

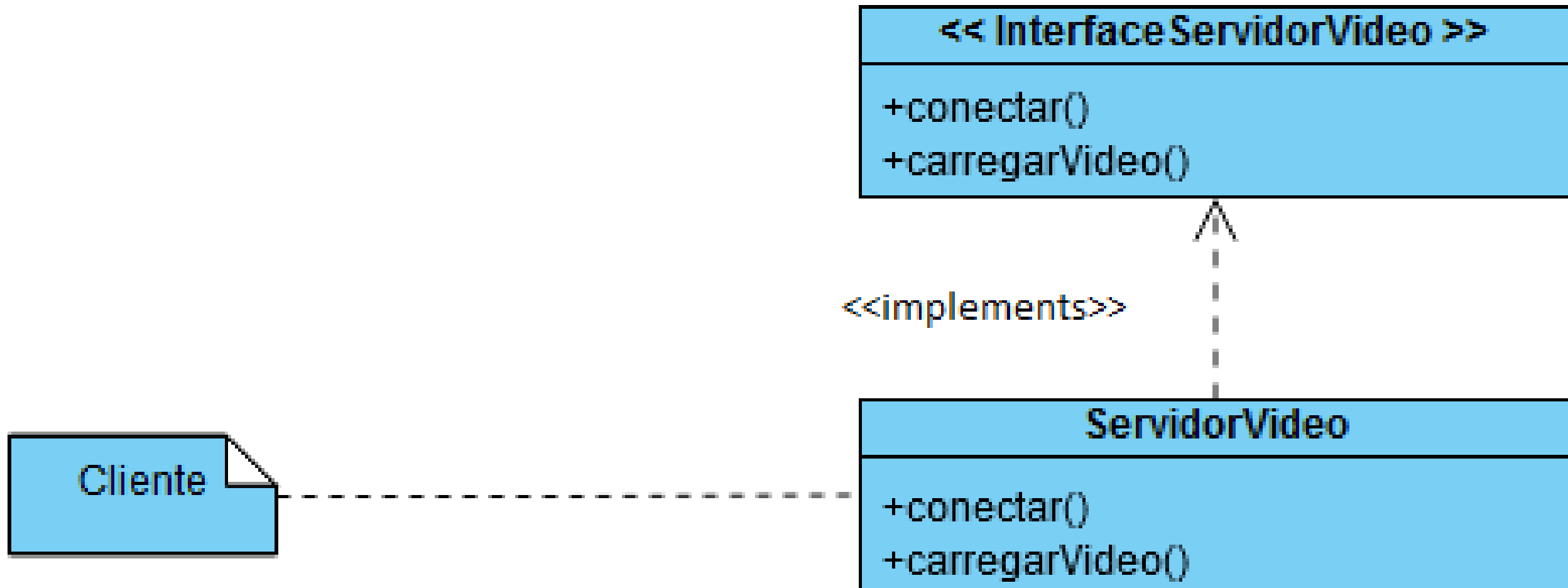
PADRÃO DE PROJETO

Proxy

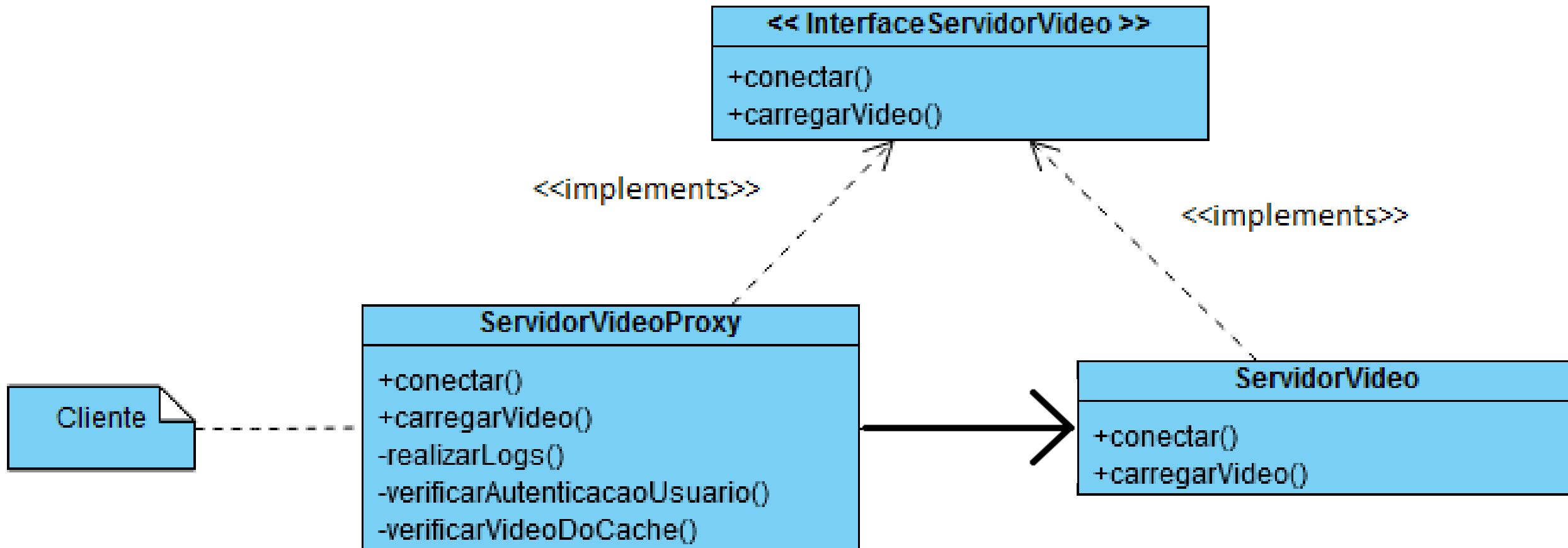
Faça

Gateway


PROXY – PROBLEMA



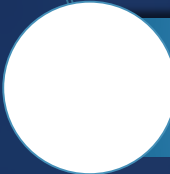
PROXY – SOLUÇÃO



PROXY – O QUE É?



O Proxy nada mais é do que um objeto que “finge” ser outro objeto real que é acessado pelo cliente.



Ele atua como um substituto de outro objeto para controlar o acesso este mesmo objeto.



O Proxy recebe os pedidos do cliente, faz alguma atividade além do pedido e repassa a solicitação para o objeto real.



Ele tem as mesmas interfaces do objeto real, por isso ele pode se passar por este objeto.


PROXY – QUANDO É USADO?



Gerenciamento de Cache: Evita que o objeto real seja acessado caso a resposta já esteja armazenada na memória do computador.

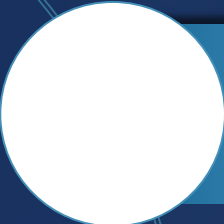


Controle de acesso: Verifica se o cliente que está acessando o método tem autenticação / permissão.



Geração de Logs: Cria Logs sobre o acesso do objeto real, armazenando dados sobre a requisição.

PROXY – QUANDO É USADO?



Lazy Instantiation: Faz algum objeto ser instanciado apenas na sua 1ª vez de uso.



Distribuição de serviços: Pode distribuir a requisição para vários outros.



Outros: Interceptar o pedido do servidor e realizar outras atividades.

PROXY – TIPOS:



Proxy Virtual – Controla o acesso de recursos de alto custo ao ser acessado ou instanciado;



Proxy Remoto – Controla o acesso de outros sistemas (subsistemas no mesmo computador ou em servidores remotos);



Proxy Acesso (Proteção) – Controla os acessos que precisam de autenticação;



Proxy Inteligente (Mais usado) – Controla o objeto real mas também faz outras tarefas em conjunto.

PROXY – VANTAGENS:



O cliente não precisa ter conhecimento se está usando proxy ou não;



O Proxy respeita o princípio SOLID, podendo ser substituído por outro Proxy;

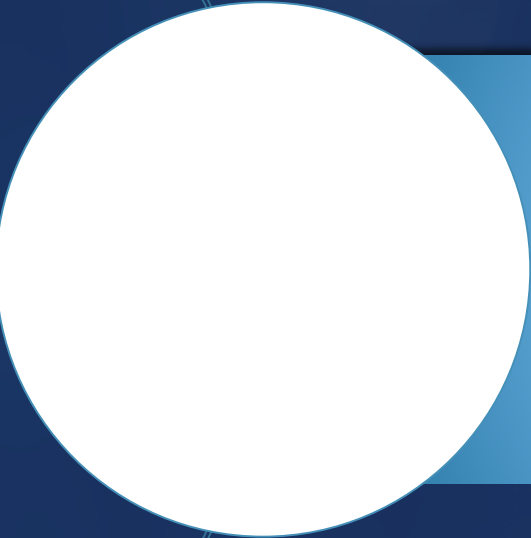


O Proxy, em alguns casos, funciona mesmo que o objeto real esteja indisponível.;

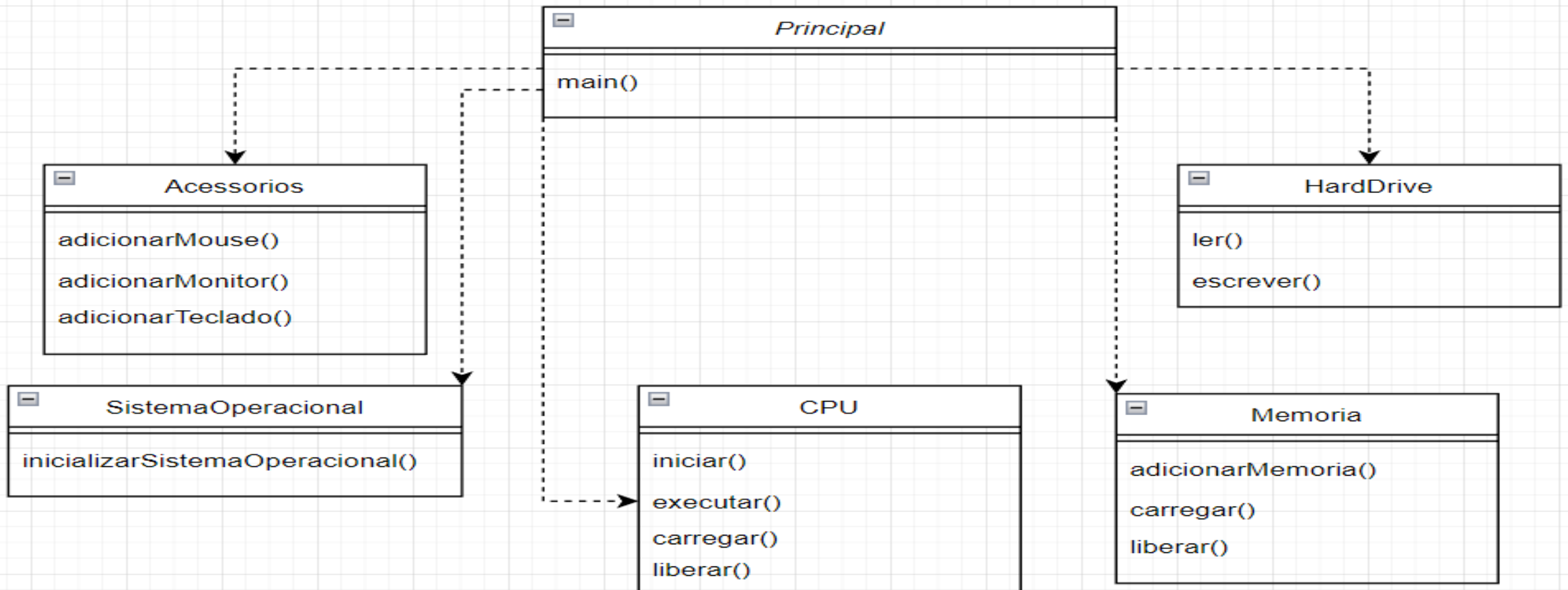


É possível controlar o tempo de vida dos objetos reais dentro do Proxy;

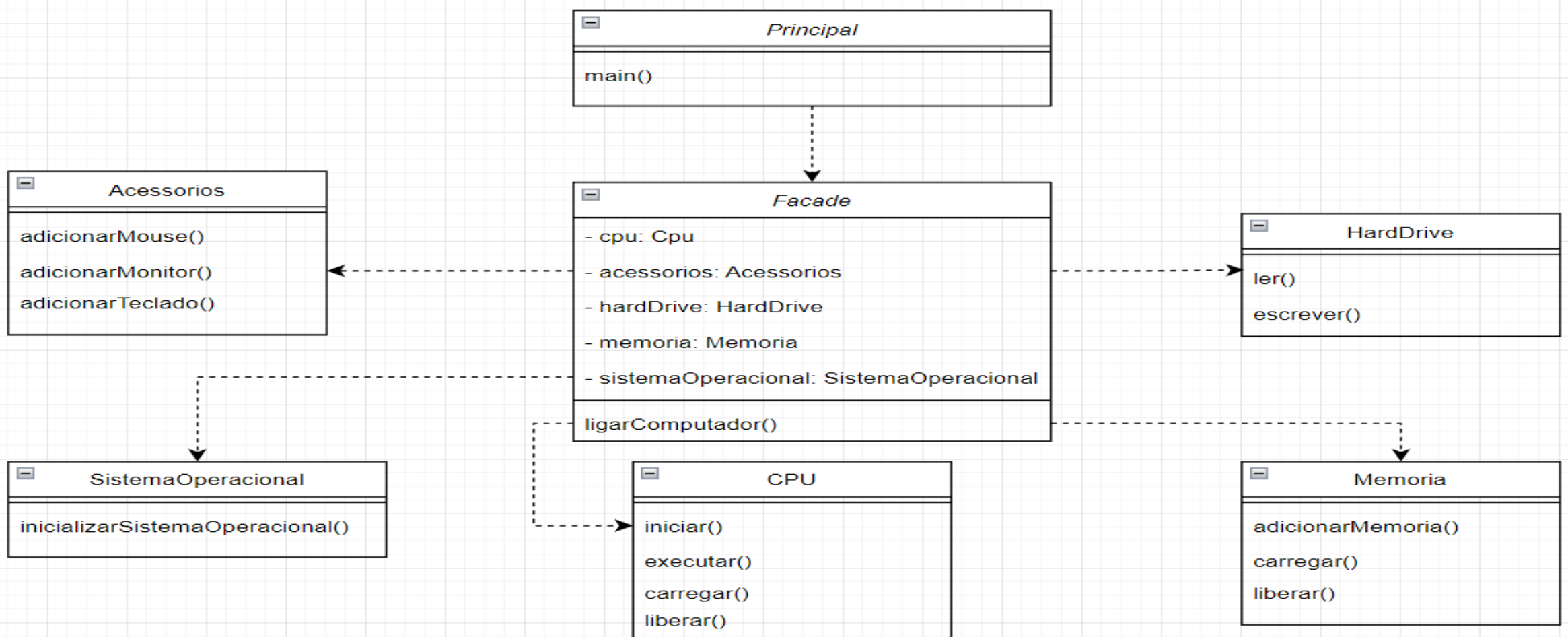
PROXY – DESVANTAGENS:



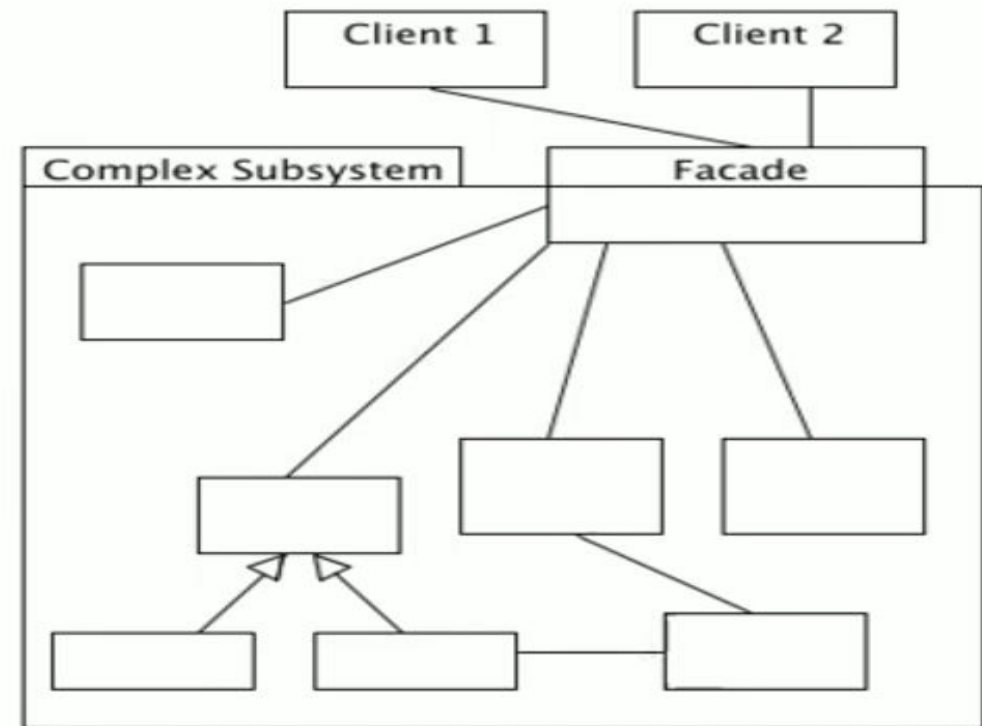
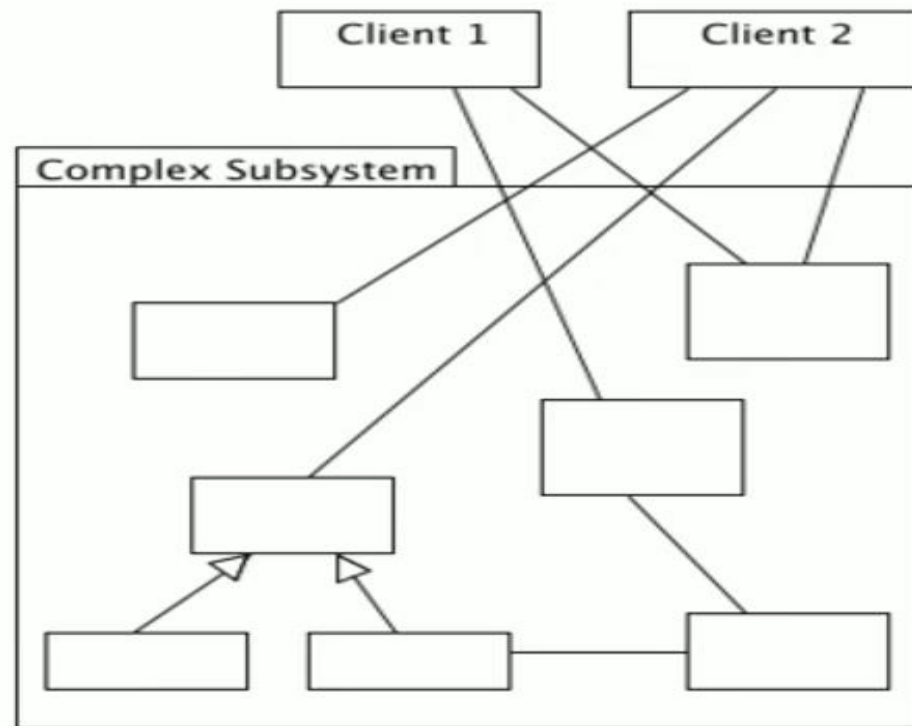
Cria mais classes dentro do sistema, aumentando a complexidade do mesmo.



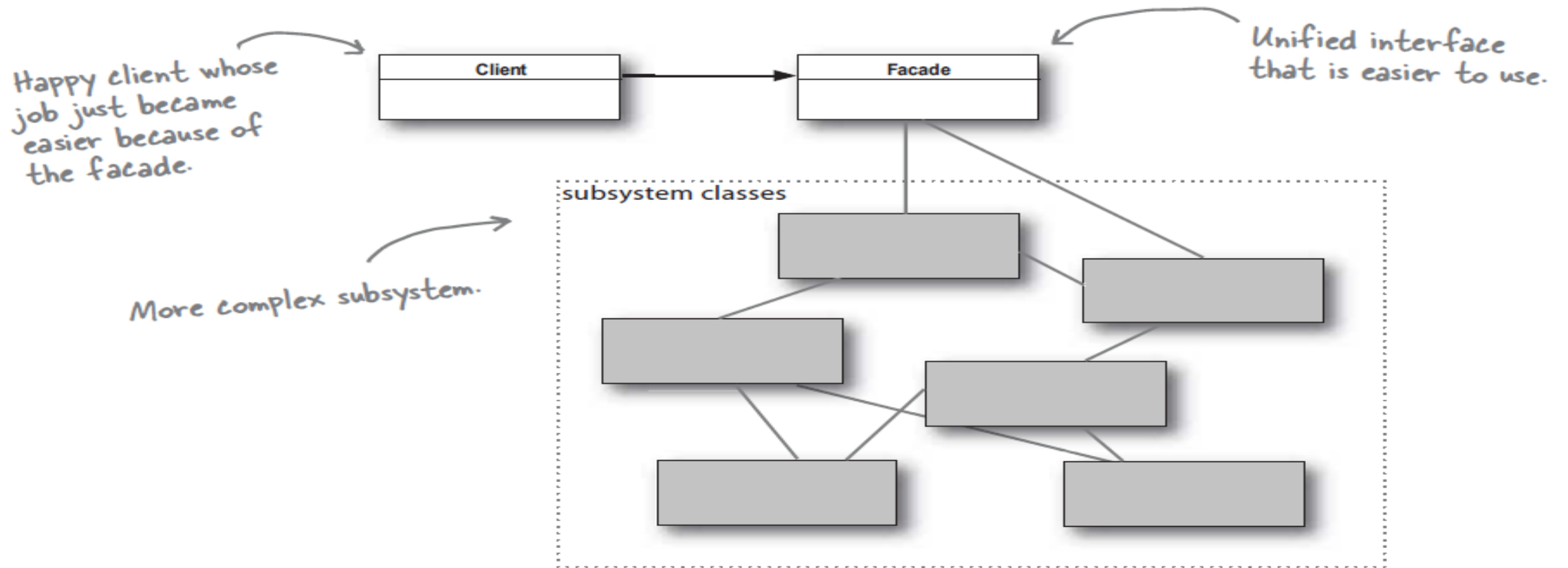
PROJETO "INICIAR CPU" – DIAGRAMA DE CLASSE SEM FAÇADE



PROJETO "INICIAR CPU" – DIAGRAMA DE CLASSE COM FAÇADE



PADRÃO DE PROJETO “FAÇADE” – (FACHADA) - FUNCIONAMENTO



PADRÃO DE PROJETO “FAÇADE” – (FACHADA) - FUNCIONAMENTO

- Fornece uma interface simplificada para um conjunto complexo de classes ou subsistemas.
- Encapsula um conjunto de interfaces mais complexas em **uma interface única** e simplificada.
- Essa camada de abstração oculta a complexidade subjacente dos subsistemas ou classes, permitindo que os clientes interajam de forma mais simples e direta, sem precisar conhecer os detalhes internos do subsistema.
- Torna mais fácil o uso das funcionalidades pelos clientes.

PADRÃO DE PROJETO ESTRUTURAL “FAÇADE” – (FACHADA)

- A "lei de Deméter" é um princípio de programação que busca promover a modularidade e a manutenibilidade do código. Ela preconiza que um objeto deve interagir apenas com objetos próximos a ele e não com objetos mais distantes, reduzindo assim o acoplamento entre os módulos do sistema. Esse princípio foi formulado por “Karl Lieberherr” e “Ian Holland” na “Universidade Northeastern” em 1987, e recebeu o nome em homenagem à deusa grega da agricultura, Deméter.

PADRÃO DE PROJETO “FAÇADE” – (FACHADA) - CURIOSIDADE

- **Simplificação de interfaces complexas:** Quando um subsistema ou conjunto de classes possui uma interface complexa ou difícil de usar, o padrão "Façade" pode ser usado para criar uma interface simplificada e mais amigável para os clientes interagirem.
- **Redução de acoplamento:** Quando os clientes estão diretamente acoplados a um subsistema complexo, qualquer mudança nesse subsistema pode afetar diretamente os clientes. O uso do padrão "Façade" pode ajudar a reduzir o acoplamento entre os clientes e o subsistema, fornecendo uma camada de abstração que encapsula a complexidade do subsistema.

PADRÃO DE PROJETO “FAÇADE” – (FACHADA) – QUANDO USAR?

- **Melhoria da modularidade e manutenção:** Ajudar a melhorar a modularidade e a manutenção do código, uma vez que os clientes interagem apenas com a interface simplificada fornecida pela fachada, em vez de terem que lidar diretamente com a complexidade do subsistema.
- **Separação de responsabilidades (clientes e o subsistema):** A fachada pode ser responsável por coordenar as interações com o subsistema e fornecer uma interface coesa e consistente para os clientes, enquanto o subsistema pode se concentrar em suas tarefas específicas.

PADRÃO DE PROJETO “FAÇADE” – (FACHADA) – QUANDO USAR?

-
- **Melhoria na legibilidade e compreensão do código:** É possível criar uma interface mais clara e simplificada para os clientes, o que pode tornar o código mais legível e compreensível, especialmente em situações em que o subsistema é complexo e difícil de entender.

PADRÃO DE PROJETO “FAÇADE” – (FACHADA) – QUANDO USAR?

-
- **Acrescentar novas funcionalidades:** Alterar a “Façade” ao invés de alterar diversos pontos do sistema.
 - **Simplificação da complexidade:** O principal objetivo do padrão “Façade” é simplificar a complexidade de subsistemas complexos.
 - **Abstração do subsistema:** O “Façade” atua como uma camada de abstração que encapsula a complexidade dos subsistemas.

PADRÃO DE PROJETO “FAÇADE” – (FACHADA) – VANTAGENS

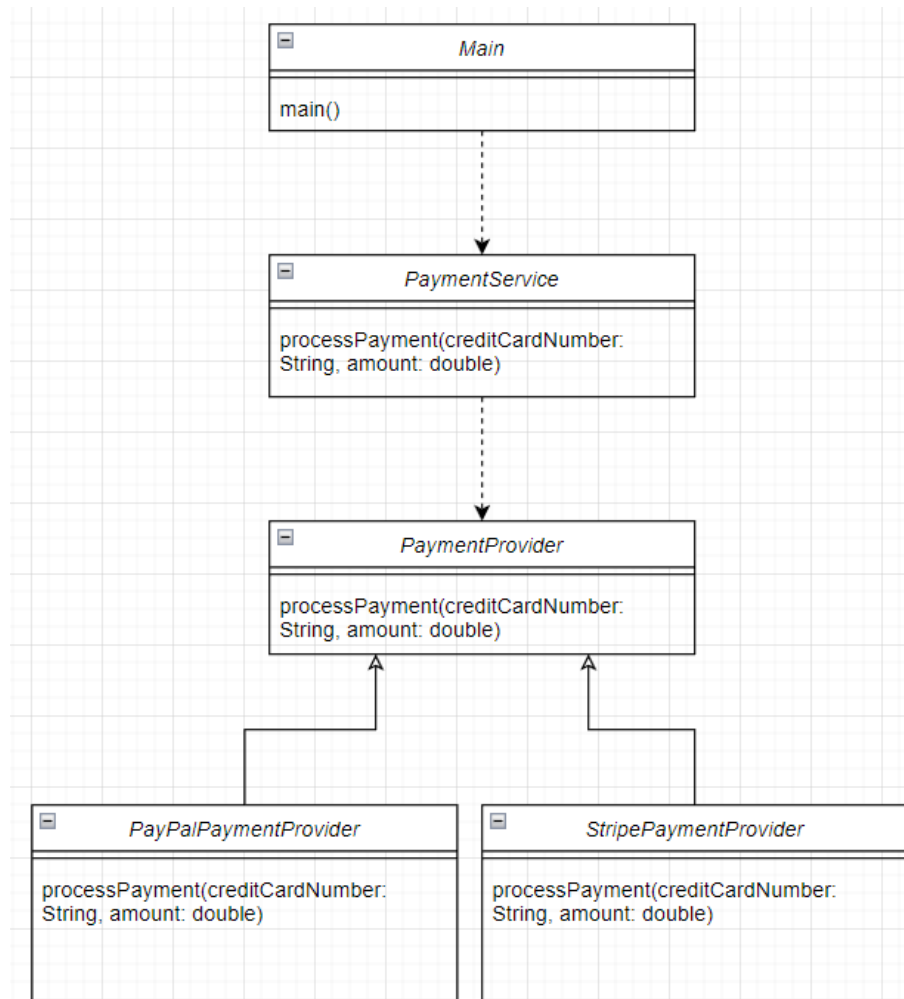
- **Redução do acoplamento:** Reduz o acoplamento entre os clientes e os subsistemas, uma vez que os clientes se comunicam apenas com a fachada em vez de interagir diretamente com os subsistemas.
- **Melhoria da manutenibilidade:** Facilita a manutenção do código, uma vez que as mudanças nos subsistemas podem ser isoladas na fachada sem afetar os clientes.

PADRÃO DE PROJETO “FAÇADE” – (FACHADA) – VANTAGENS

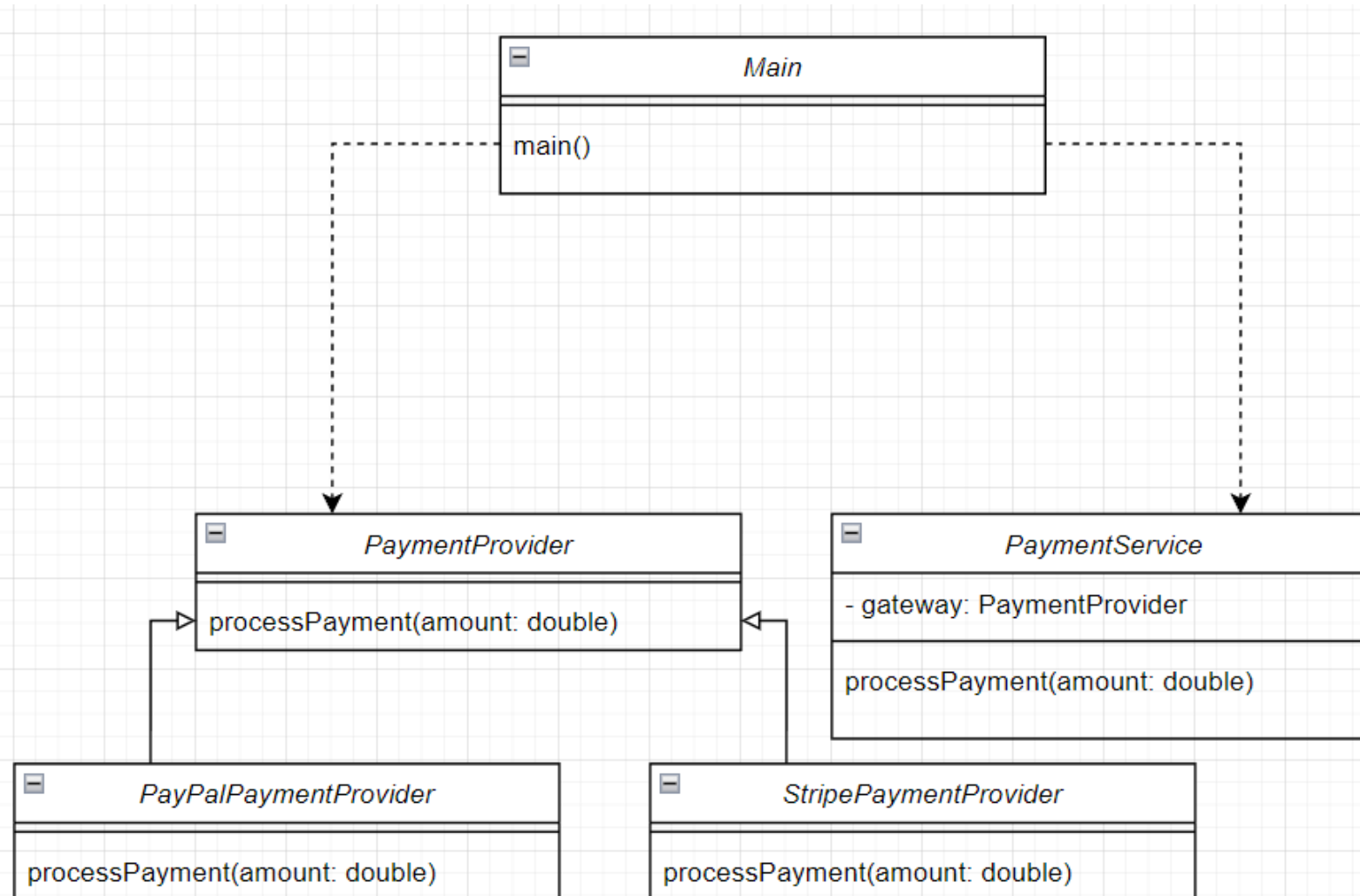
- **Restrição à personalização:** Em alguns casos, o encapsulamento da complexidade dos subsistemas pode restringir a capacidade dos clientes de personalizar ou estender as funcionalidades dos subsistemas. Isso pode limitar a flexibilidade do sistema e pode não ser adequado em algumas situações em que a personalização é um requisito importante.
- **A fachada quebra o SRP (princípio da responsabilidade única):** Uma única fachada será uma classe que faz tudo. Para minimizar o problema pode-se criar várias fachadas.

PADRÃO DE PROJETO “FAÇADE” – (FACHADA) – DESVANTAGENS

PADRÃO DE PROJETO – SEM GATEWAY - "PAGAMENTO CARTÃO"



PADRÃO DE PROJETO – COM GATEWAY - "PAGAMENTO CARTÃO"



PADRÃO DE PROJETO – GATEWAY - "ADAPTADOR DE DADOS"

- Abstrai a comunicação entre diferentes sistemas ou tecnologias heterogêneas, oferecendo uma interface unificada para acesso a esses sistemas.
- Encapsula a lógica de acesso a dados e fornecer uma camada de abstração.
- Facilita a integração de sistemas com diferentes interfaces e formatos de dados.
- Utilizado em situações em que há a necessidade de integração com sistemas ou serviços de dados externos.

PADRÃO DE PROJETO – GATEWAY – “REPOSITÓRIO” - USO

- **Integração com bancos de dados:** Usado para encapsular a lógica de acesso a um banco de dados, fornecendo uma interface única para acesso a dados, independentemente do tipo de banco de dados utilizado (SQL, NoSQL, etc.), ou mesmo para fornecer uma camada de abstração para acesso a diferentes bancos de dados.
- **Integração com serviços web ou APIs externas:** Usado para encapsular a lógica de acesso a serviços web ou APIs externas, fornecendo uma interface única para acesso a esses serviços, independentemente do fornecedor ou tecnologia utilizada.

PADRÃO DE PROJETO - GATEWAY – “REPOSITÓRIO” - USO

- **Integração com sistemas legados:** Usado para encapsular a lógica de acesso a sistemas legados, que podem ter interfaces de comunicação obsoletas ou complexas, permitindo uma integração mais simplificada com sistemas modernos.
- **Integração com sistemas externos diversos:** Usado para encapsular a lógica de acesso a diferentes sistemas externos, como sistemas de terceiros, sistemas de pagamento, sistemas de mensageria, entre outros, fornecendo uma interface única para acesso a esses sistemas e simplificando a integração.

PADRÃO DE PROJETO – GATEWAY - TIPOS

- **Data Gateway:** Também conhecido como "Data Access Gateway", esse tipo de "Gateway" é usado para encapsular a lógica de acesso a dados, como bancos de dados, serviços de armazenamento, APIs de dados, entre outros. Ele fornece uma interface única para acesso a diferentes fontes de dados, abstraindo detalhes de implementação e permitindo a troca de fontes de dados sem afetar o restante do sistema.
- **Integration Gateway:** Esse tipo de "Gateway" é usado para encapsular a lógica de integração com sistemas externos, como serviços web, APIs externas, sistemas legados, entre outros. Ele fornece uma interface única para integração com diferentes sistemas, abstraindo a complexidade das interfaces de comunicação e simplificando a integração com o restante do sistema.

PADRÃO DE PROJETO - GATEWAY - TIPOS

- **Protocol Gateway:** Esse tipo de "Gateway" é usado para encapsular a lógica de conversão de protocolos de comunicação, permitindo a interoperabilidade entre diferentes protocolos de comunicação, como HTTP, TCP/IP, SOAP, REST, MQTT, entre outros. Ele pode ser usado, por exemplo, para adaptar a comunicação entre sistemas com protocolos diferentes.
- **API Gateway:** Esse tipo de "Gateway" é usado para encapsular a lógica de exposição e gerenciamento de APIs (Application Programming Interfaces), fornecendo uma interface única para acesso a diferentes APIs, controlando o acesso, autenticação, autorização, cache, entre outros. Ele é comumente utilizado em arquiteturas de microsserviços para simplificar a exposição e gerenciamento de APIs.

PADRÃO DE PROJETO - GATEWAY - TIPOS

- **Message Gateway:** Esse tipo de "Gateway" é usado para encapsular a lógica de comunicação assíncrona entre sistemas, como troca de mensagens em sistemas de filas, eventos, ou outros mecanismos de mensageria. Ele pode ser usado para abstrair a complexidade da comunicação assíncrona e fornecer uma interface mais simplificada para a troca de mensagens entre sistemas.

PADRÃO DE PROJETO – GATEWAY - VANTAGENS

- **Abstração de detalhes de implementação:** O "Gateway" encapsula a lógica de acesso a dados, integração com sistemas externos ou conversão de protocolos, abstraindo os detalhes de implementação e permitindo que o restante do sistema interaja com ele de forma simplificada. Isso pode reduzir a complexidade do código e melhorar a manutenibilidade.
- **Reutilização de código:** Ao utilizar um "Gateway", a lógica específica de acesso a dados, integração ou conversão de protocolos pode ser centralizada em um único componente, facilitando a reutilização desse código em diferentes partes do sistema. Isso pode levar a uma maior eficiência de desenvolvimento e redução de duplicação de código.

PADRÃO DE PROJETO - GATEWAY - VANTAGENS

- **Flexibilidade e escalabilidade:** O "Gateway" pode ser projetado para ser flexível e adaptável a diferentes fontes de dados, sistemas externos ou protocolos de comunicação. Isso permite que o sistema seja facilmente escalável e adaptável a mudanças nos requisitos ou tecnologias utilizadas, sem afetar o restante do sistema.
- **Gerenciamento centralizado:** O "Gateway" pode atuar como um ponto central de gerenciamento e controle para acesso a dados, integração ou comunicação entre sistemas. Isso pode simplificar a administração do sistema, permitindo um melhor monitoramento, controle e gerenciamento das operações relacionadas.

PADRÃO DE PROJETO – GATEWAY - DESVANTAGENS

- **Complexidade adicional:** A introdução de um componente adicional, como o "Gateway", pode adicionar complexidade ao sistema, especialmente em sistemas menores ou mais simples. A criação e manutenção do "Gateway" pode exigir esforço adicional de desenvolvimento e testes.
- **Sobrecarga de desempenho:** Dependendo da implementação específica, o uso do "Gateway" pode introduzir uma sobrecarga de desempenho, uma vez que a comunicação com sistemas externos, acesso a dados ou conversão de protocolos pode introduzir latência ou consumo de recursos adicionais.

PADRÃO DE PROJETO – GATEWAY - DESVANTAGENS

- **Acoplamento:** A introdução de um "Gateway" pode aumentar o acoplamento entre componentes do sistema, uma vez que a comunicação ou integração passa a ser feita através dele. Isso pode dificultar a substituição ou extensão de componentes relacionados.
- **Complexidade de configuração:** Em alguns casos, a configuração e gerenciamento do "Gateway" pode exigir esforço adicional, especialmente quando envolve a configuração de múltiplas fontes de dados, sistemas externos ou protocolos de comunicação.





OBRIGADO

ALESSANDRO FRANCISCO LEITE

LEONARDO VALADÃO OLIVEIRA