ATIVIDADE FORMATIVA: Packages e modificadores de acesso

Daremos continuidade ao nosso **sistema de simulação de financiamentos imobiliários de um grande banco**. Neste momento, adaptaremos o código da atividade anterior para que contenha conceitos de pacotes e modificadores de acesso.

Lembramos que este projeto é **iterativo**. Portanto, usaremos como base o que você já desenvolveu na semana anterior.

E vale lembrar mais uma vez: **este projeto é individual em todas as suas fases e deve ser desenvolvido em Java**. As regras do curso em relação a prazos, locais de entrega e plágio também se aplicam a este trabalho.

O que devo desenvolver?

Veja a seguir um fluxo detalhado do funcionamento desejado para esta atividade:

1. Todos os requisitos da semana anterior.

2. Reorganize as classes que você criou em **pacotes** (*packages*), de acordo com a estrutura seguinte. Fique à vontade para modificar o nome dos *packages*.

    a. *Packages*

        i. modelo

            1. Classe:

                a. **Financiamento**

        ii. util

            1. Classe:

a. **InterfaceUsuario**

iii. main

1. Classe:

a. **Main**

3. Faça as seguintes mudanças quanto ao **nível de acesso**:

a. Classe **Financiamento** (no pacote **modelo**):

i. Todos os atributos devem ser **privados**.

ii. Todos os métodos devem ser **públicos**.

iii. Inclua um *getter* para cada um dos atributos **privados**.

iv. Crie um método **público** para mostrar na tela uma mensagem contendo os dados do financiamento, como o valor total do financiamento e o valor do imóvel.

b. Classe **InterfaceUsuario** (no pacote **util**):

i. Ajuste os métodos de entrada de dados (valor do imóvel, prazo de financiamento e taxa de juros) para que usem **estruturas condicionais** (como *if/else* ou *switch*) dentro dos seus métodos para verificar se as entradas fornecidas pelo usuário são válidas.

ii. Aceite somente valores positivos para o valor do imóvel, prazo do financiamento e taxa de juros anual.

iii. Use **estruturas de repetição** (como *do*, *do-while* ou *for*). Se algum dos valores for inválido, o programa deve informar ao usuário sobre o erro e solicitar que ele insira novamente os dados.

Informações adicionais

1. Teste por valores inválidos: o que acontecer com o seu código se alimentamos um valor do imóvel negativo, uma taxa de juros muito alta (como 100.000.000% por ano) ou um prazo de financiamento negativo?

2. Organize uma classe por arquivo: isto pode ser muito útil para testar e organizar o seu código. Também faça uso de **comentários** para que você mesmo saiba o que cada parte do código deve fazer.

3. Faça os exercícios: os exercícios e exemplos de cada unidade dão a base necessária para fazer essas atividades. Não se esqueça de resolvê-los caso esteja sentindo dificuldades.

Bons estudos!

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.