PROVA TECNICAS DE PROGRAMACAO

NOME: RUBENS DIRCEU ORTEGA JUNIOR

LINK GIT HUB: Prova-2-Design-Paterns/PROVA TECNICAS DE PROGRAMACAO.pdf at main · rubinhortega/Prova-2-Design-Paterns (github.com)

QUESTÕES

1) Considere os seguintes Designs Patterns e Princípios de Engenharia de Software
   - Design Patterns GoF
   - Design Patterns Táticos e Estratégicos - DDD
   - Princípios SOLID

   Explique cada um dos Designs Patterns e Princípios acima (3,0 pontos)

   - Design Patterns GoF - São classificados em três categorias: Criacional, Estrutural e Comportamental.

   - Design Patterns Táticos e Estratégicos – DDD - Durante a fase estratégica do DDD (design controlado pelo domínio), você está mapeando o domínio de negócios e definindo contextos limitados para seus modelos de domínio. DDD tático é quando você define os modelos de domínio com mais precisão. Os padrões táticos são aplicados dentro de um único contexto limitado**.**

   - Princípios SOLID - Os princípios SOLID são cinco princípios de design de código orientado à objeto que basicamente tem os seguintes objetivos: Tornar o código mais entendível, claro e conciso; Tornar o código mais flexível e tolerante a mudanças; Aumentar a adesão do código aos princípios da orientação a objetos.

2) Indique e relacione cada um dos 22 Design Patterns em suas categorias. Use a Tabela abaixo para complementar e explicar cada um deles (3,0 pontos)

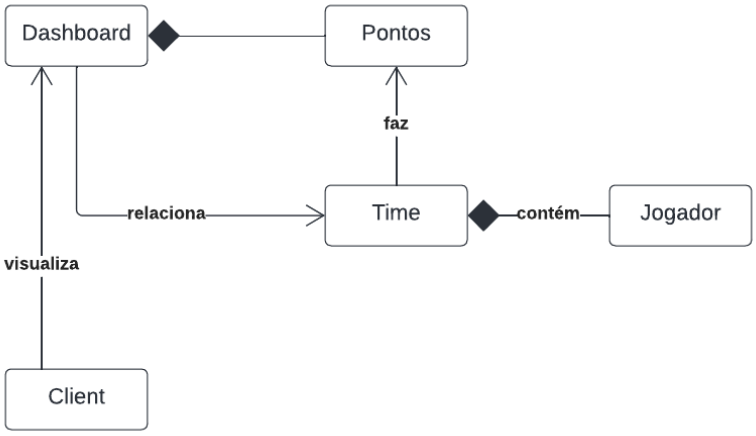
Design Pattern	Categoria	Intenção	Problema	Solução
Factory Method	Criacional	Fornece uma interface para criar objetos em uma superclasse, mas permite que as subclasses alterem	Uma aplicação pode evoluir para uma estrutura desorganizada para a criação	Você substitui as chamadas diretas de construção de objeto

		o tipo de objetos que serão criados.	de objetos diferentes	(usando o operador new) por chamadas a um método fábrica especial.
Singleton		Assegura que somente um objeto de uma determinada classe seja criado em todo o projeto		
Abstract Factory		Permite que um cliente crie famílias de objetos sem especificar suas classes concretas;		
Builder		Encapsular a construção de um produto e permitir que ele seja construído em etapas		
Prototype		permite você criar novas instancias simplesmente copiando instancias existentes		
Decorator	Estruturais	Envelopa um objeto para fornecer novos comportamentos		
Proxy		Envelopa um objeto para controlar o acesso a ele		
FlyWeigth		uma instancia de uma classe pode ser usada para fornecer muitas “instancias virtuais”		

Facade		Simplifica a interface de um conjunto de classes		
Composite		Os clientes tratam as coleções de objetos e os objetos individuais de maneira uniforme		
Bridge		Permite criar uma ponte para variar não apenas a sua implementação, como também as suas abstrações		
Adapter		Envelopa um objeto e fornece a ele uma interface diferente		
Template Method	Comportamental	As subclasses decidem como implementar os passos de um algoritimo		
Visitor		Permite acrescentar novos recursos a um composto de objetos e o encapsulamento não é importante		
Command		Encapsula uma solicitação como um objeto		
Strategy		Encapsula comportamentos intercambiáveis e usa a delegação para decidir qual deles será usado		
Chair of Responsability		Permite dar a mais de um objeto a oportunidade de		

		processar uma solicitação		
Iterator		Fornece uma maneira de acessar seqüencialmente uma coleção de objetos sem expor a sua implementação		
Mediator		Centraliza operações complexas de comunicação e controle entre objetos relacionados		
Memento		Permite restaurar um objeto a um dos seus estados prévios, por exemplo, quando o usuário seleciona um “desfazer”;		
Interpreter		Permite construir um intérprete para uma linguagem		
State		Encapsula comportamentos baseados em estados e usa a delegação para alternar comportamentos		
Observer		Permite notificar outros objetos quando ocorre uma mudança de estado		

3) Considere o seguinte Diagrama UML:



a) Crie um código em Python para representar esse Caso de Uso e aplique os Design Patterns aprendidos durante o curso - com destaque para os seguintes Design Patterns Singleton, Factory, Adapter e os princípios SOLID (3,0 pontos)

b) Faça um Diagrama UML da sua solução incluindo os Design Patterns aplicados nesse Caso de Uso (1,0 pontos)

ATENÇÃO

- **A PROVA DEVE SER ENTREGUE ATÉ DOMINGO 27/11 ÀS 23:59**
- **A SOLUÇÃO FINAL COM AS RESPOSTAS DAS QUESTÕES E DO CÓDIGO FONTE DEVEM SER ENVIADAS NO TEAMS EM PDF INDICANDO**
  - **NOME DO ALUNO**
  - **RESPOSTAS**
  - **LINK PARA O REPOSITÓRIO PESSOAL DO GIT**