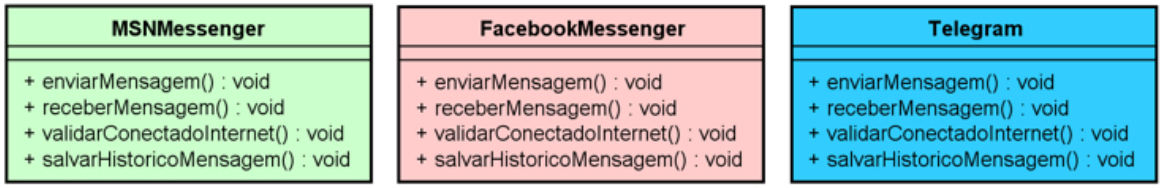
Herança

Nem tudo se copia, às vezes se herda.

Já imaginou, você ter sido classificado para a vaga de emprego de seus sonhos e como desafio, seria justamente você criar um diagrama de classes e em seguida os respectivos arquivos .java, que apresentasse os fundamentos de POO, com base no contexto de vários aplicativos de mensagens instantâneas? Sorte sua que você está nos acompanhando, em nossa jornada! 😜

Com base na nossa classe **MsnMessenger**, você já poderia dar os primeiros passos para se dar bem no processo seletivo, simplemente, copiar e colar a estrutura, para as novas classes **FacebookMessenger, Telegram** e **BINGO** 😁😁😁**!!!**

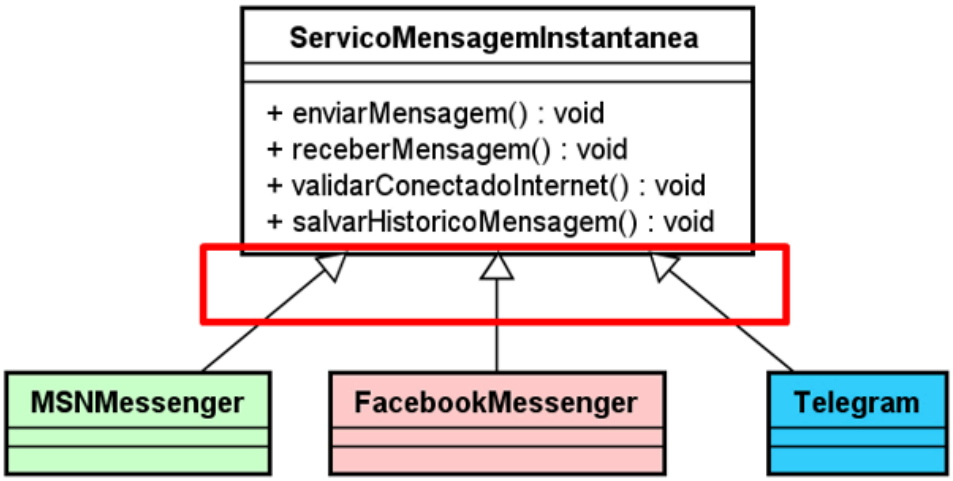


Agora é escrever o código das classes acima e esperar pela contratação !!!

Lamentamos dizer, mas esta não seria a melhor alternativa para a proposta citada acima.

Além de uma compreensão do desafio, é necessário que, tenhamos domínio dos pilares de POO e aplicá-los em situações iguais a esta.

NOTE: Todas as três classes, possuem a mesma estrutura comportamental e diante deste contexto, se encaixa perfeitamente o segundo pilar da POO, a Herança.



Representação UML do sistema de mensagens insntantâneas

ServicoPai | MSN | Facebook | Telegram | ComputadorPedrinho

```
//a classe MSNMessenger é ou representa
public class ServicoMensagemInstantanea {
    public void enviarMensagem() {
        //primeiro confirmar se esta conectado a internet
        validarConectadoInternet();
        System.out.println("Enviando mensagem");
        //depois de enviada, salva o histórico da mensagem
        salvarHistoricoMensagem();
    }
    public void receberMensagem() {
        System.out.println("Recebendo mensagem");
    }

    //métodos privadas, visíveis somente na classe
    private void validarConectadoInternet() {
        System.out.println("Validando se está conectado a internet");
    }
    private void salvarHistoricoMensagem() {
        System.out.println("Salvando o histórico da mensagem");
    }
}
```

Podemos avaliar a importância de compreender os pilares de POO, para ter uma melhor implementação, reaproveitamento e reutilização de código, em qualquer contexto das nossas aplicações, mas não para por ai.

> Será que todos os sistemas de mensagens, realizam as suas operações de uma mesma maneira? e agora ? este é um trabalho para os pilares **Abstração** e **Polimorfismo**.

Previous
Encapsulamento

Next
Abstração

Last updated 1 year ago