reservas de passagens aérea

Você deve desenvolver um sistema Java orientado a objetos para uma companhia aérea chamada **SkyWings**, simulando reservas de passagens. O projeto deve ser organizado em pacotes:

- model: classes abstrata Passagem e subclasses PassagemEconomica, PassagemExecutiva e PassagemPrime.
- exceptions: exceções personalizadas ValorPassagemInvalidoException e ClasseInvalidaException.
- main: classe TesteReservas para testar o sistema.

A classe abstrata Passagem possui atributos como nome do passageiro, destino e valor da passagem (com validação mínima de R$100,00). Possui também um método abstrato mostrarDetalhes() a ser implementado nas subclasses.

Cada subclasse representa um tipo de passagem e possui atributos extras:

- **Econômica**: bagagemInclusa
- **Executiva**: numeroRefeicoes
- **Prime**: servicoPersonalizado

Na classe de testes, instancie objetos das três subclasses, armazene em um vetor de Passagem[], percorra com um laço e use mostrarDetalhes() para aplicar polimorfismo. Use try/catch para capturar exceções se os valores forem inválidos.