SSC0130 - Trabalho 2

- **Gustavo Siqueira Barbosa - 10728122**
- **Luiz Fernando Santos - 10892680**

1. (ENADE 2011, adaptado) Sobre padrões de projeto, assinale V ou F.

- (F) Prototype é um tipo de padrão estrutural. Prototype é criacional.
- (F) Singleton tem por objetivo garantir que uma classe tenha ao menos uma instância e fornecer um ponto global de acesso para ela. Singleton garante que uma classe tenha no máximo uma instância, podendo não ser instanciada também.
- (V) Template Method tem por objetivo definir o esqueleto de um algoritmo em uma operação, postergando a definição de alguns passos para subclasses. Definimos um esqueleto geral a ser implementado posteriormente pelas classes que a herdam.
- (F) Iterator fornece uma maneira de acessar sequencialmente os elementos de um objeto agregado sem expor sua representação subjacente.

---

2. Dê o nome dos seguintes padrões de projeto:

- Oferece uma interface unificada e de alto nível que torna mais fácil o uso de um sistema.
  - Fachada
- Garante que uma classe possui, no máximo, uma instância e oferece um ponto único de acesso a ela.
  - Singleton
- Facilita a construção de objetos complexos com vários atributos, sendo alguns deles opcionais.
  - Builder
- Converte a interface de uma classe para outra interface esperada pelos clientes. Permite que classes trabalhem juntas, o que não seria possível devido à incompatibilidade de suas interfaces.
  - Wrapper
- Oferece uma interface ou classe abstrata para criação de uma família de objetos relacionados.
  - Abstract factory
- Oferece um método para centralizar a criação de um tipo de objeto.
  - Factory
- Funciona como um intermediário que controla o acesso a um objeto base.
  - Proxy
- Permite adicionar dinamicamente novas funcionalidades a uma classe.

  - Decorator
- Oferece uma interface padronizada para caminhar em estruturas de dados.
  - Iterator
- Permite parametrizar os algoritmos usados por uma classe.
  - Strategy
- Torna uma estrutura de dados aberta a extensões, isto é, permite adicionar uma função em cada elemento de uma estrutura de dados, mas sem alterar o código de tais elementos.
  - Visitor
- Permite que um objeto avise outros objetos de que seu estado mudou.
  - Observer
- Define o esqueleto de um algoritmo em uma classe base e delega a implementação de alguns passos para subclasses.
  - Template Method

3. Dentre os padrões de projeto que respondeu na questão (2), quais são padrões criacionais?

- Factory, Abstract Factory, Singleton e Builder

5. Qual a semelhança entre Proxy, Decorador e Visitor? E qual a diferença entre esses padrões?

- **Semelhanças**
  - Os três padrões se tratam de estratégias que visam facilitar a implementação de novos métodos em objetos por meio de classes intermediárias, mantendo o encapsulamento das classes.
- **Diferenças**
  - Decorator é útil quando temos uma classe com várias características opcionais que podem ou não serem implementadas sobre um objeto base, adicionando as características conforme necessário.
  - Proxy, além de implementar novas funcionalidades, controla o acesso a um objeto, restringindo o que o cliente pode ou não fazer com determinada funcionalidade.
  - Visitor é útil quando temos um método que queremos implementar em uma família de objetos, mas suas naturezas distintas dificultam a implementação. Desta forma, este padrão é utilizado quando temos uma hierarquia e queremos implementar um determinado método que se comporta de maneira diferente em cada uma delas.

7. Suponha uma classe base A. Suponha que queremos adicionar quatro funcionalidades opcionais F1, F2, F3 e F4 em A. Essas funcionalidades podem ser adicionadas em qualquer ordem, isto é, a ordem não é importante. Se usarmos herança, quantas subclasses de A teremos que implementar? Se optarmos por uma solução por meio de decoradores, quantas classes teremos que implementar (sem contar a classe A). Justifique e explique sua resposta.

Sabendo que a ordem das funcionalidades não deve importar e elas são opcionais, podemos obter o total de subclasses que deveriam ser criadas fazendo a combinação

$$1 + \frac{4!}{3!} + \frac{4!}{(2!2!)} + \frac{4!}{3!}$$

de modo que teríamos 15 subclasses. Utilizando decoradores, por outro lado, criaríamos apenas uma subclasse para cada funcionalidade, que seriam por fim usadas como uma composição para obter as demais combinações, totalizando assim 4 subclasses.

10. Suponha a API de Java para E/S. Suponha que para evitar o que chamamos de paternite, você fez a união das classes `FileInputStream` e `BufferedInputStream` em uma única classe. Como discutimos na Seção 6.13, o mecanismo de buffer será ativado por default na classe que você criou. Porém, como você tornaria possível desativar buffers nessa nova classe, caso isso fosse necessário?

Um parâmetro booleano que determine a ativação ou desativação do buffer pode ser introduzido na inicialização da classe. Alternativamente, pode ser adicionado um novo método que ative ou desative o buffer após a inicialização da classe.

11. Suponha o exemplo de Visitor que usamos na Seção 6.11. Especificamente, suponha o seguinte código, mostrado no final da seção.

```
PrintVisitor visitor = new PrintVisitor();

foreach(Veiculo veiculo: listaDeVeiculosEstacionados) {
  veiculo.accept(visitor);
}
```

Suponha que `listaDeVeiculosEstacionados` armazene três objetos: `umCarro`, `umOnibus` e `umOutroCarro`. Desenhe um diagrama de sequência UML que mostre os métodos executados por esse trecho de código (suponha que ele é executado por um objeto main).