EXERCICIO 01

1. Analise: um estudo detalhado dos dados adquiridos, analise dos requisitos levantados, ou seja, utilizar as necessidades dos clientes, depois de compreendido o problema, para resolução do problema solicitado.
   Projeto: Como o sistema funcionará, como os requisitos do cliente serão atendidos.
2.
   - Maior facilidade para reutilização de código.
   - Nível mais elevado de abstração.
   - Maior adequação à arquitetura cliente/servidor.
   - Maior facilidade de comunicação com os usuários e com outros profissionais de informática.
   - Ciclo de vida mais longo para os sistemas.
   - Desenvolvimento acelerado de sistemas.
   - Possibilidade de se construir sistema muito mais complexos.
   - Menor custo para desenvolvimento e manutenção de sistemas.
3. Para saber as exigências, solicitações, desejos, necessidades do sistema que deverá realiza-las. O detalhamento das funções que ela exigirá.
4.
5. Iterator é um padrão para percorrer listas, conjuntos, mapas etc. Ele melhorar a desempenho para as buscas, ou seja, ele busca e remove objeto mais rápido.
6.
7. Return: é usado para retornar um valor esperado ou não esperado que possa ser reutilizado.
   Throw: é usado para mostra um erro esperado ou não do sistema.
8.
9. Vantagens: permite vários objetos de um mesmo tipo base sejam tratados da mesma maneira, aumentar um sistema de maneira controlada e mais localizada, reutilização de código.
   Desvantagens:

10. Pois na interface ela define ações que devem ser obrigatoriamente executadas, mas cada classe pode executar de forma diferente. Já Classes abstratas elas obrigam outras classes a utilizar os seus métodos. Ou seja, interface da mais liberdade no código do que a das classes abstratas.
11. Herança de tipo: é por meio de interfaces.
    herança de implementação: é por meio de classes normais ou abstratas.
12. Vantagens: Mudança na classe não altera o resto do código ou sistema, o código é mais simples.
    Desvantagens: Mudanças externas impõem mudanças a uma classe interna, o código e difícil de entender, o reuso e comprometido.
13. Na decomposiçao, os metodos menores que possuem caracteristicas exclusivas, diminui a complexidade, assim esconde algumas caracteristica não necessarias para o entendimento do metodo.

14. O que cada classe pode fazer, a sua obrigação. Pois assim o abaixa o acoplamento da classe. Mais poderia arruinar seu código, se você mudar algo no seu código ele pode parar de funcionar.
15. Pois em herança cada classe tem sua responsabilidade independente da outra, já em composição tem responsabilidade em tudo.