04

Criando o serviço e utilizando o delegate methods

Agora vamos criar a classe intermediaria que ficará com a responsabilidade de realizar o processo de validação antes de salvar um pagamento, crie a classe `ServicoPagamento` no pacote **br.com.alura.bytebank.servico** para representar um serviço de pagamentos e adicione um atributo do tipo `RegistroDePegamento` já instanciado.

A ideia do serviço é permitir que todos os clientes o utilize como interface para se comunicar com o meio persistente da aplicação, no caso, o `RegistroDePegamento` , portanto, precisamos replicar o comportamento da classe `RegistroDePegamento` dentro da classe `ServicoPagamento` .

Utilizando o delegate methods

Para isso, o IntelliJ nos fornece a feature conhecida como **Delegate methods** que já encapsula todos os comportamentos públicos de um objeto que queremos delegar o comportamento.

Essa opção fica disponível no **generate** (**Alt + Insert / Ctrl + N**) > **"Delegate Methods..."**. Basta apenas selecionar o objeto que pretende aplicar a feature, no caso vai ter apenas o objeto do tipo `RegistroDePegamento` .

Então selecione todos os métodos que deseja delegar. Observe que existem apenas 2 métodos possíveis. A possibilidade de métodos reflete na quantidade de métodos públicos que a classe possui, portanto, serão listados apenas todos os métodos que o objetivo tiver como público.

Já que a ideia do serviço é gerar essa interface entre clientes e o meio persistente, marque todos os métodos públicos, justamente para que não exista o motivo de usar a classe `RegistroDePegamento` . Então, clique em **OK**

Ajustando o código dentro do método delegado

Dentro do método `registra()` do serviço adicione o processo de validação chamando o método `verificaTipo` da classe `ValidaPagamento` , porém, faça isso antes de chamar o `registra()` da classe `RegistroDePegamento` .

> Lembre-se que a classe `ValidaPagamento` ainda não possui um método de validação que receba `pagamentos` , portanto, você pode optar por fazer um **foreach** dentro do `registra()` do `ServicoPagamento` e chamar a `verificaTipo()` ou fazer uma sobrecarga criando o `verificaTipo()` que recebe uma lista de pagamentos, então, dentro dele você faz o **foreach**. Pode decidir o que preferir, na aula foi feita a abordagem da sobrecarga.

Modificando pontos do código para usar o serviço

Com o serviço pronto, faça com que todas os pontos do projeto que estejam usando a classe `RegistroDePegamento` , com excessão do `ServicoPagamento` que é o único que deve usar, utilizem o `ServicoPagamento` no lugar.

Para isso você pode utilizar o **Find Usages** (**Alt + F7**) para verificar os pontos do código que utilizam o `RegistroDePegamento` . Então, quando encontrar, vá até a linha que os chama. Em seguida, utilize o **Type Migration** (**Ctrl + Shift + F6 / CMD + Shift + F6**) para modificar o tipo do objeto de `RegistroDePegamento` para `ServicoPagamento` .

> **Observações:** O **Type Migration** funciona apenas no tipo das variáveis, portanto, mesmo mudando as variáveis, caso for um objeto, a instância será mantida como a antiga, portanto, vai ter que ser modificada manualmente. Em outras

palavras, ele é mais efetivo em variáveis que não possuem uma atribuição local, como é o caso de parâmetros dos métodos.

Por fim, após ter realizado todo o processo de refatoração, execute o projeto e veja se tudo está funcionando como o esperado.