Design Pattern **GRASP**

Desenvolvimento com qualidade com GRASP

Os padrões GRASP englobam uma série de princípios baseados em conceitos de Orientação a Objetos. Partindo de análises que procuram definir quais as obrigações dos diferentes tipos de objetos em uma aplicação, estes patterns disponibilizam uma série de recomendações que procuram favorecer a obtenção de sistemas mais bem estruturados. Partindo de práticas consagradas no desenvolvimento de sistemas orientados a objetos, os padrões GRASP procuram fornecer diretrizes para a construção de aplicações bem estruturadas e que possam ser facilmente adaptáveis diante da necessidade de mudanças. A consequência direta das recomendações propostas por estes patterns é um código mais bem organizado, de fácil manutenção e ainda, capaz de ser compreendido por diferentes desenvolvedores sem grandes dificuldades.

SOLID

O que é o **S.O.L.I.D**?

O S.O.L.I.D é um acrônimo que representa cinco princípios da programação orientada a objetos e design de código teorizados pelo nosso querido Uncle Bob (Robert C. Martin) por volta do ano 2000. O autor Michael Feathers foi responsável pela criação do acrônimo:

[S]ingle Responsibility Principle (Princípio da Responsabilidade Única) **[O]**pen/Closed Principle (Princípio do Aberto/Fechado) **[L]**iskov Substitution Principle (Princípio da Substituição de Liskov) **[I]**nterface Segregation Principle (Princípio da Segregação de Interfaces) **[D]**ependency Inversion Principle (Princípio da Inversão de Dependências).

GOF

Alguns programadores mais experientes começaram a perceber que os mesmos problemas começaram a aparecer várias e várias vezes e a solução para aqueles problemas eram sempre as mesmas e começaram a catalogar esses padrões. Em 1995 um grupo de pessoas, mas especificamente quatro pessoas escreveram um livro iniciando os Design Patterns mais conhecido de mercado, são eles Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides. Eles foram conhecidos como Gang of four ou GOF. Os Designs Patterns do GOF são classificados em três categorias: Criação, Estrutural e Comportamental. Na figura abaixo podemos ver os patterns com sua classificação.

Tipos de padrões **Criacional**:

Singleton: assegura que somente um objeto de uma determinada classe seja criado em todo o projeto;

Abstract Factory: permite que um cliente crie famílias de objetos sem especificar suas classes concretas;

Builder: encapsular a construção de um produto e permitir que ele seja construído em etapas;

Prototype: permite você criar novas instancias simplesmente copiando instancias existentes;

Factory Mathod: as subclasses decidem quais classes concretas serão criadas.

text: text

Estruturais

Decorator: envelopa um objeto para fornecer novos comportamentos;

Proxy: envelopa um objeto para controlar o acesso a ele;

FlyWeigth: uma instancia de uma classe pode ser usada para fornecer muitas “instancias virtuais”;

Facade: simplifica a interface de um conjunto de classes;

Composite: Os clientes tratam as coleções de objetos e os objetos individuais de maneira uniforme;

Bridge: permite criar uma ponte para variar não apenas a sua implementação, como também as suas abstrações;

Adapter: envelopa um objeto e fornece a ele uma interface diferente;

Comportamental

Template Method: As subclasses decidem como implementar os passos de um algoritimo;

Visitor: permite acrescentar novos recursos a um composto de objetos e o encapsulamento não é importante;

Command: encapsula uma solicitação como um objeto;

Strategy: encapsula comportamentos intercambiáveis e usa a delegação para decidir qual deles será usado;

Chair of Responsability: permite dar a mais de um objeto a oportunidade de processar uma solicitação;

Iterator: fornece uma maneira de acessar seqüencialmente uma coleção de objetos sem expor a sua implementação;

Mediator: centraliza operações complexas de comunicação e controle entre objetos relacionados;

Memento: permite restaurar um objeto a um dos seus estados prévios, por exemplo, quando o usuário seleciona um “desfazer”;

Interpreter: permite construir um intérprete para uma linguagem;

State: encapsula comportamentos baseados em estados e usa a delegação para alternar comportamentos;

Observer: permite notificar outros objetos quando ocorre uma mudança de estado.

Padrões de Objetos: Template Method, Mediator, Iterator, Visitor, Memento, Interpreter, Strategy, Command, Chair of Responsability, Observer, State.

Padrões de classe: Factory Method, Adapter, Template Method, Interpreter.