

Atividade 5 _ DESIGN PATTERNS_

Aluno: Caio Gomes Alcântara glória

Caio Gomes Alcântara glória - 763989
1. Samuel Luiz da Cunha - 762496

Deseja-se implementar uma solução para o desenvolvimento de um display, o qual mostra o estado de um objeto X a partir do objeto em si, utilizando-se padrão de projeto. Displays alternativos são fornecidos, e a necessidade principal é que todos eles sejam automaticamente notificados e atualizados para refletir a mudança ocorrida quando o estado do objeto mudar.

Qual padrão para resolver esse problema?

☐ Adapter

☐ State

☒ Observer

☐ Iterator

2.

Assinale a opção que apresenta o padrão de projeto que tem por objetivo separar o display de estado de um objeto a partir do objeto em si e que permite que sejam fornecidos displays alternativos.

☐ State

☐ Facade

☒ Decorator

☐ Iterator

3.

Em relação a padrões de projeto de software, assinale a afirmativa incorreta.

☐ Builder é um padrão utilizado quando se deseja separar a construção de um objeto complexo de sua representação de modo que o mesmo processo de construção possa criar diferentes representações.

☐ Adapter é um padrão utilizado quando se deseja converter a interface de uma classe em outra interface, esperada pelos clientes.

☒ Singleton é um padrão utilizado quando se deseja compor objetos em estrutura de árvore para representarem hierarquias partes-todo.

☐ Proxy é um padrão também conhecido como surrogate utilizado quando se deseja fornecer um substituto ou marcador da localização de outro objeto para controlar o acesso ao mesmo.

4.

Os padrões de projeto orientados a objeto podem ter finalidade de criação, estrutural ou comportamental. Os padrões de criação se preocupam com o processo de criação de objetos. Os padrões estruturais lidam com a composição de classes ou de objetos. Os padrões comportamentais caracterizam as maneiras pelas quais classes ou objetos interagem e distribuem responsabilidades. Assinale a alternativa que apresenta apenas padrões de projeto comportamentais.

☒ Mediator, Interpreter e Command.

☐ Proxy, Builder e Mediator.

☐ Singleton, Composite e Interpreter.

☐ Prototype, Abstract Factory e Builder.

5.

Padrões de projeto são soluções reutilizáveis de software orientado a objetos.

Considere as duas afirmativas a seguir e depois informe a alternativa correta.

I - Padrões de criação definem de quem é a responsabilidade de instanciar objetos.

II - Padrões de projeto se preocupam com algoritmos e a atribuição de responsabilidades entre objetos.

☒ As duas afirmativas estão corretas

☐ Apenas a afirmativa I está correta

☐ Apenas a afirmativa II está correta

☐ Nenhuma está correta

6.

Assinale a opção correta com referência aos padrões comportamentais GoF e aos padrões GRASP.

☒ O padrão SINGLETON pode ser um anti-padrão

Padrões comportamentais distribuem responsabilidade pela instanciação de objetos entre seus participantes, EXATAMENTE da mesma forma que nos padrões GRASP. ☒

☐ Em padrões comportamentais, toda a distribuição de comportamento entre classes é feita por meio de delegação. ☒

☐ Todo CONCRET ITERATOR que participa de uma aplicação do padrão ITERATOR está associado a um SINGLETON ☒

7.

Os padrões BRIDGE e FLYWEIGHT são:

☒ Padrões GoF estruturais

☐ Padrões GRASP avançados

☐ Padrões GoF comportamentais

☐ Padrões GoF estruturais

8.

Deseja-se implementar uma solução para o desenvolvimento de um display, o qual mostra o estado de um objeto X a partir do objeto em si, utilizando-se padrão de projeto. Displays alternativos são fornecidos, e a necessidade principal é que todos eles sejam automaticamente notificados e atualizados para refletir a mudança ocorrida quando o estado do objeto mudar.

Qual padrão para resolver esse problema?

☒ Observer

☐ State

☐ Adapter

9.

Diante da crescente demanda por automatização de processos de negócio, o gerente de desenvolvimento de sistemas de informação busca a maximização do reuso de software. A abordagem de reuso que utiliza abstrações genéricas, não incluindo detalhes de implementação, que mostram objetos abstratos e concretos e interações, é:

☒ design pattern;

desenvolvimento baseado em componentes;

☐ framework de aplicação;

bibliotecas de programas.

10.

Assinale a opção que apresenta o padrão de projeto que tem por objetivo separar o display de estado de um objeto a partir do objeto em si e que permite que sejam fornecidos displays alternativos.

☒ Decorator

Iterator

☐ Facade

State

11.

Em relação a padrões de projeto de software, assinale a afirmativa incorreta.

☒ Singleton é um padrão utilizado quando se deseja compor objetos em estrutura de árvore para representarem hierarquias partes-todo.

Proxy é um padrão também conhecido como surrogate utilizado quando se deseja fornecer um substituto ou marcador da localização de outro objeto para controlar o acesso ao mesmo.

☐ Adapter é um padrão utilizado quando se deseja converter a interface de uma classe em outra interface, esperada pelos clientes.

Builder é um padrão utilizado quando se deseja separar a construção de um objeto complexo de sua representação de modo que o mesmo processo de construção possa criar diferentes representações.

12.

Em relação aos padrões de projeto de software assinale a alternativa correta.

☒ Factory Method é um padrão utilizado quando se deseja definir uma interface para criar um objeto e deixar as subclasses decidirem que classe instanciar.

Singleton é um padrão utilizado quando se deseja separar a construção de um objeto complexo de sua representação de modo que o mesmo processo de construção possa criar diferentes representações.

Adapter é um padrão utilizado quando se deseja desacoplar uma abstração de sua implementação, de modo que as duas possam variar independentemente.

Proxy é um padrão utilizado quando se quer garantir que uma classe tenha somente uma instância e fornecer um ponto global de acesso a mesma.

13.

Os padrões de projeto orientados a objeto podem ter finalidade de criação, estrutural ou comportamental. Os padrões de criação se preocupam com o processo de criação de objetos. Os padrões estruturais lidam com a composição de classes ou de objetos. Os padrões comportamentais caracterizam as maneiras pelas quais classes ou objetos interagem e distribuem responsabilidades. Assinale a alternativa que apresenta apenas padrões de projeto comportamentais.

- ☒ Mediator, Interpreter e Command.
- ☐ Proxy, Builder e Mediator.
- ☐ Singleton, Composite e Interpreter.
- ☐ Prototype, Abstract Factory e Builder.

14.

O Instituto Nacional do Clima (INC) utilizará uma biblioteca de simulações de clima fabricada por uma empresa americana. Porém, após a aquisição, percebeu-se que as interfaces disponibilizadas pelas classes dessa biblioteca são incompatíveis com as interfaces das classes de outros sistemas do INC. A maneira correta de contornar esse problema é a utilização do padrão de projetos.

- ☒ adapter.
- ☐ singleton.
- ☐ memento.
- ☐ decorator.

15. Troque a implementação do programa abaixo para que a classe Incremental seja Singleton:

```
class Incremental {
    private static int count = 0;
    private int numero;

    public Incremental() {
        numero = ++count;
    }

    public String toString() {
        return "Incremental " + numero;
    }
}

public class TesteIncremental {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            Incremental inc = new Incremental();
            System.out.println(inc);
        }
    }
}
```

16.

Escreva um programa que conte até 10 e envie os números para uma ferramenta de log. Esta ferramenta de log deve ser construída por uma fábrica. Utilize Factory Method para permitir a escolha entre dois tipos de log: em arquivo (log.txt) ou diretamente no console. A escolha deve ser por um parâmetro passado ao programa ("arquivo" ou "console").

15) Public Class Incremental {

Private static Incremental instance;

Private static int count = 0;

Private int number;

Private Incremental() {

number = ++ count;

}

Public static Incremental getInstance() {

if (instance == null) {

instance = new Incremental();

return instance;

@Override

Public String toString() {

return "Incremental" + number;

}

main ... For (int i = 0; i < 10; i++) {

Incremental inc = Incremental.getInstance();

System.out.println (inc); }

16)

→ Página Anterior Para volver =>

16) bibliotecas Arquivo ebuffer

```
...
Public Class logger {
    Private static final String LOG_File = "log.txt";
    Public static void log (String palavra) {
        try (bufferedWriter writer = new bufferW (new FileW(1));
        {
            writer.write (palavra);
            writer.newLine();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

∴ main ... for (int i = 1, i <= 10; i++) {
    logger.log ("Contador: " + i);
}
```