Formulário de proposta de solução - SOLID com Java: Orientação a Objetos com Java

Vamos ter alguns cenários para serem resolvidos aqui e queremos saber os passos para sua solução em cada um dos cenários.   
  
Atualmente existe uma classe em um sistema dentro do Orange Talents com um método chamado analisaDadosResposta que recebe como argumento objeto cujo tipo é uma classe criada no próprio sistema para se conectar com formulários do google forms. Só que agora esse método precisa analisar dados que também podem vir do Microsoft Forms. No futuro, também pode ser necessário analisar dados que vão vir via TypeForms. Qual seria solução/alteração para que o método analisaDadosResposta não precisasse ser modificado em função das novas fontes de dados do futuro?  
Existe um outro método aqui nos sistemas da Orange Talent, analisaMomentoAtual, que recebe como argumento um objeto do tipo Aluno. Dentro deste método, o único método chamado a partir de um objeto do tipo Aluno, é o getCursosFeitos(). Neste momento, qualquer alteração na classe Aluno pode influenciar no comportamento do método analisaMomentoAtual. Como você faria para que o método analisaMomentoAtual possa continuar a receber um Aluno, mas através de uma interface mais estável? Só para deixar claro, a ideia é que o ponto do código onde temos algo como analisaMomentoAtual(aluno) continue do mesmo jeito. A invocação do método em si não precisa ser alterada. Isso é bem desafiador.

analisaDadosResposta

tem a classe recebe formulário

1

Qual seria solução/alteração para que o método analisaDadosResposta não precisasse ser modificado em função das novas fontes de dados do futuro?

Mesmo sendo fontes diferentes esses dados ainda continuam sendo um formulário, criaria uma interface formulário e implementaria no caso a classe para cada tipo de formulário, assim independente de quantidade de outras fontes tiver de formulário conseguimos implementar sem ferir de fato qualquer método que receba um formulário.

2

analisaMomentoAtual(Aluno){

}

que recebe como argumento um objeto do tipo Aluno



**SOLID com Java: Orientação a Objetos com Java**

|  |
| --- |
| **Cenário:** |
| Vamos ter alguns cenários para serem resolvidos aqui e queremos saber os passos para sua solução em cada um dos cenários.   1. Atualmente existe uma classe em um sistema dentro do Orange Talents com um método chamado **analisaDadosResposta**que recebe como argumento objeto cujo tipo é uma classe criada no próprio sistema para se conectar com formulários do google forms. Só que agora esse método precisa analisar dados que também podem vir do Microsoft Forms. No futuro, também pode ser necessário analisar dados que vão vir via TypeForms. Qual seria solução/alteração para que o método **analisaDadosResposta**não precisasse ser modificado em função das novas fontes de dados do futuro? 2. Existe um outro método aqui nos sistemas da Orange Talent, **analisaMomentoAtual,** que recebe como argumento um objeto do tipo Aluno. Dentro deste método, o único método chamado a partir de um objeto do tipo Aluno, é o **getCursosFeitos().**Neste momento, qualquer alteração na classe Aluno pode influenciar no comportamento do método **analisaMomentoAtual.**Como você faria para que o método **analisaMomentoAtual**possa continuar a receber um Aluno, mas através de uma interface mais estável? Só para deixar claro, a ideia é que o ponto do código onde temos algo como **analisaMomentoAtual(aluno)**continue do mesmo jeito. A invocação do método em si não precisa ser alterada. Isso é bem desafiador. |

|  |
| --- |
| **O que seria bom ver nessa resposta?** |

* **Peso 6:**Para o desafio de analisar os dados da resposta é preciso ter a criação de uma interface que define comportamentos necessários para cada implementação de recuperação dos dados do formulário.
* **Peso 4:**Para o desafio de gerar mais estabilidade para o método que analisa o momento atual do aluno, a solução imaginada aqui é a criação de uma interface específica que tenha apenas o método **getCursosFeitos().**A classe Aluno poderia implementar essa interface.

|  |
| --- |
| **Resposta do Especialista:** |
| * Criaria uma interface chamada, por exemplo, Form e criaria classes GoogleForm, TypeForm e MicrosoftForm. Todas implementariam a mesma interface. Dessa forma o método não precisa mudar para cada fonte de dados nova. Aqui trabalhamos com o princípio chamado Open Closed. No final é polimorfismo padrão. Importante ficar na cabeça que, em orientação a objetos, flexibilidade tem relação íntima com polimorfismo. De vez em quando isso também é chamado de binding dinâmico. * Para o caso do método que estava recebendo um aluno direto, poderia ser criada uma interface para expor apenas o método que retorna os cursos feitos. Aqui o uso da interface nem é por conta do polimorfismo e sim apenas para gerar estabilidade em cima de uma classe que é utilizada no sistema todo. Essa solução é inspirada pelo princípio da segregação pela interface. |