Aluno: Matheus Alencar de Medeiros Lucena

1 -

Na fase de análise, é compreendido o software como um todo, com especificações do que ele deve fazer e o que deve ser feito com ele sem se preocupar com como o software será feito. Nessa etapa é feito um estudo detalhado dos requisitos levantados que devem ser validados e verificados para a próxima fase.
Na fase de projeto é definido como o software realizará as funções definidas. Nessa fase os requisitos não funcionais são especificados (arquitetura do sistema, linguagem, etc). As classes são criadas e as dependências são estabelecidas visando a realização das funcionalidades descritas.

2 -

Abstração: A abstração é a capacidade de transformamos do mundo real em mundo virtual, através de uma representação estabelecendo identidade, atributos e métodos e isso é um ponto positivo pois é entendido pelo mundo computador uma entidade real.

Reuso de código: Nas linguagens Orientadas a Objeto, o reuso do código é uma realidade e permite que o mesmo código seja utilizado diversas vezes sem precisar sempre escrevê-lo. A herança ajuda ao reuso de código ser real.

Leitura e manutenção do código: A representação do sistema através da abstração se assemelha ao mundo real, fazendo com que o código seja mais legível e mais fácil de mantê-lo.

Criação de bibliotecas: As bibliotecas representadas por classes em linguagens orientadas a objeto são claras e permitem reutilização fácil.

3 -

Os requisitos funcionais são basicamente as funcionalidades que o software deve oferecer ao usuário. São utilizados para a criação das classes, das dependências entre elas e das decisões tomadas quanto ao software. Na descrição dos requisitos deve ter o que o usuário precisa que o software tenha sem a necessidade da preocupação de como será implementado.

4 -

5 -

O iterator serve para percorrer uma estrutura de dados e acessar um elemento dessa estrutura. O iterator é usado para estruturas de dados como Sets e Maps e é inteligente por possuir os métodos .next() e .hasNext().

6 -

7 -

8 -

9 -

O polimorfismo ajuda ao usuário a ler o código de forma simples, além da capacidade de ignorar detalhes específicos de objetos e manipulá-los uniformemente. Outra vantagem é a Escalabilidade, que permite que classes sejam adicionadas a uma familia existente mas execute código próprio independente da super classe (classe mãe).

O acoplamento causado pelo polimorfismo através da herança é descartável e ele necessita da interface definida.

10 -

11 -

Herança de tipo diz respeito a herança no qual classes herdam interfaces e são referenciadas por outros tipos, como “ContaBancaria conta = new ContaCorrente();”

Herança de implementação é a que herda todo o código da classe pai de uma só vez, sem a necessidade de reescrita ou sobre-escrita de código. Diz respeito a reutilização de código.

12 -

A principal desvantagem é que, caso uma classe seja modificada, pode acontecer erro em todas as classes que dependem dessa modificada. As alterações realizadas custam muito e demandam muito tempo. A reutilização do código fica mais difícil, pois torna-se necessário criar outras dependências.

13 -

14 -

Segundo o princípio da Orientação a Objetos, cada classe deve uma única responsabilidade para evitar que classes possuam responsabilidades que não precisava ter. Em um sistema grande, se existirem muitas classes com responsabilidades mútuas, o sistema torna-se difícil de manutenção. O melhor a se fazer é criar classes, cada uma com apenas uma responsabilidade, e deve fazer bem.

15 -