OOD

Object Oriented Design

Introdução

- É a disciplina de definir os objetos e suas interações para resolver um problema que foi identificado e documentado durante a Análise Orientada a Objeto
- É um método de design que engloba o processo de decomposição orientada a objetos e uma notação para representar modelos lógicos e físicos, bem como estaduais e dinâmicos, do sistema a ser desenvolvido.

Conceitos

O Design Orientado a Objeto é permeado e guiado por diversos conceitos que guiam sua execução. Estes conceitos são:

- Abstração
- Objeto
- Encapsulamento
- Herança
- Interface
- Polimorfismo

Princípios

- Os padrões SOLID são considerados uma boa prática de programação orientada a objetos;

- Onde, aplicados ao design otimizam a aplicação do orientado objeto;

- A preocupação do SOLID está no acoplamento e coesão;

Princípios

- A sigla SOLID é um acrônimo dos cinco princípios básicos do orientado objeto:

    - S: O “S” é de SRP (**S**igle Responsability Principle) Princípio de Responsabilidade Única, que afirma que uma classe deve possuir apenas um tipo de responsabilidade de atuação;

    - O: OCP (**O**pen Close Principle) Princípio Aberto-Fechado, que diz que entidades de software (classes, métodos, módulo, etc) devem estar abertas para extensão, mas fechadas para modificação;

Princípios

- A sigla SOLID é um acrônimo dos cinco princípios básicos do orientado objeto:

    - L: é de LSP (**L**iskov substitution Principle), que de forma mais básica afirma que classes derivadas devem poder ser substituídas por suas classes base

    - I: ISP (**I**nterface segregation principle). O Princípio da Segregação da Interface trata da coesão de interfaces e diz que clientes não devem ser forçados a depender de métodos que não usam.

Princípios

- A sigla SOLID é um acrônimo dos cinco princípios básicos do orientado objeto:

    - D: DIP(**D**ependency inversion principle). O Princípio da inversão da dependência afirma que deve-se depender de uma abstração em vez de uma implementação, para isso deve-se fazer com que módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações, e abstrações não devem depender de detalhes. Detalhes devem depender de abstrações.

Entradas para DOO

As entradas para o projeto orientado a objetos é fornecida pela saída da análise orientada a objetos

Alguns artefatos de entrada típicos para DOO são:

- Modelo Conceitual
- Casos de Uso
- Diagrama de Sequência do Sistema

Design do Sistema

- O DOO de um sistema envolve um contexto e o projeto da arquitetura do sistema.

- O contexto é subdividido em estático (visão do diagrama de pacotes UML) e dinâmico (visão dos casos de uso).

- O projeto da arquitetura baseia-se em princípios arquitetônicos e de domínio do software. Normalmente particiona-se o sistema em camadas.

Design do Sistema

- Etapas do Design do Sistema:

- Decomposição Orientada a Objetos
- Identificar Concorrência
- Identificar Padrões
- Controlar Eventos
- Gerenciar Condições de Fronteira

Design do Objeto

- Após a hierarquia de subsistemas ter sido desenvolvida, os objetos no sistema são identificados e seus detalhes são projetados

- A ênfase muda de conceitos de domínio do software para conceitos computacionais.

Design do Objeto

- Etapas do Design do Objeto:

- Identificação do Objeto
- Representação do Objeto
- Classificação das Operações
- Design do Algoritmo
- Design dos Relacionamentos

Otimização do Design

- Após o design ser feito e antes de sua implementação
- Tem como foco, minimizar o custo no desenvolvimento.
- Excesso na otimização deve ser evitado. O projetista deve buscar um equilíbrio entre otimização e compreensibilidade.

Técnicas de otimização

- Adição de associações redundantes nos modelos
- Omissão de associações não utilizáveis
- Otimização de algoritmos
  - Rearranjo das tarefas computacionais;
  - Reversão da ordem de execução dos laços definidos no modelo funcional.
- Salvar e armazenar atributos derivados.

Documentação

- Áreas de uso:
  - Design de software desenvolvido por um grupo de desenvolvedores;
  - Estratégias de desenvolvimento de software interativas;
  - Desenvolvimento de versões subsequentes de um software;
  - Encontrar condições e áreas de teste;
  - Manutenção de software.

Documentação

- Conteúdo
  - Arquitetura do sistema em alto nível;
  - Abstrações e mecanismos-chave;
  - Ilustrações do comportamento dos aspectos principais do software.

- Características
  - Concisa, consistente e completa;
  - Sem ambiguidades;
  - Relacionável às especificações de requisitos;
  - Bem estruturada;
  - Mais diagramática e menos descritiva.

Saídas do DOO

- Diagrama de Sequência:
  - Representando a sequência de processos num programa de computador.
  - Representa as mensagens trocadas entre processos ou objetos de uma forma simples e lógica.
- Diagrama de Classes:
  - Descreve a estrutura de um sistema, mostrando as classes do sistema, seus atributos e os relacionamentos entre as classes.
  - É usado para modelagem conceitual geral da sistemática da aplicação e para modelagem detalhada.

Conclusão

Por quê DOO?

Sempre que iniciasse o desenvolvimento de um software, busca-se a resolver algum problema ou uma situação do mundo real. Por conta disso, a missão de representar essa solução de forma a ser entendida por um computador se torna complexa e onerosa. Porém, o mercado sempre busca representar estes cenários da forma mais detalhada e próxima da realidade possível. Sendo assim, faz-se necessária a utilização de métodos que facilitem esta representação. O Design Orientado a Objetos tem como base conceitos como abstração, encapsulamento, decomposição, generalização e composição, onde todos estes conceitos contribuem para uma melhor representação dos cenários da vida real.