Programação orientada a objetos: interface

Exercício 1 *(Abstract class) Ingressos são comercializados a todo momento em Fortaleza, principalmente em bilheterias virtuais. Seja para cinema, show, micareta ou peças de teatro, é possível adquirir um ingresso online. São comercializados diferentes tipos de ingressos. Eles podem ser VIP ou não. Podem ser meia entrada ou não. Podem ser também de lotes distintos (1, 2, 3, ...).*

Implemente uma classe ***abstrata*** *chamada Ingresso. A classe deve possuir: atributo "nome" (recebe o nome do evento); atributo "meia" (indica se é meia entrada ou não); atributo "preço" (em reais); atributo "lote" (1, 2, 3, ...); método "mostra".*

Implemente as subclasses IngressoVip e IngressoComum. Elas herdam os mesmos atributos da classe Ingresso. Ambas as classes devem possuir o método obrigatório calculaReembolso.

- *Para ingressos comuns, o reembolso é de 5% do valor se a entrada é inteira ou se o lote é o primeiro. Do contrário, o reemboloso é de 3%;*
- *Para ingressos VIPs, o reembolso é de 10% do valor se a entrada é inteira e lote é o primeiro. Do contrário, o reemboloso é de 6%.*

Implemente o método calculaReembolso em cada uma das subclasses. Utilize ele como cabeçalho na classe abstrata Ingresso.

Na classe principal, declare e receba os n ingressos e armazene eles em um único vetor.

Interface: define um "contrato" (padrão) a ser seguido por todas as classes que a implementam.

Vamos implementar uma **interface** chamada Seguranca. Ela irá nos fornecer métodos que qualquer sistema que possui algo relacionado à segurança tem (por exemplo: um verificador de senha, um verificador de token, um verificador de CPF, etc.).

Exercício 2 *(Interface) Observe a problemática abaixo:*

- *Implemente a interface Seguranca. Ela deve possuir o método booleano verificaSenha(senha), além do método mensagemSenha() que mostra uma mensagem de sucesso/falha após o usuário digitar a senha.*

Implemente a classe Conta. A classe deve possuir como atributos: CPF, senha e saldo do usuário. Como métodos, deve possuir "saca(valor, senha)" e "deposita(valor, senha)". O ato de sacar ou depositar o valor só é realizado se a senha estiver correta! Portanto, a classe deve possuir o método verificaSenha(senha) para tal tarefa e mensagemSucesso() e mensagemFracasso() para informar ao usuário do sucesso/fracasso.

Implemente a classe Corretora. A classe deve possuir como atributos: CPF, senha e saldo do usuário. Como métodos, deve possuir "compraAcao(precoAcao, quantidadeAcao, senha)" e "vendaAcao(precoAcao, quantidadeAcao, senha)". O ato de aportar só é realizado se a senha estiver correta e o usuário possuir saldo suficiente! Portanto, a classe deve possuir o método verificaSenha(senha) para tal tarefa e mensagemSucesso() e mensagemFracasso() para informar ao usuário do sucesso/fracasso.

Sabemos que todo sistema bancário deverá possuir operações de saque e depósito! Sendo assim, não poderíamos implementar uma interface com esses métodos?

Interface fornece suporte para **polimorfismo**!

Exercício 3 *(Abstract class vs. Interface) Fazer o Exercício 1 usando Interface.*