INSTITUTO FEDERAL
Goiano
Campus Morrinhos

Programação Orientada a Objetos
Professor Msc. Marcel Melo

Você foi desafiado(a) a desenvolver um protótipo para um sistema de gerenciamento de investimentos para a corretora "InvesteMais". Este sistema precisa lidar com diferentes tipos de clientes e produtos, consolidando todos os ativos de um cliente em uma carteira de investimentos. Sua tarefa é modelar e implementar a estrutura de classes necessária para este sistema.

Primeiramente, você deverá criar a hierarquia de Clientes. Comece com uma classe abstrata Cliente, que possuirá os atributos nome (String) e email (String). Esta classe também deverá declarar um método abstrato public abstract String getIdentificadorDocumento();. Em seguida, crie duas classes concretas que herdam de Cliente: PessoaFisica e PessoaJuridica. PessoaFisica terá um atributo cpf e implementará getIdentificadorDocumento() para retorná-lo. Similarmente, PessoaJuridica terá um atributo cnpj e implementará o método para retornar seu respectivo identificador.

Para centralizar os ativos, crie uma classe CarteiraInvestimentos. Cada cliente terá sua própria carteira. Esta classe possuirá um cliente (do tipo Cliente) e uma lista de investimentos (ex: List<Investimento>). Seu construtor receberá o cliente dono da carteira. Implemente um método public void adicionarInvestimento(Investimento investimento) que adiciona um novo produto à lista, mas somente se o cliente do investimento for o mesmo cliente dono da carteira. Adicione também um método public double calcularValorTotalInvestido(), que deverá percorrer a lista de investimentos e retornar a soma do saldo atual de cada um deles, representando o valor total de mercado da carteira.

O coração do sistema será a estrutura de Investimentos. Crie uma classe abstrata Investimento , da qual todos os produtos financeiros herdarão. Ela deverá conter os atributos cliente (do tipo Cliente) e saldo (double). Seu construtor deve receber um cliente e um valor inicial. A classe Investimento oferecerá métodos para aplicar(double valor) e resgatar(double valor), que manipulam o saldo. O método de resgate não deve permitir que o saldo se torne negativo. Crucialmente, Investimento declarará dois métodos abstratos: public abstract double calcularSaldoProjetado(int numeroMeses);, que estimará o saldo futuro, e public abstract void simularPassagemDeMes();, que aplicará efetivamente os rendimentos e/ou taxas mensais ao saldo do investimento.

A partir da classe Investimento, você desenvolverá quatro tipos específicos de produtos:

1. **TesouroPrefixado**: Esta classe, que herda de Investimento, terá como atributos específicos nomeTitulo (String, ex: "Tesouro Prefixado 2029"), taxaJurosAnual (double, ex: 0.10 para 10%) e percentualImpostoRenda (double, fixo em 0.15 sobre o rendimento). Seu construtor deve receber todos os dados necessários. **Atenção**: Este produto é exclusivo para PessoaFisica. No construtor, verifique o tipo do cliente; se não for PessoaFisica, lance uma IllegalArgumentException. Ao implementar calcularSaldoProjetado(int numeroMeses), considere a taxa de juros mensal (taxaJurosAnual / 12), calcule o saldo bruto projetado (saldoAtual * Math.pow((1 + taxaJurosMensal), numeroMeses)), o rendimento bruto (saldoBrutoProjetado - saldoAtual), o imposto devido (rendimentoBruto * percentualImpostoRenda), e retorne o saldo líquido projetado (saldoAtual + rendimentoBruto - impostoDevido). Para simularPassagemDeMes(), calcule o rendimento do mês (this.getSaldo() * (this.taxaJurosAnual / 12.0)) e atualize o saldo somando este rendimento.

2. **AcaoBolsa**: Herdando de Investimento, esta classe terá os atributos codigoAcao (String, ex: "PETR4"), nomeEmpresa (String, ex: "Petrobras PN") e taxaCorretagemFixaMensal (double, ex: 10.00 reais). Seu construtor inicializará estes dados. Na implementação de calcularSaldoProjetado(int numeroMeses), para simplificar, assuma uma valorização mensal média

de 0.8% (0.008). Calcule o saldo bruto projetado (saldoAtual * Math.pow((1 + 0.008), numeroMeses)) e subtraia o total de taxas de corretagem do período (taxaCorretagemFixaMensal * numeroMeses), retornando o valor final (mínimo de 0). Para simularPassagemDeMes(), aplique a valorização (this.setSaldo(this.getSaldo() * (1 + 0.008))) e depois deduza a taxaCorretagemFixaMensal, garantindo que o saldo não fique negativo (mínimo 0).

3. **FundoInvestimento**: Também herdeira de Investimento, esta classe possuirá nomeFundo (String), cnpjGestora (String) e taxaAdministracaoAnual (double, ex: 0.02 para 2%) como atributos específicos. Ao implementar calcularSaldoProjetado(int numeroMeses), assuma um rendimento mensal bruto de 1.0% (0.01) e uma taxa de administração mensal (taxaAdministracaoAnual / 12). Calcule iterativamente, mês a mês para a projeção: saldoProjetadoMes = saldoProjetadoMesAnterior * (1 + 0.01); taxaAdmDoMesProjetado = saldoProjetadoMesAnterior * taxaAdministracaoMensal; saldoProjetadoMes -= taxaAdmDoMesProjetado. Retorne o saldo projetado final. Para simularPassagemDeMes(), calcule o rendimento bruto mensal (this.getSaldo() * 0.01), a taxa de administração devida no mês (this.getSaldo() * (this.taxaAdministracaoAnual / 12.0)) e atualize o saldo (this.setSaldo(this.getSaldo() + rendimentoBruto - taxaAdmDevida)), assegurando que não fique negativo.

4. **Debenture**: Este é um tipo de investimento que herda de Investimento e será exclusivo para clientes PessoaJuridica. Seus atributos específicos são nomeEmpresaEmissora (String), taxaJurosAnualDebenture (double), e percentualTributacaoPJ (double, ex: 0.20 para 20% sobre o rendimento). O construtor deve receber todos os dados necessários e **obrigatoriamente** verificar se o cliente associado é uma instância de PessoaJuridica. Caso contrário, deve lançar uma IllegalArgumentException. Para calcularSaldoProjetado(int numeroMeses), a lógica é similar ao Tesouro Prefixado, mas usando taxaJurosAnualDebenture e percentualTributacaoPJ. Calcule a taxa de juros mensal (taxaJurosAnualDebenture / 12), o saldo bruto projetado, o rendimento bruto, o imposto devido (rendimentoBruto * percentualTributacaoPJ), e retorne o saldo líquido projetado. Para simularPassagemDeMes(), calcule o rendimento do mês (this.getSaldo() * (this.taxaJurosAnualDebenture / 12.0)) e adicione este rendimento ao saldo. A tributação será considerada na projeção ou resgate, de forma similar aos outros produtos de renda fixa.

Finalmente, crie uma classe Principal (ou Main) para testar sua implementação. Nela, instancie objetos de PessoaFisica e PessoaJuridica. Para cada cliente, crie uma CarteiraInvestimentos. Em seguida, crie instâncias dos diferentes produtos de investimento e adicione-os à carteira de seu respectivo cliente, respeitando as restrições (ex: TesouroPrefixado para PessoaFisica, Debenture para PessoaJuridica). Demonstre as operações de aplicação e resgate. Simule a passagem de alguns meses e, após cada simulação, exiba o saldo individual de cada investimento e, mais importante, use o método calcularValorTotalInvestido() da carteira para mostrar o patrimônio total consolidado do cliente. Teste todas as validações, incluindo as de restrição de tipo de cliente.