ES3A3 – Exercícios
Aula 9 – Padrões de Projeto

1. (1,0 ponto) Por que um bom desenvolvedor deve ao menos conhecer os principais padrões de projeto existentes no mercado?

R: Deve conhecer porque esses padrões proporcionam soluções para cenários que já tiveram resolução. Ou seja, não há a necessidade de "recriar a roda", já que existem referências prévias. Além disso, o conhecimento e aplicação desses conceitos proporcionam uma melhor qualidade do software; economia de tempo, já que ele pode usar um padrão já existente e etc.

2. (1,0 ponto) Para aplicar os padrões de projeto adequadamente, aproveitando os benefícios de cada solução, basta conhecer todos os padrões existentes atualmente? Justifique sua resposta.

R: Acredito que não. Afinal, não basta conhecer os padrões, obviamente é importante sabê-los, mas sim entendê-los e saber o cenário ideal na qual serão aplicados adequadamente. Aliás, certamente exigira um razoável nível de conhecimento/experiência do profissional.

3. (2,0 pontos) Faça uma pesquisa e descreva em poucas linhas cada um dos 23 padrões GoF mencionados na aula, exceto os padrões Observer, Singleton e Factory já detalhados na apresentação da aula.

1. Padrão Adapter: permite que objetos com interfaces incompatíveis trabalhem juntos. Ele converte a interface de uma classe em outra que o cliente espera encontrar.

2. Padrão Bridge: separa uma abstração da sua implementação, permitindo que elas possam variar independentemente.

3.Padrão Builder: separa a construção de um objeto complexo de sua representação, permitindo que o mesmo processo de construção possa criar diferentes representações.

4.Padrão Chain of Responsibility: evita acoplamento entre remetentes e destinatários de uma solicitação, dando a vários objetos a oportunidade de lidar com a solicitação.

5.Padrão Command: encapsula uma solicitação em um objeto, permitindo que você faça solicitações diferentes e agende ou registre solicitações.

6. Padrão Composite: permite que você construa estruturas de objetos em árvores e possa tratar objetos individuais e composições de objetos de forma uniforme.

7. Padrão Decorator: permite que você adicione comportamentos a objetos individuais sem afetar outros objetos da mesma classe.

8. Padrão Facade: fornece uma interface simplificada para um conjunto complexo de classes, bibliotecas e serviços.

9. Padrão Flyweight: usa compartilhamento para suportar grandes quantidades de objetos de forma eficiente.

10. Padrão Interpreter: define uma representação gramatical para um idioma, além de um interpretador para interpretar sentenças nessa linguagem.

11. Padrão Iterator: fornece uma maneira de acessar sequencialmente os elementos de uma coleção sem expor sua representação subjacente.

12. Padrão Mediator: permite que você reduza a complexidade ao limitar a comunicação entre objetos a um objeto mediador.

13. Padrão Memento: permite que você capture o estado de um objeto e o restaure posteriormente sem violar o encapsulamento.

14. Padrão Proxy: fornece um objeto de substituição para outro objeto para controlar o acesso a ele.

16. Padrão Template Method: define a estrutura de um algoritmo em uma superclasse e permite que as subclasses substituam etapas específicas do algoritmo sem mudar sua estrutura.

17. Padrão Visitor: permite que você adicione novas operações a um conjunto de classes sem mudar essas classes.

18. Padrão Abstract Factory: fornece uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.

19. Padrão Prototype: permite que você crie novos objetos a partir de um modelo existente, evitando a criação de subclasse para cada objeto a ser criado.

20. Padrão Strategy: define uma família de algoritmos, encapsula cada um deles e os torna intercambiáveis. Isso permite que o algoritmo varie independentemente dos clientes que o usam.

Padrão State: permite que um objeto altere seu comportamento quando seu estado interno muda. O objeto parecerá ter mudado de classe.

4. (6,0 pontos) Considerando os exemplos apresentados em aula, use a linguagem Java para implementar outros três exemplos de aplicação dos padrões de projeto Observer, Singleton e Factory.