1. O que é GoF?

Os padrões de projeto GoF foram definidos pelos autores Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides no livro "Design Patterns: Elements of Reusable Object-Oriented Software". Esses padrões são soluções comprovadas para problemas comuns encontrados no desenvolvimento de software orientado a objetos. Eles são divididos em três categorias principais:

- **Criacionais**: Focam na criação de objetos.
- **Estruturais**: Tratam da composição de classes e objetos.
- **Comportamentais**: Definem a interação e responsabilidade entre objetos.

---

2. Os 23 Padrões de Projeto GoF

2.1 Padrões Criacionais

1. **Factory Method**: Permite a criação de objetos sem especificar a classe exata que será instanciada.
2. **Abstract Factory**: Cria grupos de objetos relacionados sem especificar suas classes concretas.
3. **Builder**: Constrói objetos complexos passo a passo.
4. **Prototype**: Cria novos objetos clonando um modelo existente.
5. **Singleton**: Garante que apenas uma instância de uma classe seja criada.

2.2 Padrões Estruturais

6. **Adapter**: Permite a comunicação entre interfaces incompatíveis.
7. **Bridge**: Separa uma abstração de sua implementação.
8. **Composite**: Permite trabalhar com estruturas hierárquicas de objetos como se fossem individuais.
9. **Decorator**: Adiciona funcionalidades dinamicamente a objetos.
10. **Facade**: Fornece uma interface simples para um sistema complexo.
11. **Flyweight**: Otimiza o uso da memória compartilhando objetos comuns.
12. **Proxy**: Atua como substituto ou intermediário para outro objeto.

2.3 Padrões Comportamentais

13. **Chain of Responsibility**: Permite que várias classes processem uma requisição sem especificar explicitamente o manipulador.
14. **Command**: Encapsula uma solicitação como um objeto, permitindo fila de solicitações e reversibilidade de ações.
15. **Interpreter**: Define uma gramática para interpretar expressões.
16. **Iterator**: Fornece uma maneira sequencial de acessar elementos de uma coleção sem expor sua representação interna.

17. **Mediator**: Define um objeto central para controlar a comunicação entre vários objetos.
18. **Memento**: Permite salvar e restaurar estados de objetos sem violar seu encapsulamento.
19. **Observer**: Permite que objetos sejam notificados automaticamente quando outro objeto é modificado.
20. **State**: Permite que um objeto altere seu comportamento quando seu estado muda.
21. **Strategy**: Permite selecionar um algoritmo em tempo de execução.
22. **Template Method**: Define um esqueleto de algoritmo e permite que subclasses implementem partes dele.
23. **Visitor**: Permite adicionar novas operações a uma estrutura de classes sem modificar essas classes.