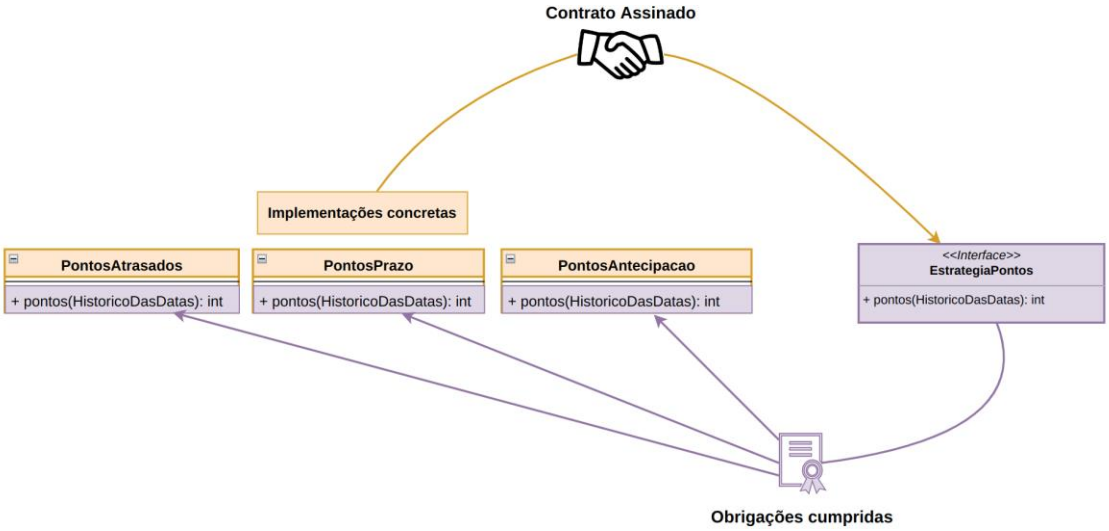
Interfaces – Java

Uma ***interface*** é de forma simples falando, um **Contrato**. Esse ***contrato*** tem a capacidade de transformar uma **Classe** em ***"algo mais"***, desde que **ela** **se comprometa a cumprir o que o Contrato pede**. De maneira simples, a **Interface** serve para **potencializar soluções específicas** através de **uma abstração genérica**.

Por exemplo, eu tenho uma **aplicação** que precisa **dar pontos** para a conclusão de uma Meta, dependendo da data de conclusão eu vou aplicar critérios de pontuação diferente, **antes do prazo**, **depois do prazo** e no **prazo correto.**

O que é **GENÉRICO** nessa **aplicação**: ***A ideia de dar pontos!***
O que é **ESPECÍFICO** nessa **aplicação: Como os pontos devem ser aplicados**

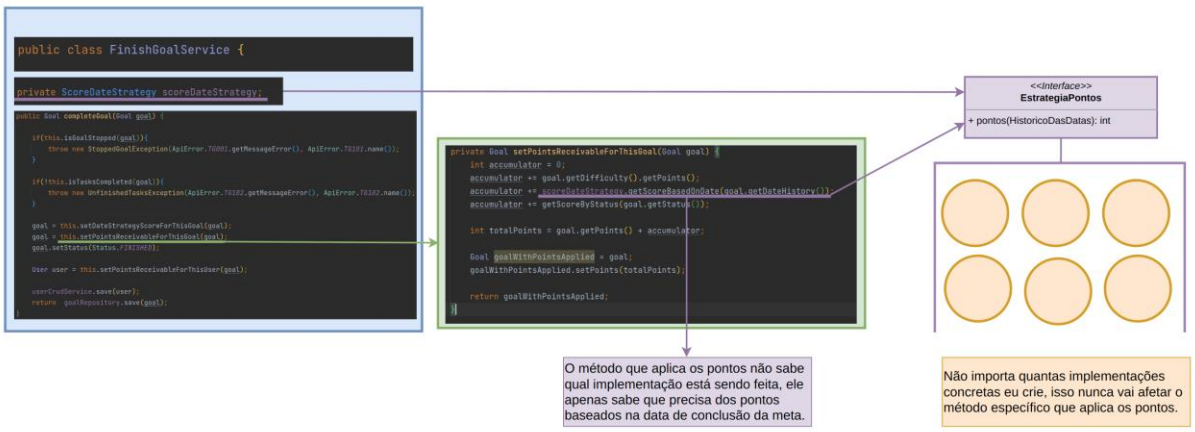


Como o código se beneficia dessa Abordagem – Java

Sem dúvida nenhuma, o principal benefício que uma Interface te proporciona é **a flexibilidade do código**. **Ela** tem um **significado muito mais semântico do que físico**, a **Interface** te diz o q**ue deve acontecer quando um comportamento específico dela for chamado**, não importa como você vai fazer isso, só faça.

Ainda com o **exemplo da aplicação que dá pontos para metas concluídas**, quando eu finalizo uma meta, minha **classe de serviço que representa a finalização** precisa aplicar os pontos da meta. Logicamente **ela precisa de alguma implementação que diga quantos pontos aplicar**, ***dependendo da data de conclusão real e a estipulada por quem criou a meta.***

O **método principal da classe de serviço não precisa saber qual critério de pontuação foi efetivamente aplicado**, ele **não precisa mudar mesmo que depois surjam diferentes critérios de pontuação, por que ele já usa a Interface:**



Uma coisa importante que precisa ser falada é a respeito da inicialização do objeto concreto da Interface. Só de usar uma interface nós já estamos aplicando um pouco do conceito de inversão de controle e injeção de dependência.

Com a **Interface** nós estamos dizendo que é responsabilidade de alguém de fora **injetar o Objeto concreto** dentro da nossa aplicação, mas o fato mesmo é que **CLASSE DE SERVIÇO PRECISA DO OBJETO CONCRETO DO TIPO DA INTERFACE INSTANCIADO!!!**

Sintaxe e Particularidades das Interfaces – Java

Toda interface é pública e a grande maioria das ***assinaturas de interface são abstratas*** (não tem implementação) e **todas são públicas. Interfaces** podem ***herdar*** de **outra Interface**, por exemplo a **JPARepository** é uma interface que ***carrega todos os comportamentos da hierarquia de Repository***: