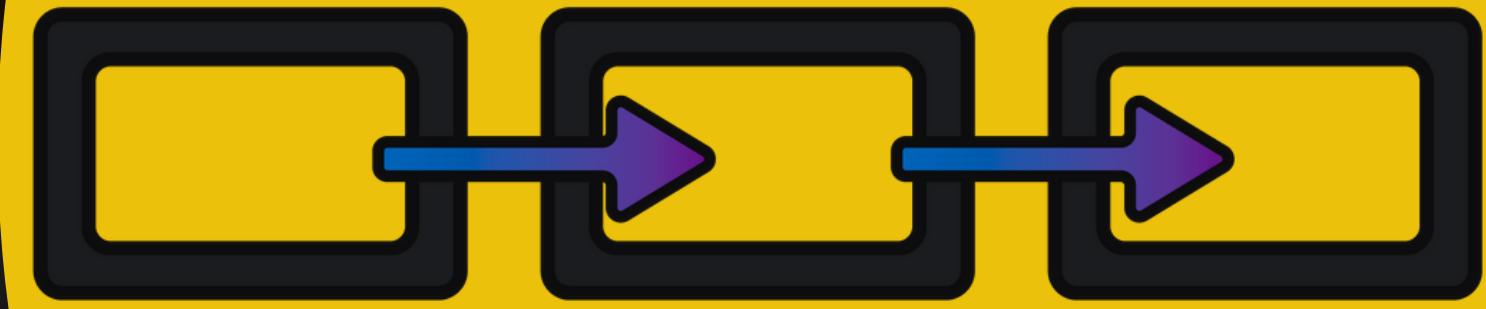


MÓDULO PADRÕES DE PROJETOS COMPORTAMENTAIS

VAMOS FOCAR AGORA EM COMO AS
RESPONSABILIDADES SÃO PROPAGADAS

AULA CHAIN OF RESPONSABILITY

NINGUÉM SOLTA A MÃO DE NINGUÉM





OBJETIVOS

- APRESENTAR O PROBLEMA GERAL
- APRESENTAR UMA SOLUÇÃO UTILIZANDO O CHAIN OF RESPONSABILITY
- VER COMO UMA APLICAÇÃO PODE SER CRIADA JUNTANDO PEÇAS ATÔMICAS

PROBLEMAS

- COMO POSSO EVITAR O ACOPLAMENTO ENTRE O REMETENTE DE UMA SOLICITAÇÃO E SEU RECEPTOR?
- COMO PERMITIR QUE MAIS QUE UM OBJETO POSSA ATENDER ALGUMA REQUISIÇÃO?



SOLUÇÃO

- DEFINIR UMA CADEIA DE OBJETOS ONDE CADA UM PODERÁ RESPONDER ÀQUELA SOLICITAÇÃO OU ENVIAR PARA O PRÓXIMO OBJETO TRATÁ-LA.
- QUEM FAZ A SOLICITAÇÃO NÃO PRECISA SABER O TAMANHO DA CADEIA, OU MESMO COMO (POR QUEM) ELA SERÁ RESOLVIDA

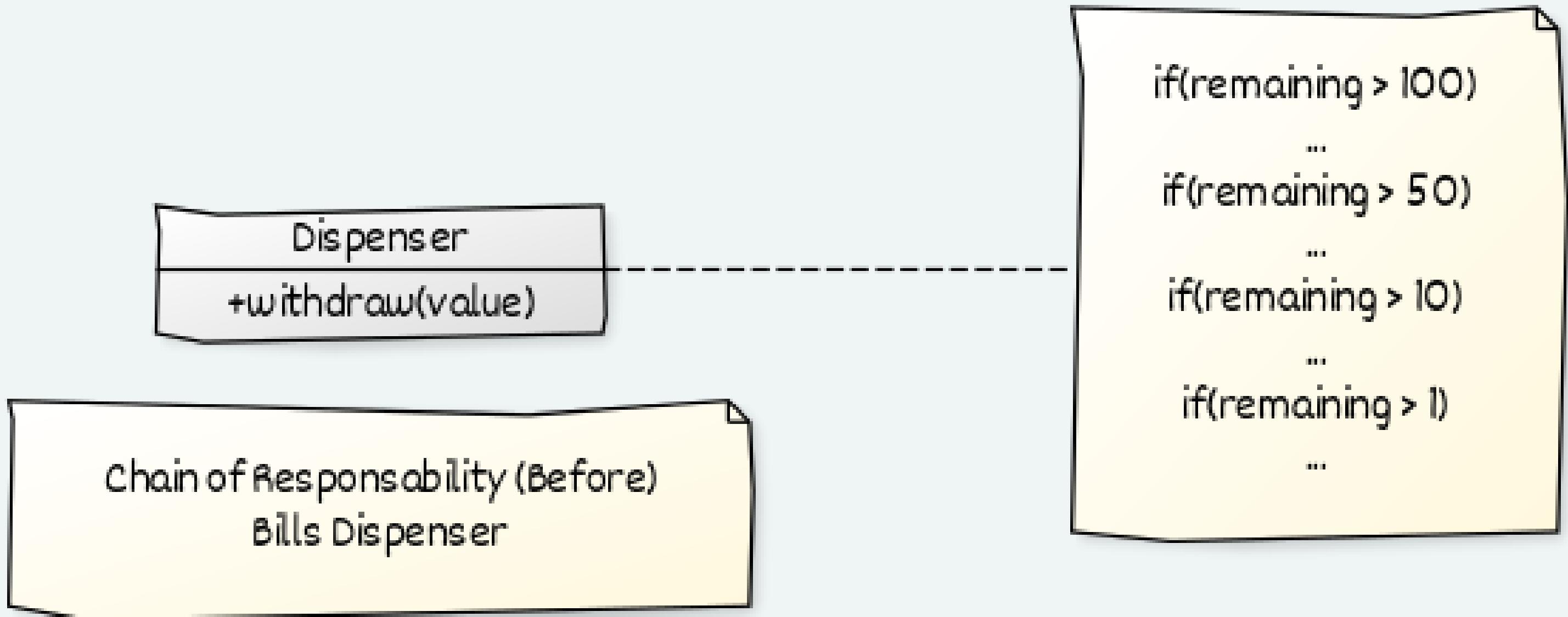




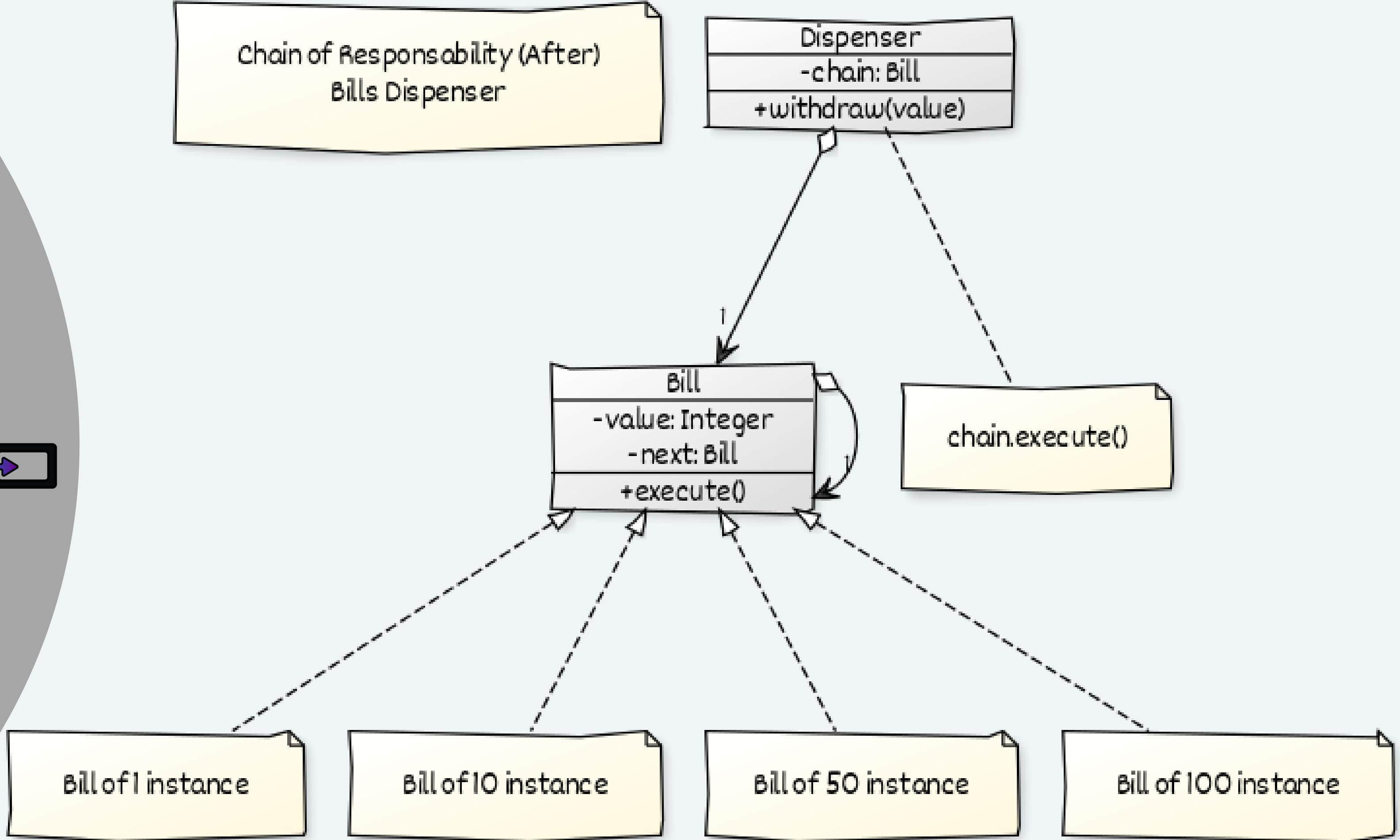
Evita acoplar o remetente de uma requisição ao seu destinatário ao dar a mais de um objeto a chance de servir a requisição. Compõe os objetos em cascata e passa a requisição pela corrente até que um objeto a sirva.

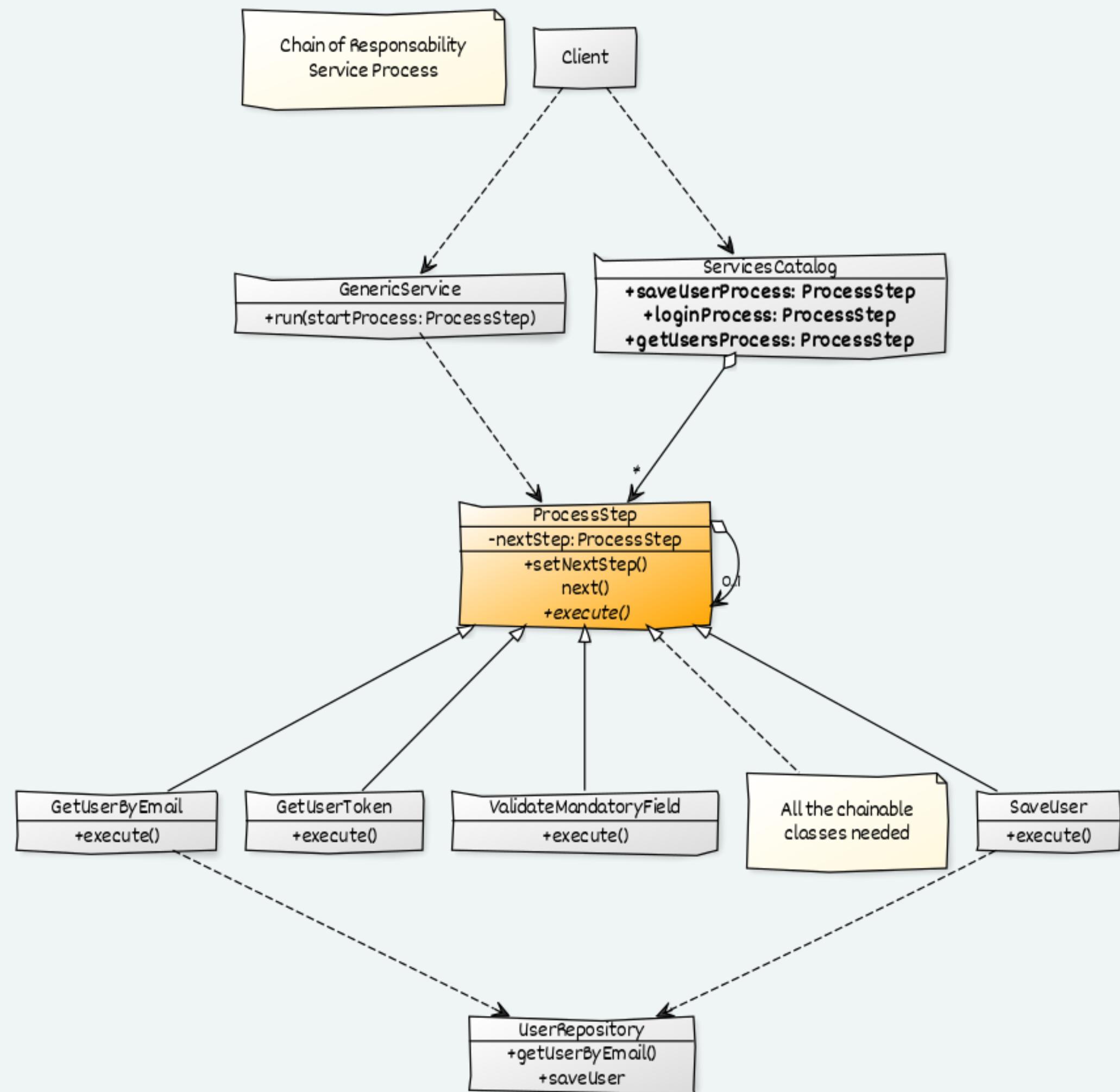
GOF





CREATED WITH YUML





AULA MEMENTO

CHECKPOINT!





OBJETIVOS

- APRESENTAR O PROBLEMA GERAL
- APRESENTAR UMA SOLUÇÃO UTILIZANDO O MEMENTO

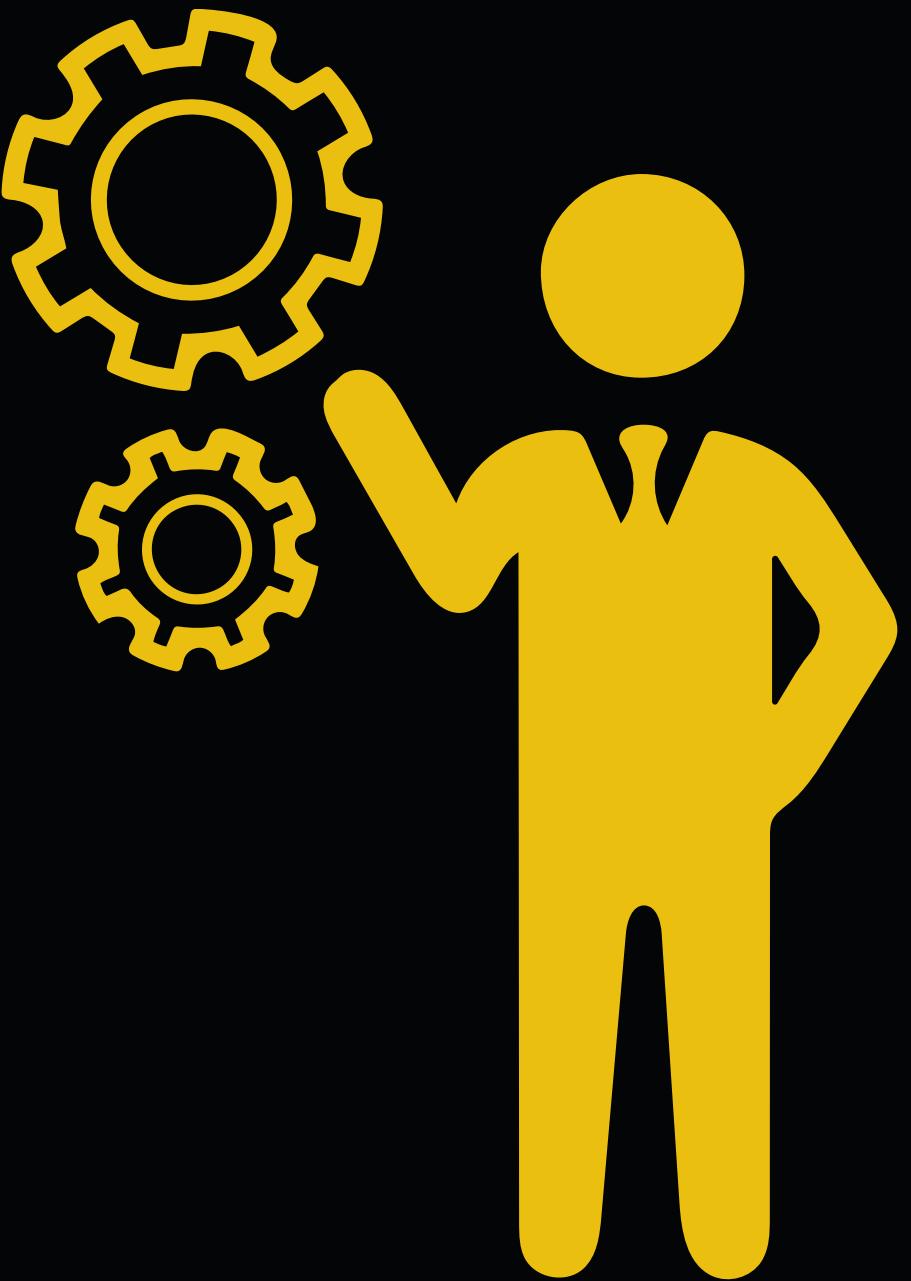
PROBLEMAS

- COMO UM OBJETO PODE CAPTURAR SEU ESTADO INTERNO PARA PODER SER RESTAURADO POSTERIORMENTE?



SOLUÇÃO

- DEFINIR UMA ESTRUTURA DE MENTO
ONDE CADA ESTADO PODERÁ SER SALVO
E RESTAURADO QUANDO NECESSÁRIO

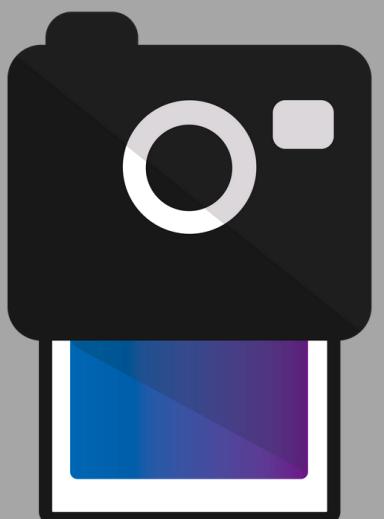
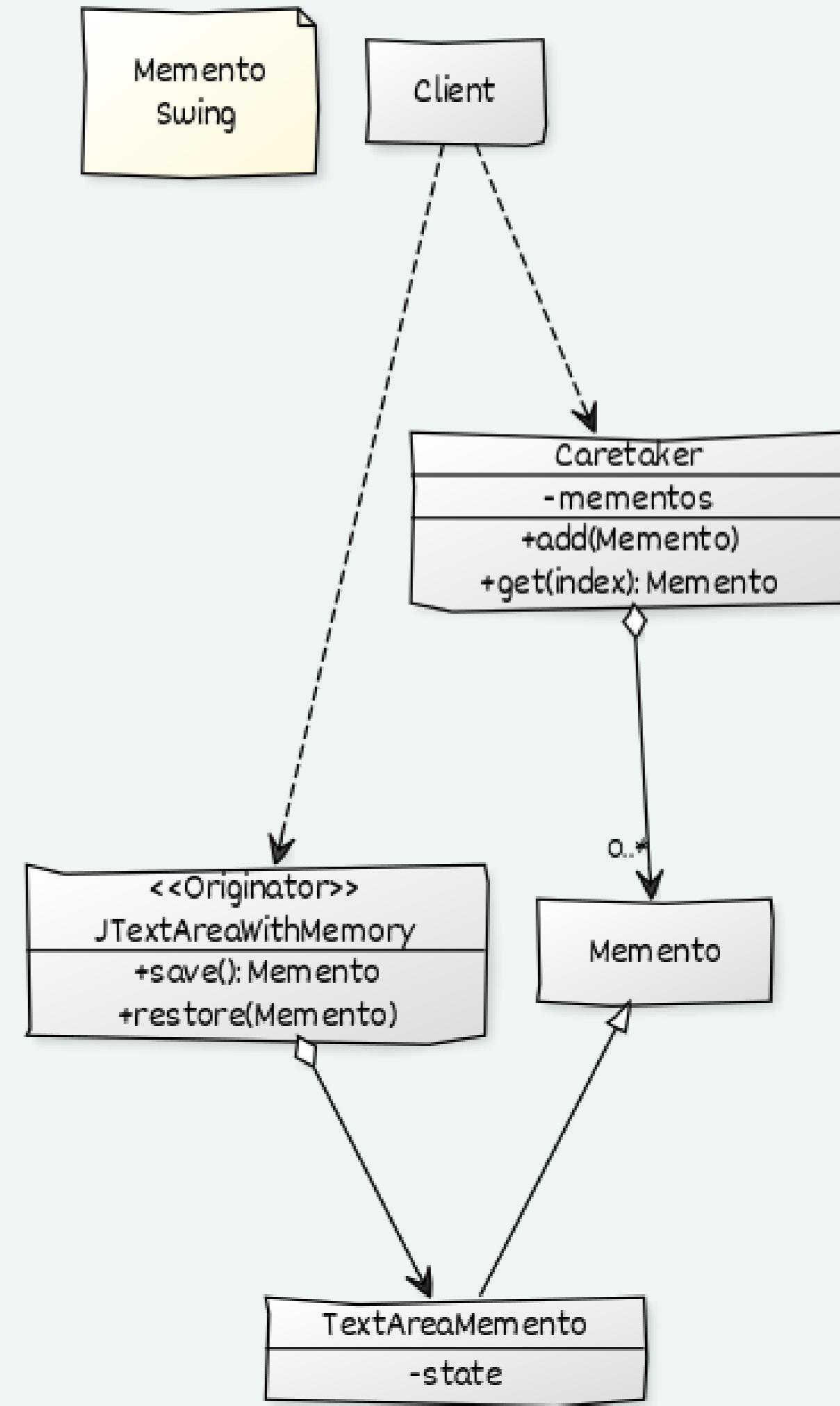




Sem violar o encapsulamento, capturar e externalizar o estado interno de um objeto para que o objeto possa ter esse estado restaurado posteriormente.

GOF

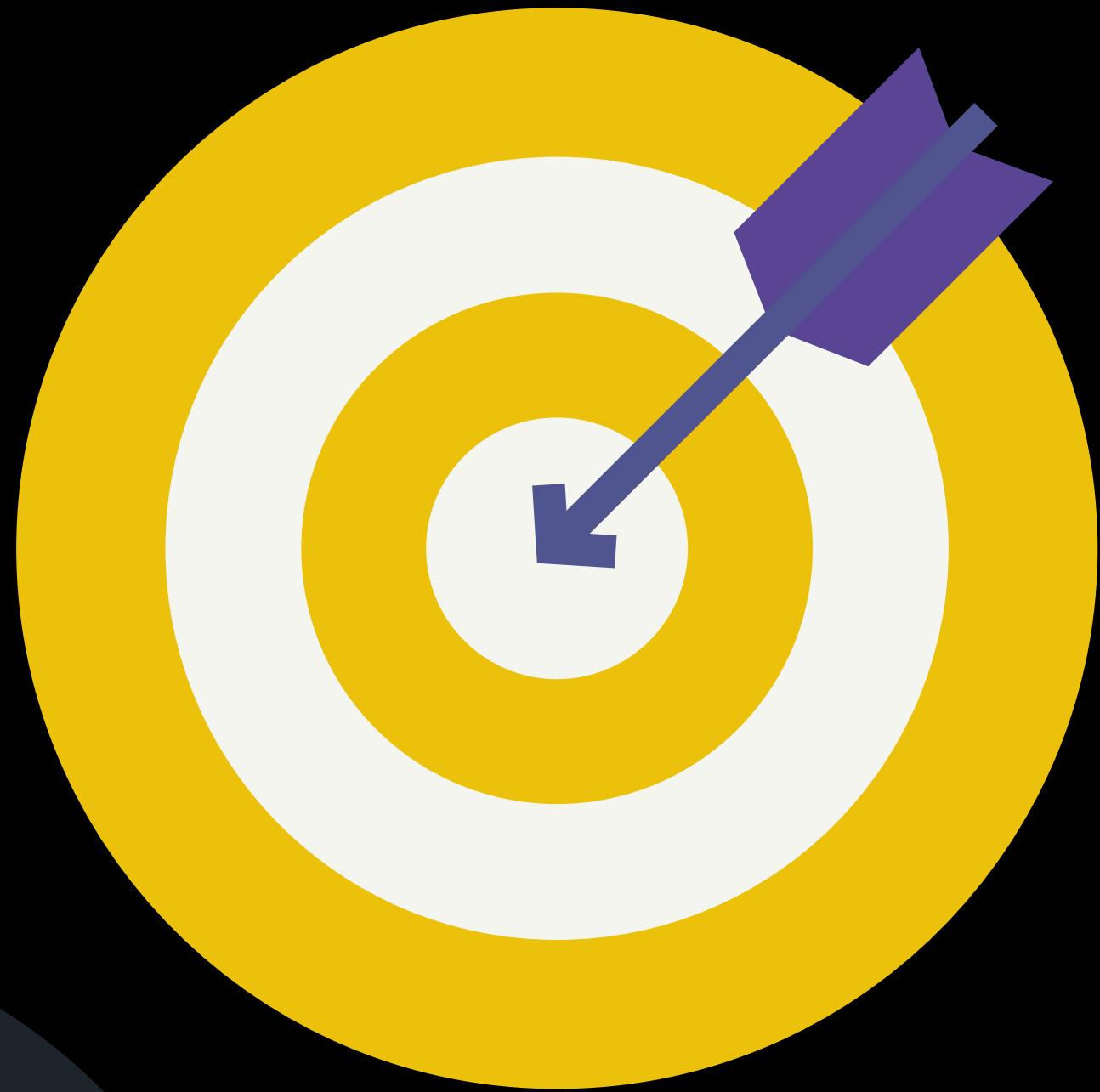




AULA COMMAND

JUST DO IT!





OBJETIVOS

- APRESENTAR O PROBLEMA GERAL
- APRESENTAR UMA SOLUÇÃO UTILIZANDO O COMMAND
- APRESENTAR COMO ESTRATÉGIAS DE MIGRAÇÃO PODEM TRABALHAR UTILIZANDO COMMAND

PROBLEMAS

- COMO POSSO REPRESENTAR UMA REQUISIÇÃO DENTRO DE UM OBJETO?
- PRECISO FAZER UMA REQUISIÇÃO MAS NÃO SEI COMO ELA SERÁ RESOLVIDA OU ATÉ MESMO QUEM IRÁ RESPONDÊ-LA?



SOLUÇÃO

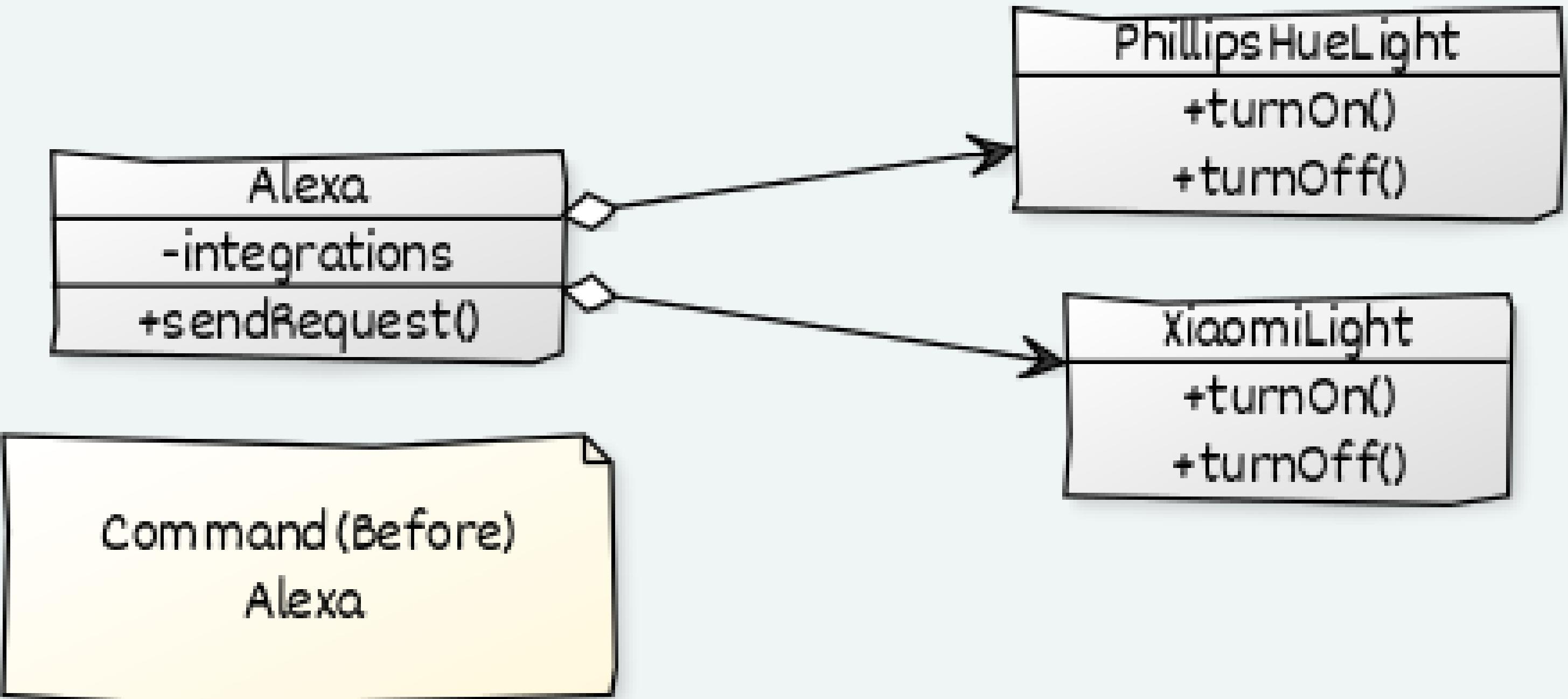
- ENCAPSULAR A REQUISIÇÃO EM UM OBJETO COMMAND SEPARADO
- O COMANDO NÃO TEM OS DETALHES DE QUEM E COMO SERÁ RESOLVIDO



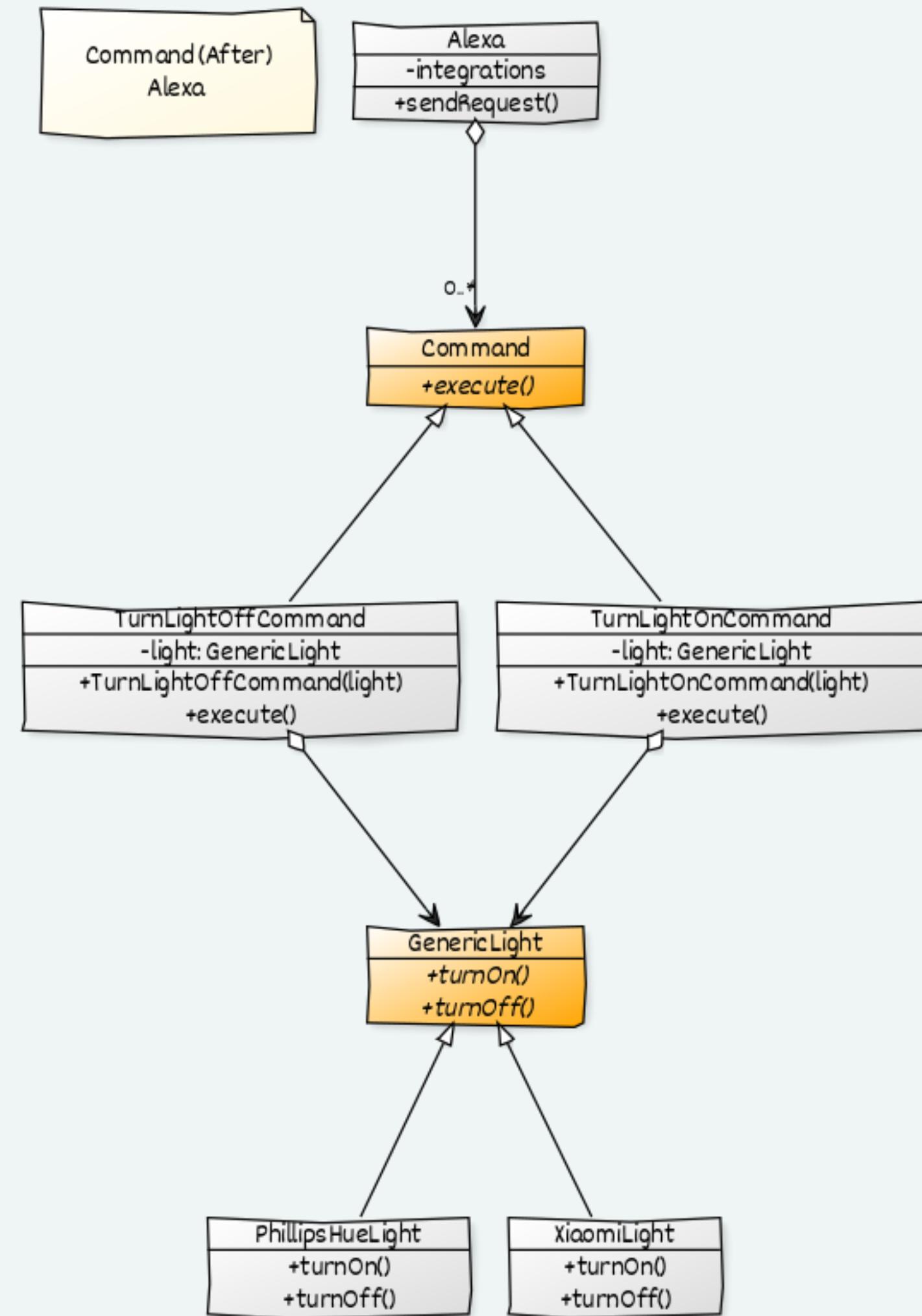
Encapsular uma requisição como um objeto, permitindo que clientes parametrizem diferentes requisições, filas ou requisições de log, e suportar operações reversíveis.

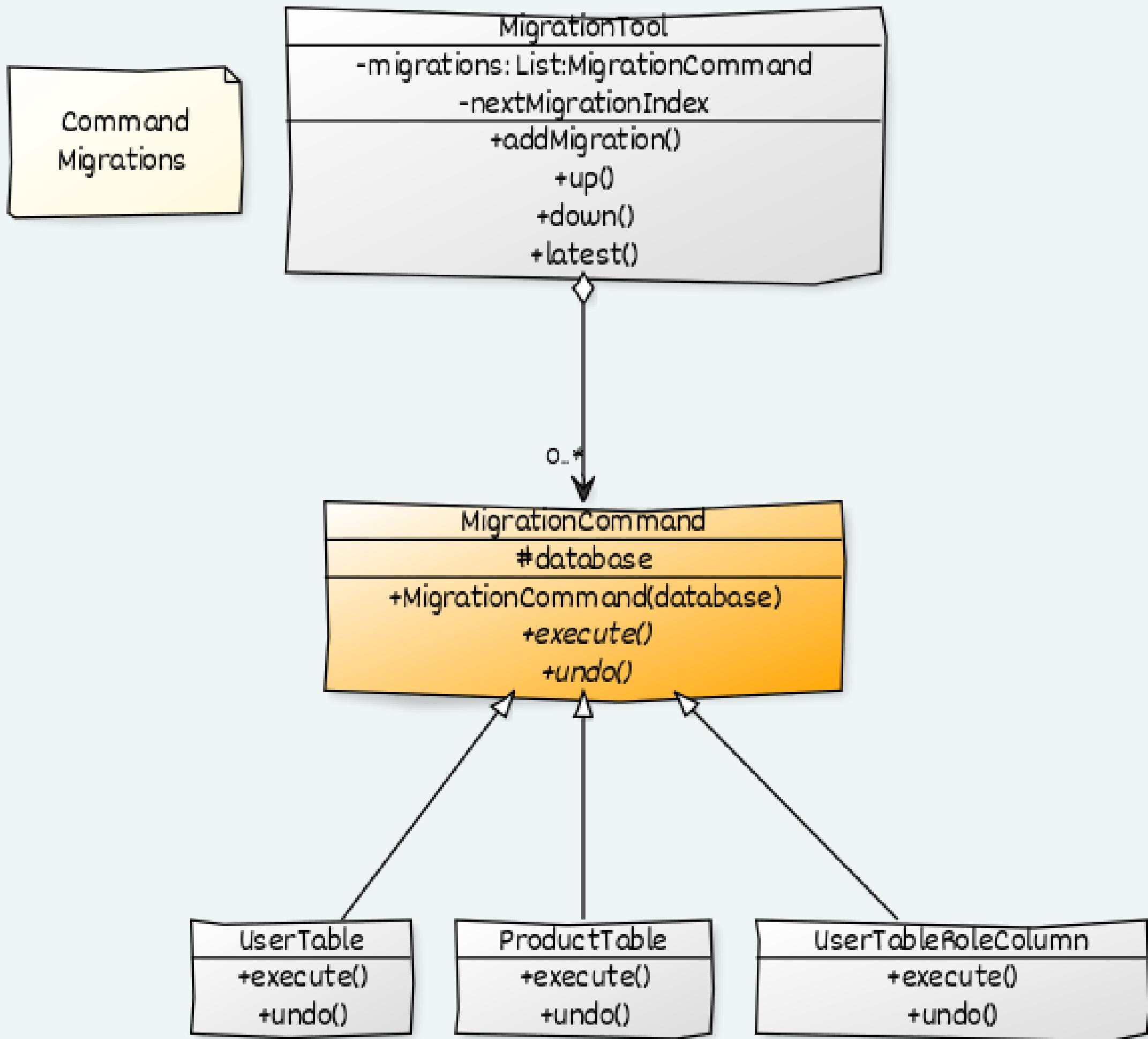
GOF





CREATED WITH YUML

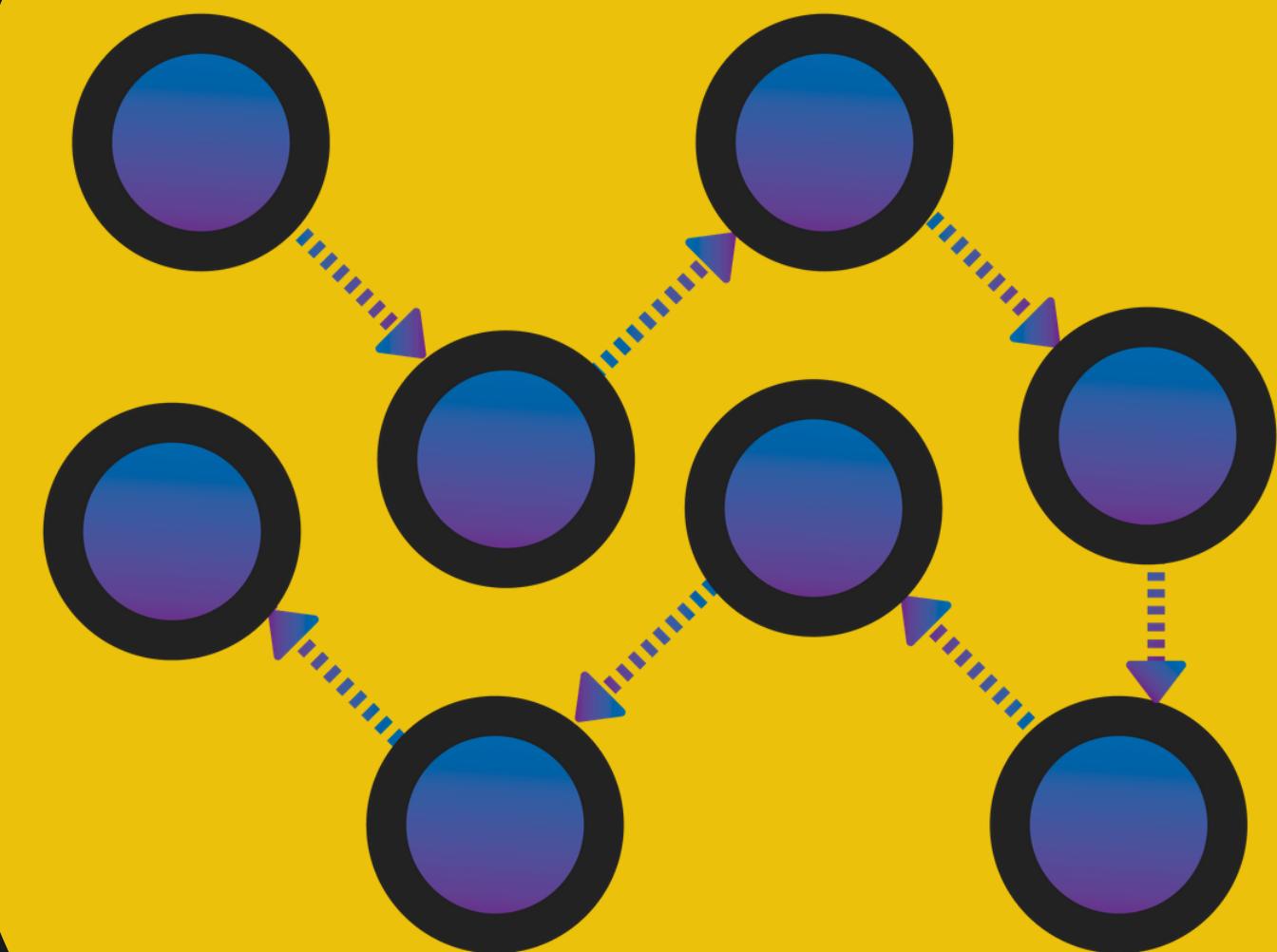


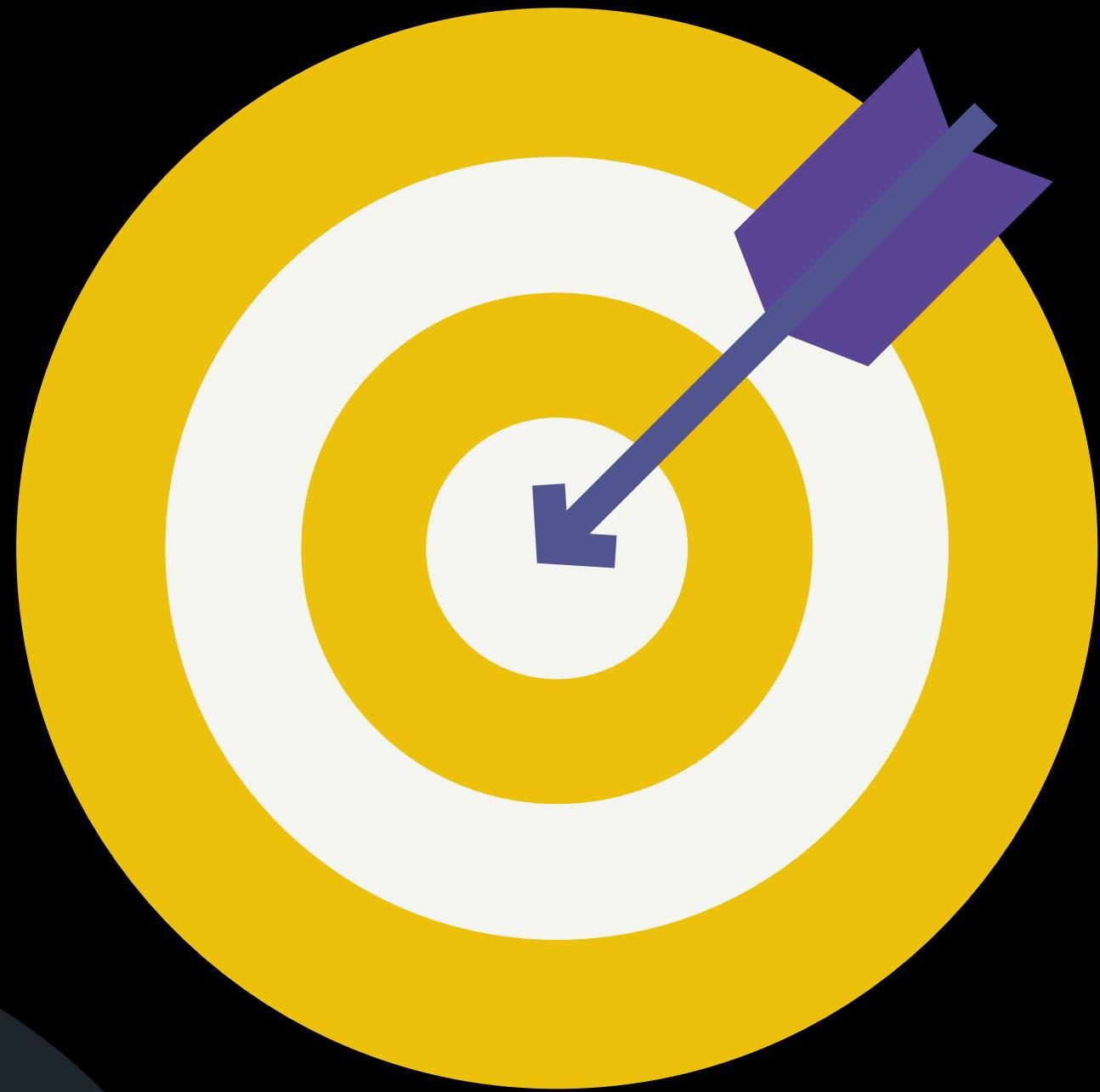


AULA ITERATOR

PRIMEIRO ELE, DEPOIS ELE ALÍ E DEPOIS

AQUELE LÁ...





OBJETIVOS

- APRESENTAR O PROBLEMA GERAL
- APRESENTAR UMA SOLUÇÃO UTILIZANDO O ITERATOR
- VER COMO ESTE PADRÃO SE INTEGRA COMPLETAMENTE COM LAÇOS E STREAMS

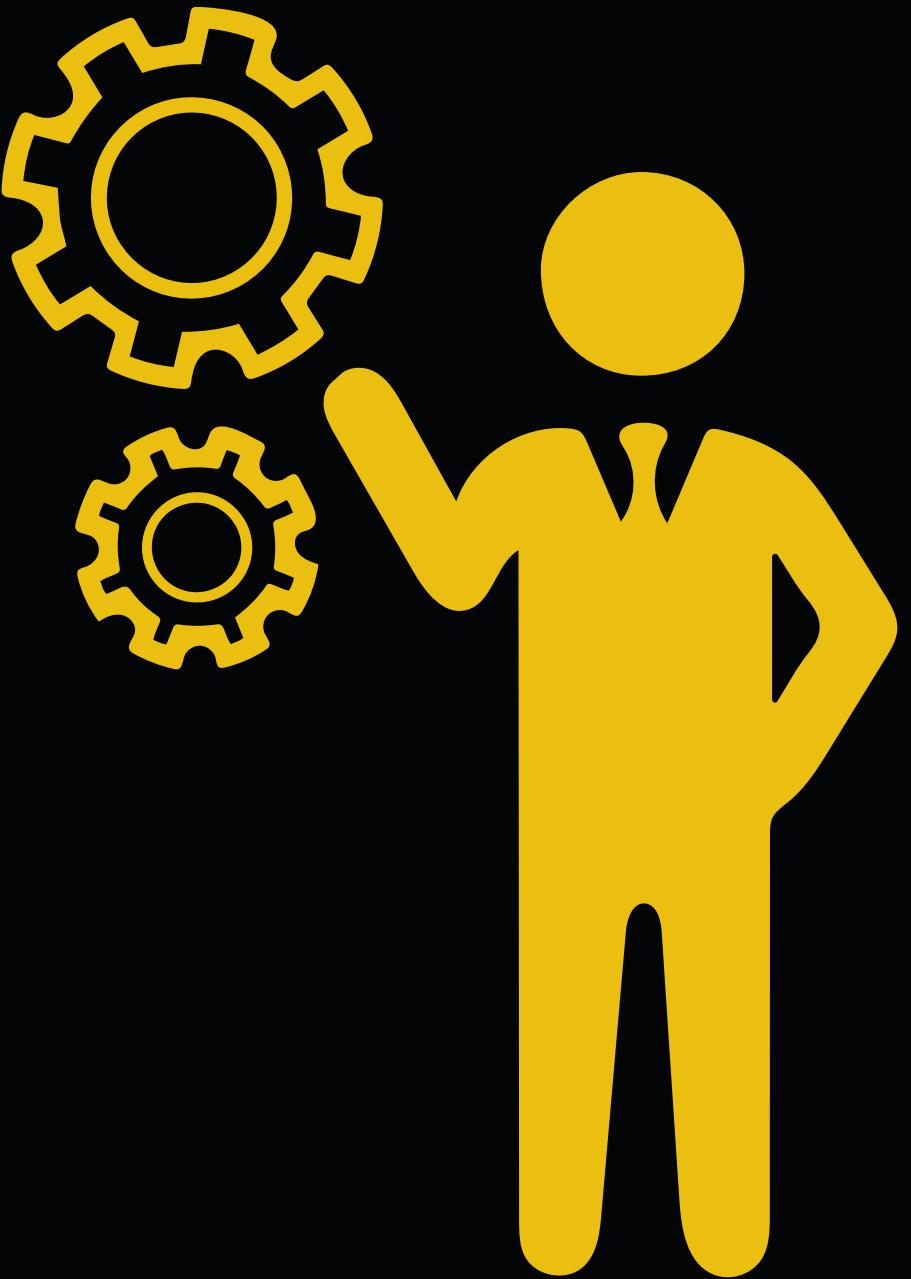
PROBLEMAS

- COMO POSSO NAVEGAR ATRAVÉS DE
UMA COLEÇÃO DE OBJETOS SEM A
NECESSIDADE DE CONHECER OS
DETALHES DESTA ESTRUTURA?



SOLUÇÃO

- ENCAPSULAR A LÓGICA DE NAVEGAÇÃO
ENTRE OS ELEMENTOS DESTA COLEÇÃO
EM UMA ESTRUTURA DE ITERATOR

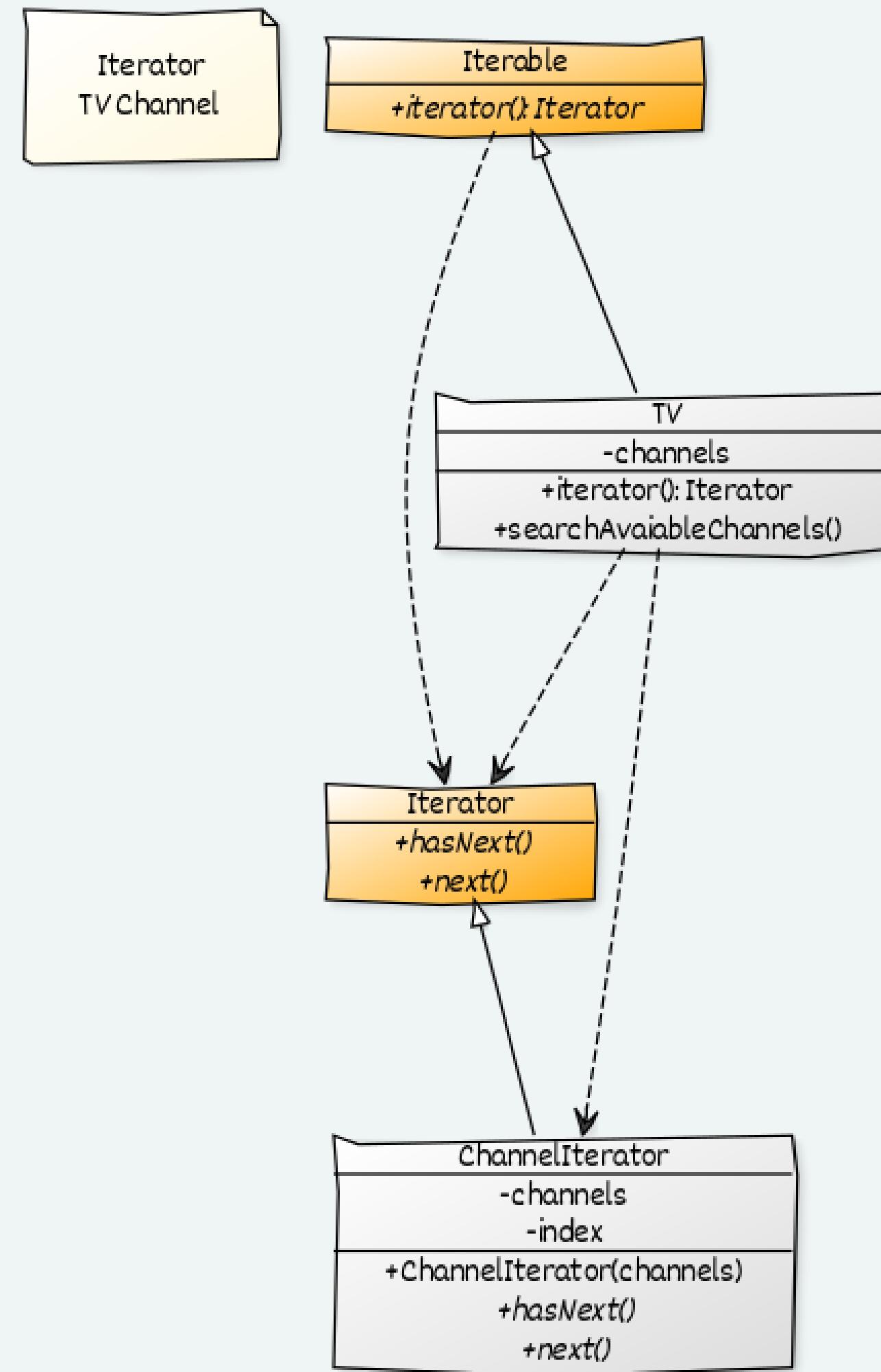
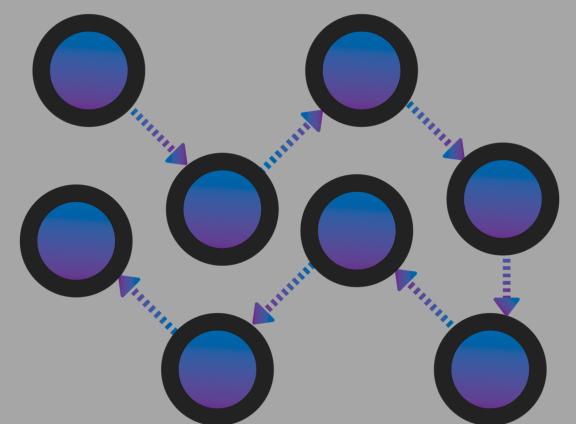




Prover uma maneira de acessar os elementos de um objeto agregado seqüencialmente sem expor sua representação interna.

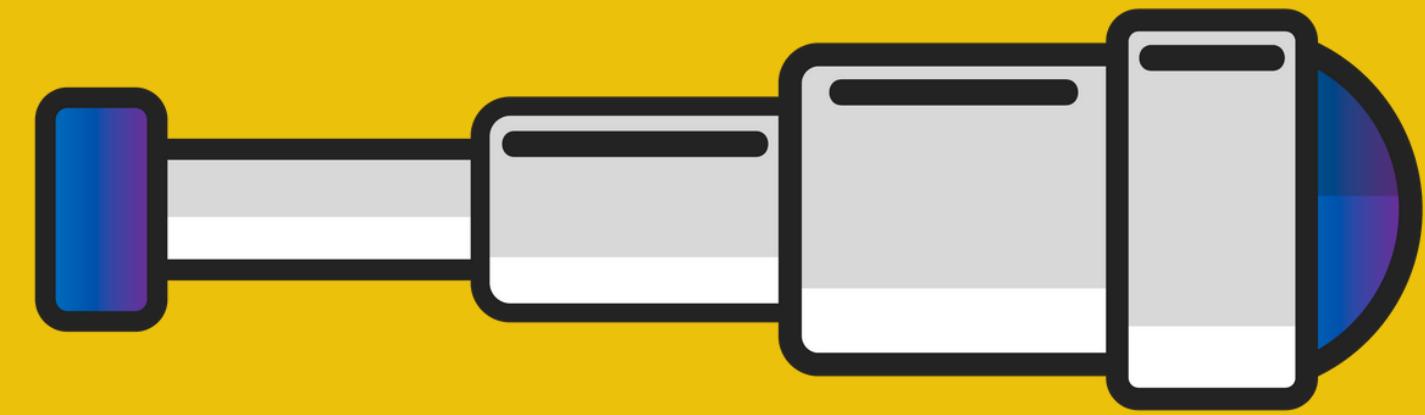
GOF

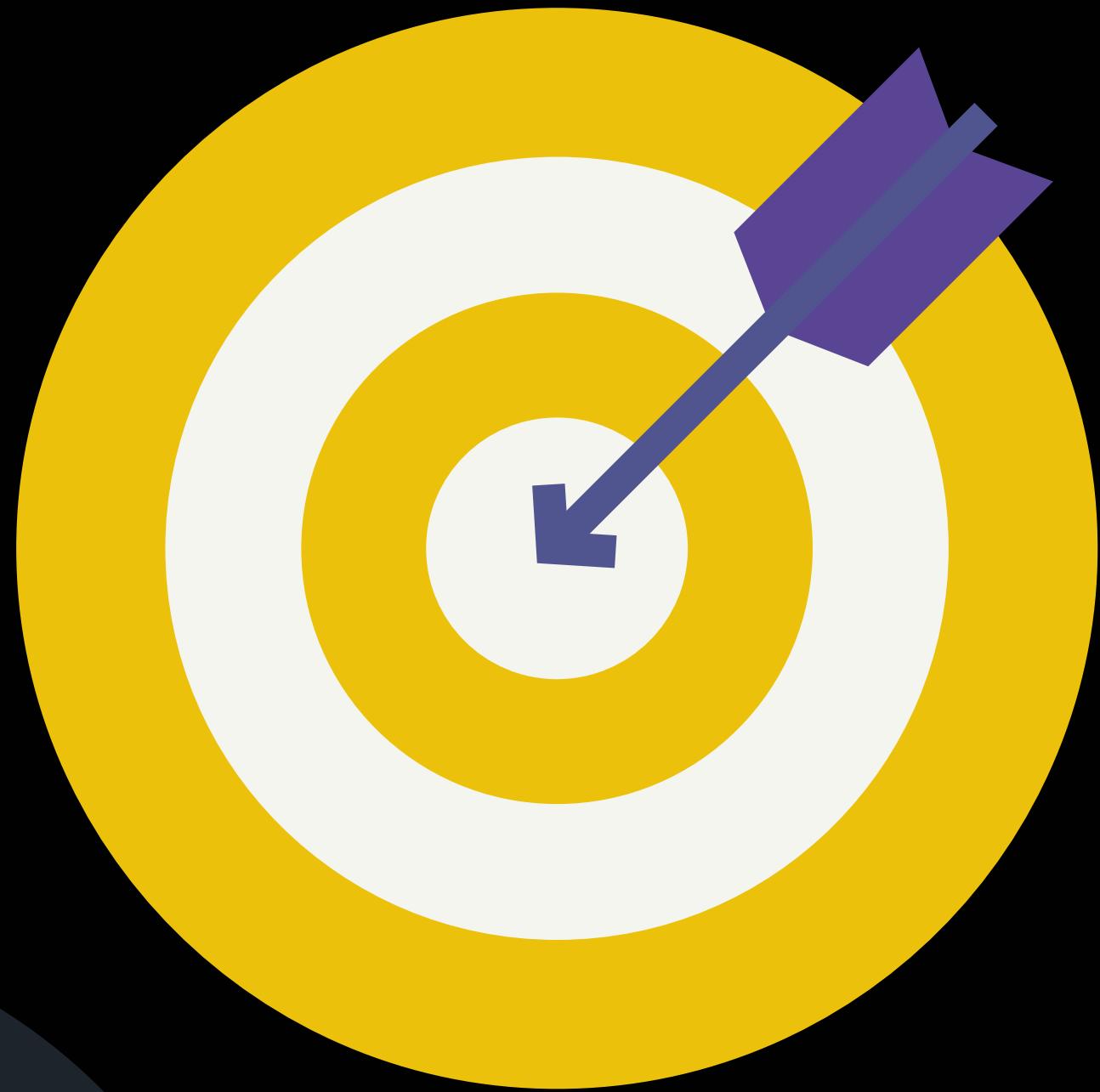




AULA OBSERVER

DON'T CALL ME, I'LL CALL YOU!





OBJETIVOS

- APRESENTAR O PROBLEMA GERAL
- APRESENTAR UMA SOLUÇÃO UTILIZANDO O OBSERVER
- APRENDER A REDUZIR A QUANTIDADE DE REQUISIÇÕES APENAS COM A TROCA DE RESPONSABILIDADE

PROBLEMAS

- COMO POSSO MODELAR UM RELACIONAMENTO 1-N SEM DEIXAR TODOS ELES ACOPLADOS?
- COMO UM OBJETO PODE NOTIFICAR OUTROS OBJETOS QUANDO NECESSÁRIO?



SOLUÇÃO

- CRIAR UMA ESTRUTURA DE OBSERVER
PARA QUE ELE POSSA NOTIFICAR TODOS
OS OBJETOS QUE SOLICITARAM SER
AVISADOS QUANDO UM DETERMINADO
EVENTO OCORRA

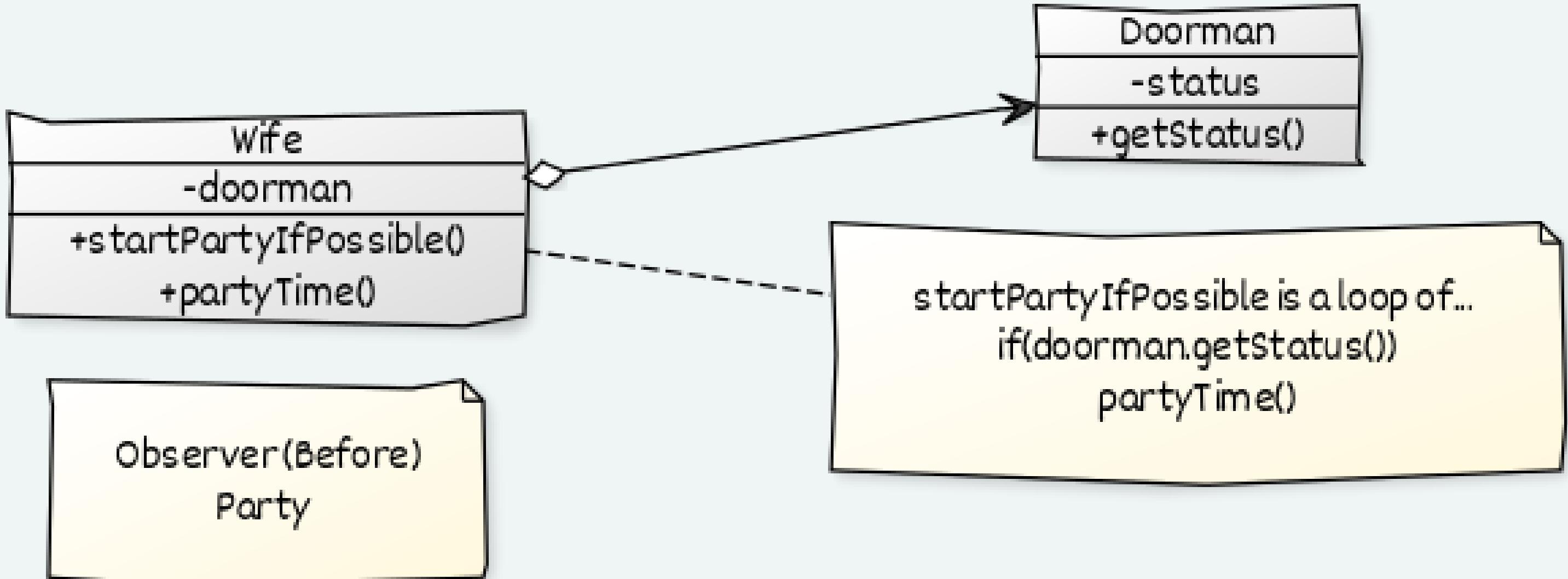




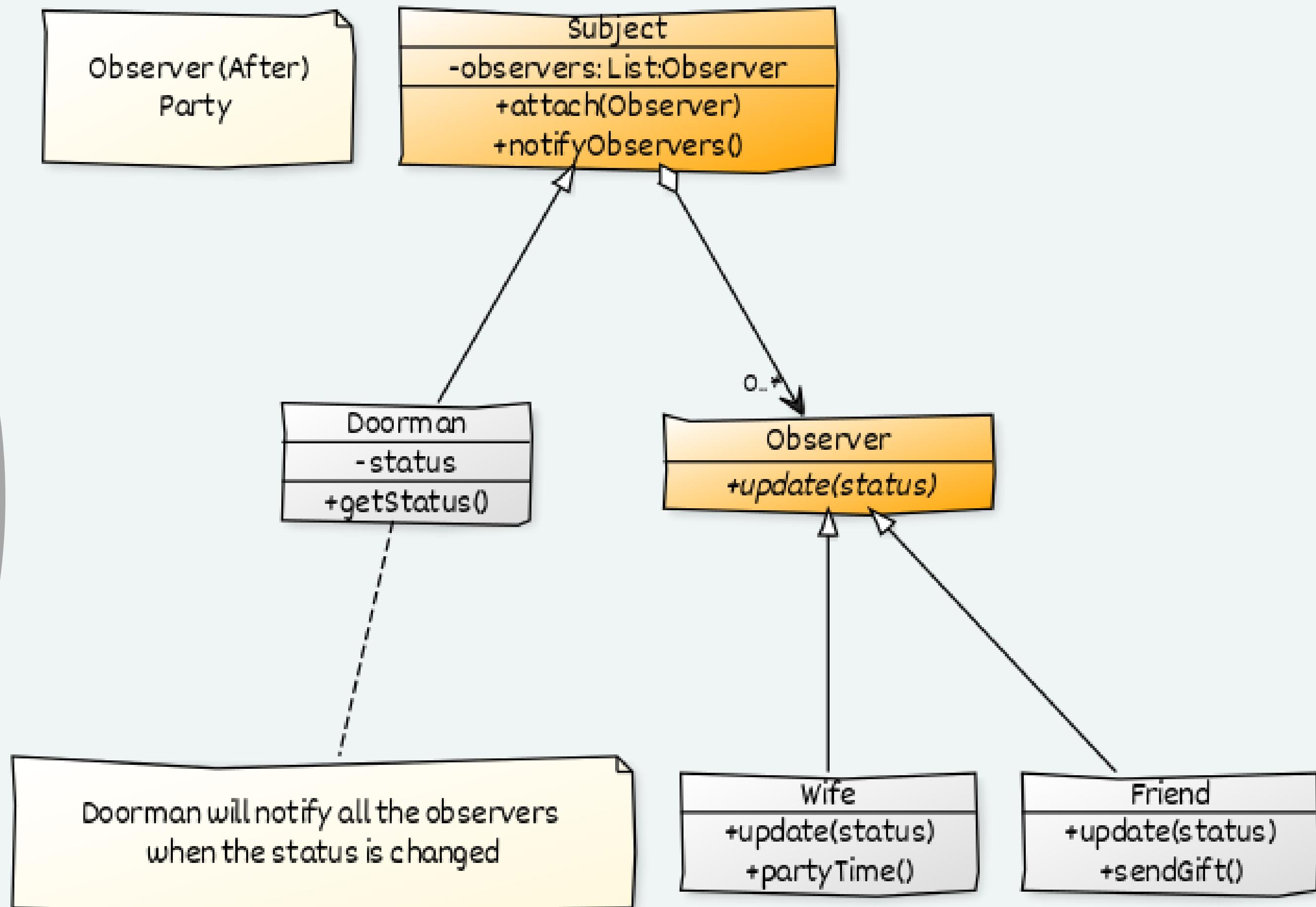
Definir uma dependência um-para-muitos entre objetos para que quando um objeto mudar de estado, todos os seus dependentes sejam notificados e atualizados automaticamente.

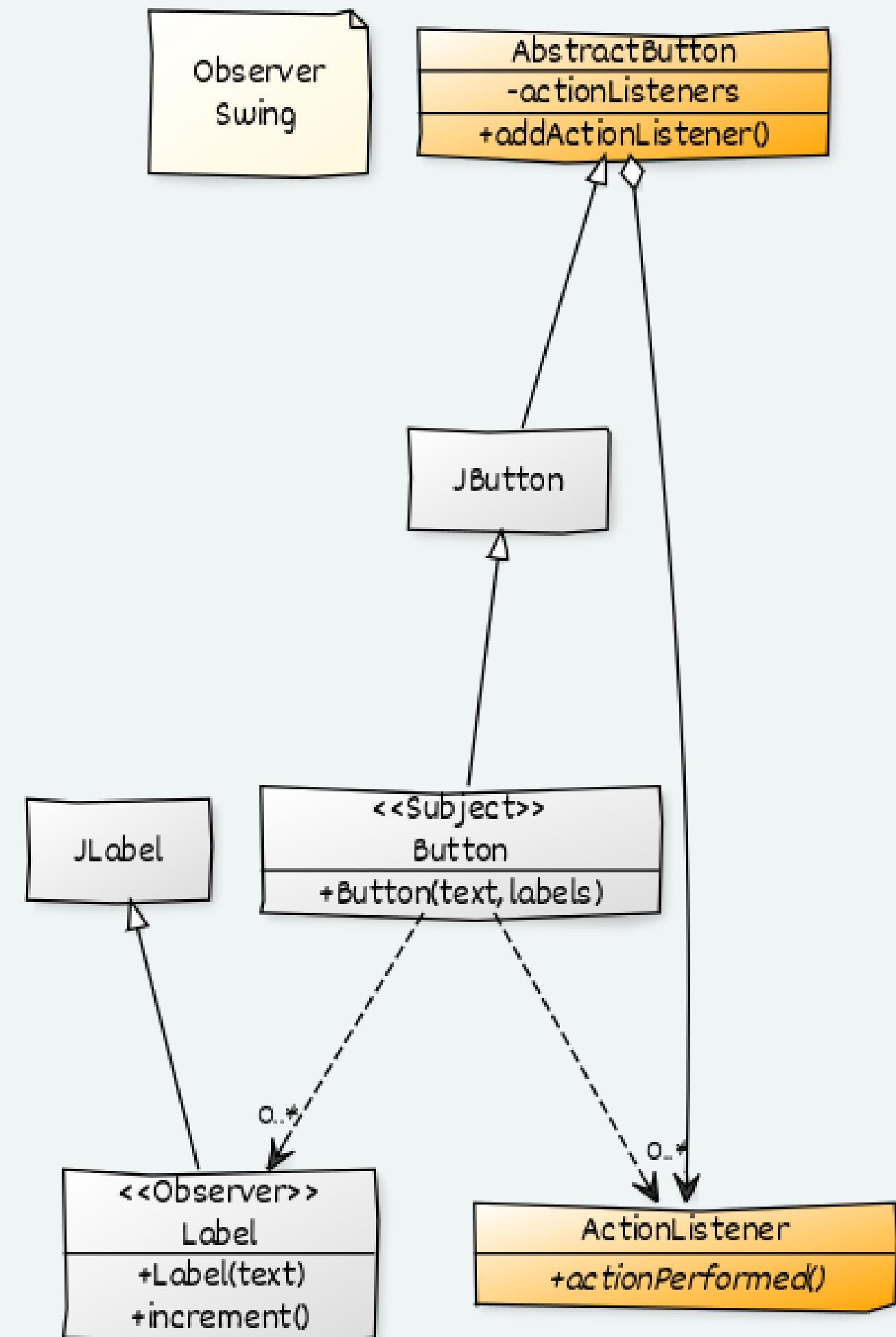


GOF



CREATED WITH YUML





AULA STATE

A MÁQUINA DE TURING





OBJETIVOS

- APRESENTAR O PROBLEMA GERAL
- APRESENTAR UMA SOLUÇÃO UTILIZANDO O STATE
- VER COMO JOGOS PODEM SER REPRESENTADOS COM ESTE PADRÃO

PROBLEMAS

- COMO POSSO ALTERAR O
COMPORTAMENTO DE UM OBJETO
QUANDO SEU ESTADO INTERNO MUDA?
- COMO PERMITIR QUE NOVOS
COMPORTAMENTOS SEJAM ADICIONADOS
E INTEGRADOS COM OS DEMAIS?



SOLUÇÃO

- MODELAR OS COMPORTAMENTOS POSSÍVEIS ATRAVÉS DE STATES
- DEFINIR COMO SERÃO REALIZADAS AS MUDANÇAS DE ESTADOS
- CADA STATE IRÁ TOMAR CONTROLE DA EXECUÇÃO DE ACORDO COM O ESTADO INTERNO DO OBJETO

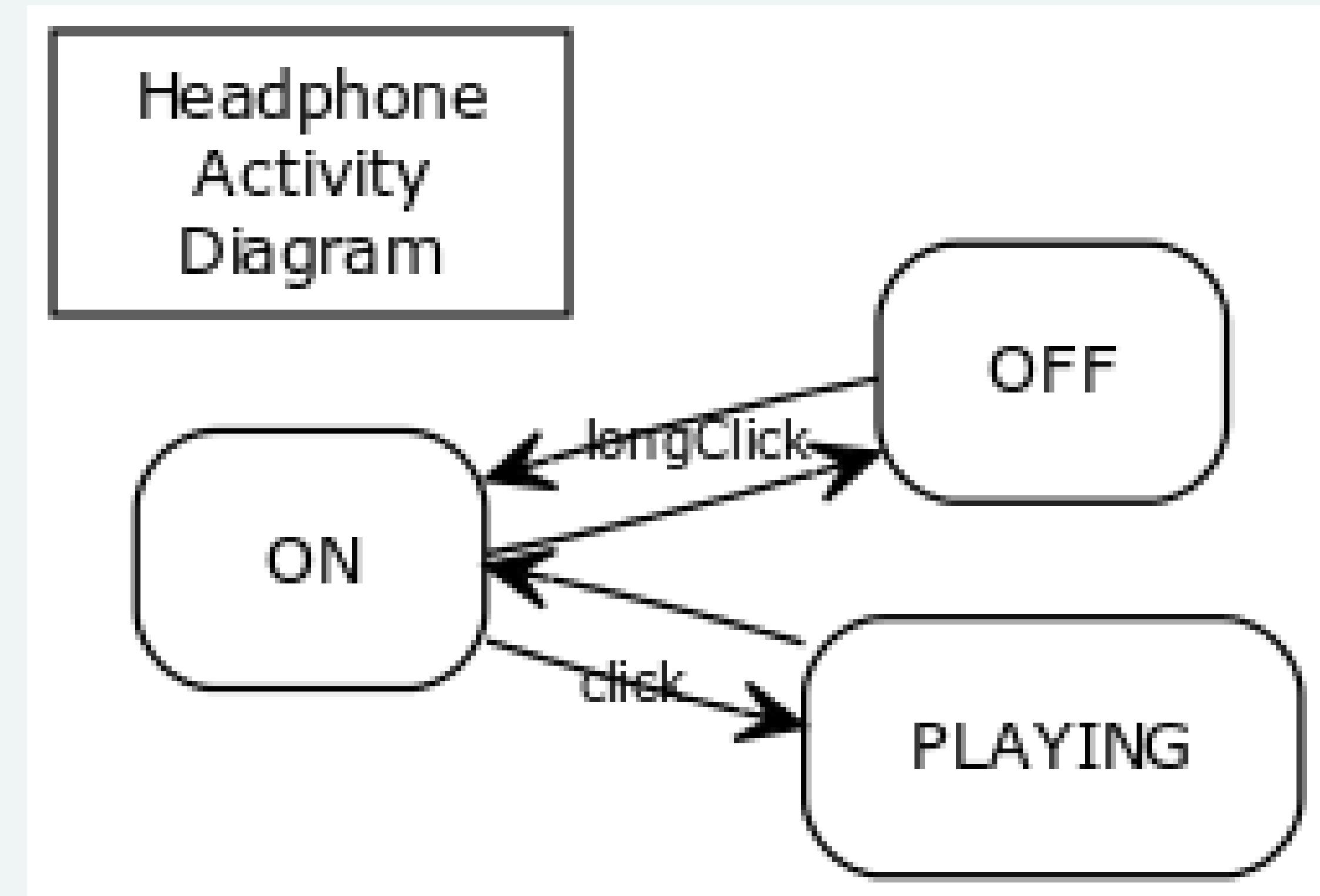




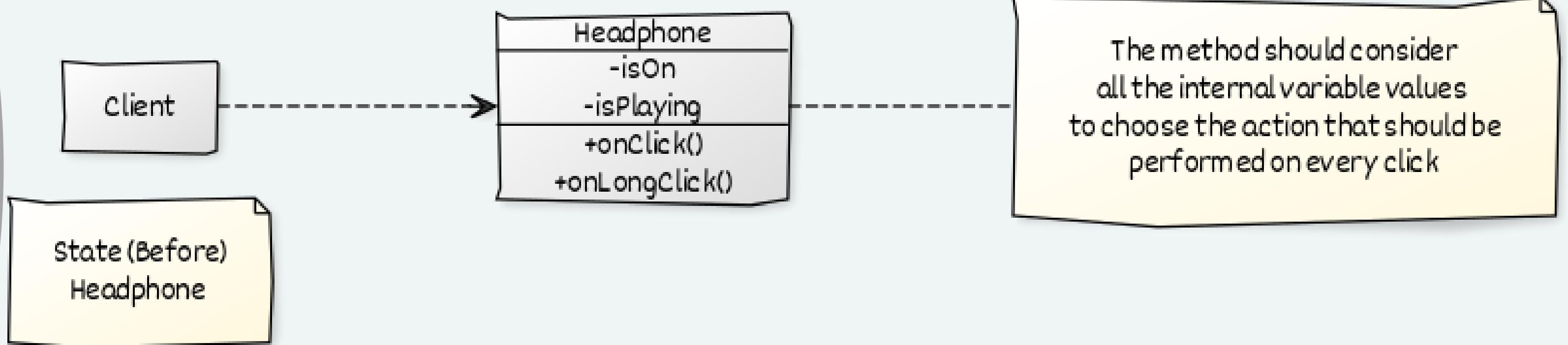
Permitir a um objeto alterar o seu comportamento quanto o seu estado interno mudar. O objeto irá aparentar mudar de classe.

GOF

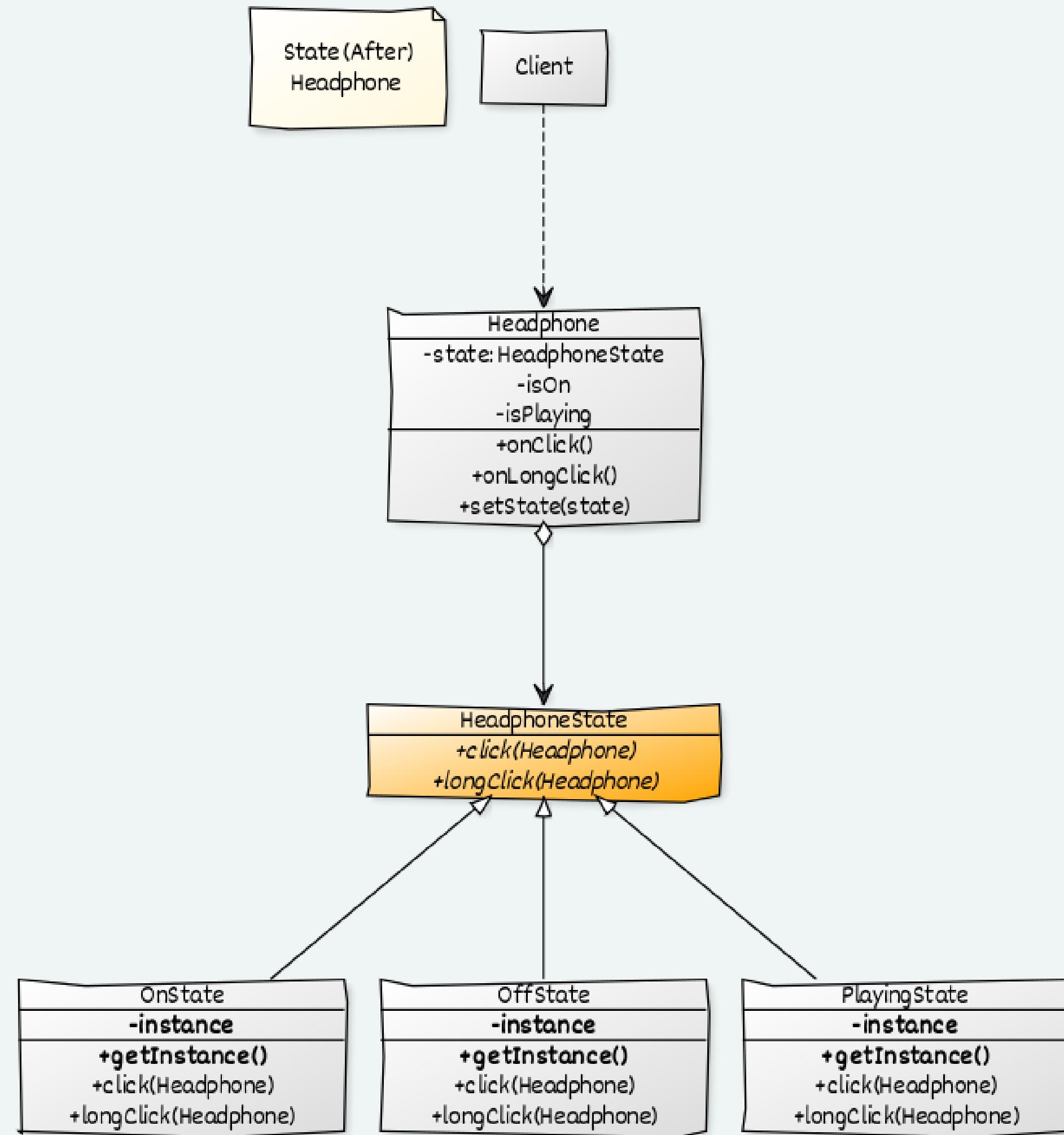
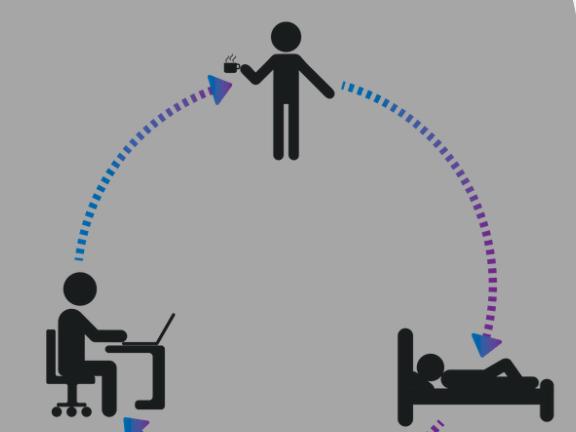


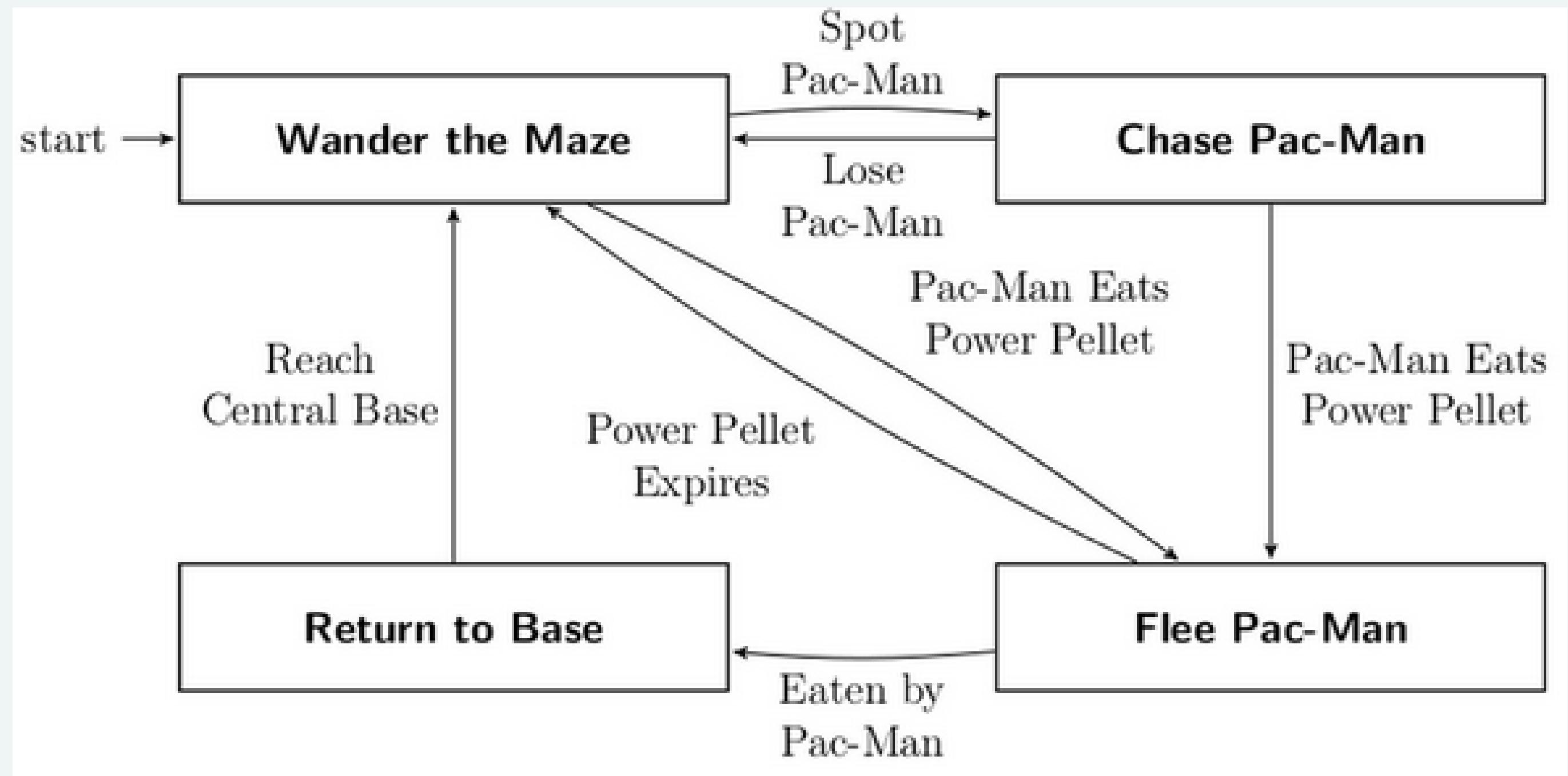


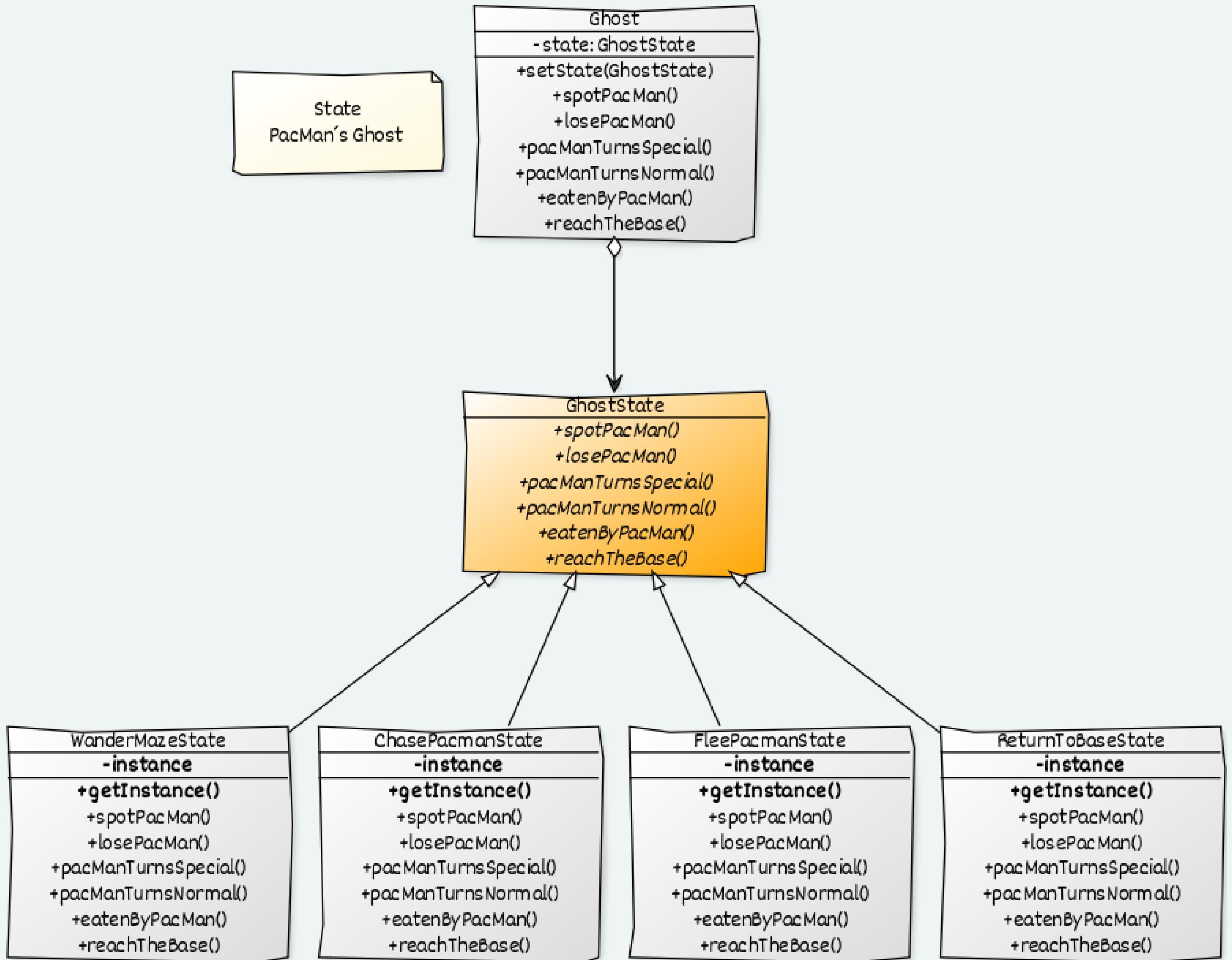
CREATED WITH YUML



CREATED WITH YUML







AULA STRATEGY

DO GREGO STRATEGY





OBJETIVOS

- APRESENTAR O PROBLEMA GERAL
- APRESENTAR UMA SOLUÇÃO UTILIZANDO O STRATEGY
- FILOSOFAR UM POUCO SOBRE A SUA EXISTÊNCIA

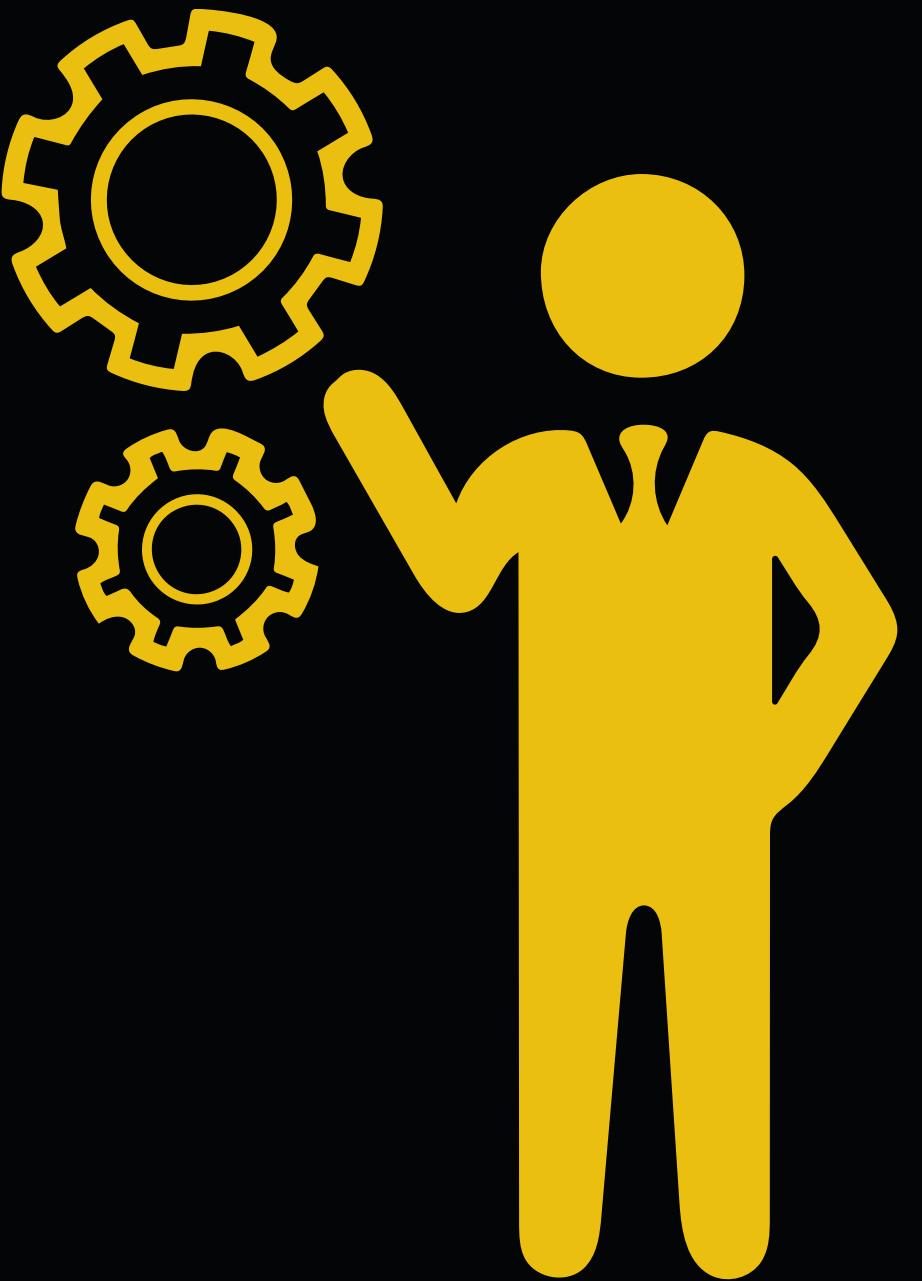
PROBLEMAS

- COMO UMA CLASSE PODE UTILIZAR UM ALGORITMO DEFINIDO DINAMICAMENTE?
- COMO POSSO SELECIONAR E TROCAR UMA LÓGICA EM TEMPO DE EXECUÇÃO?



SOLUÇÃO

- ENCAPSULAR OS ALGORITMOS POSSÍVEIS PARA O MESMO PROBLEMA EM UMA ESTRUTURA DE STRATEGY
- O CLIENTE IRÁ DELEGAR A EXECUÇÃO PARA ESTAS ESTRATÉGIAS AO INVÉS DE POSSUIR TODA A LÓGICA INTERNAMENTE

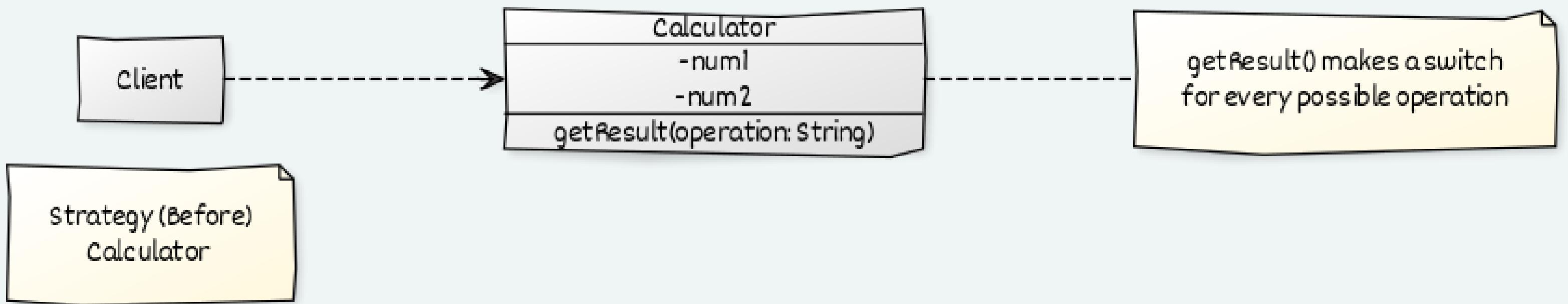




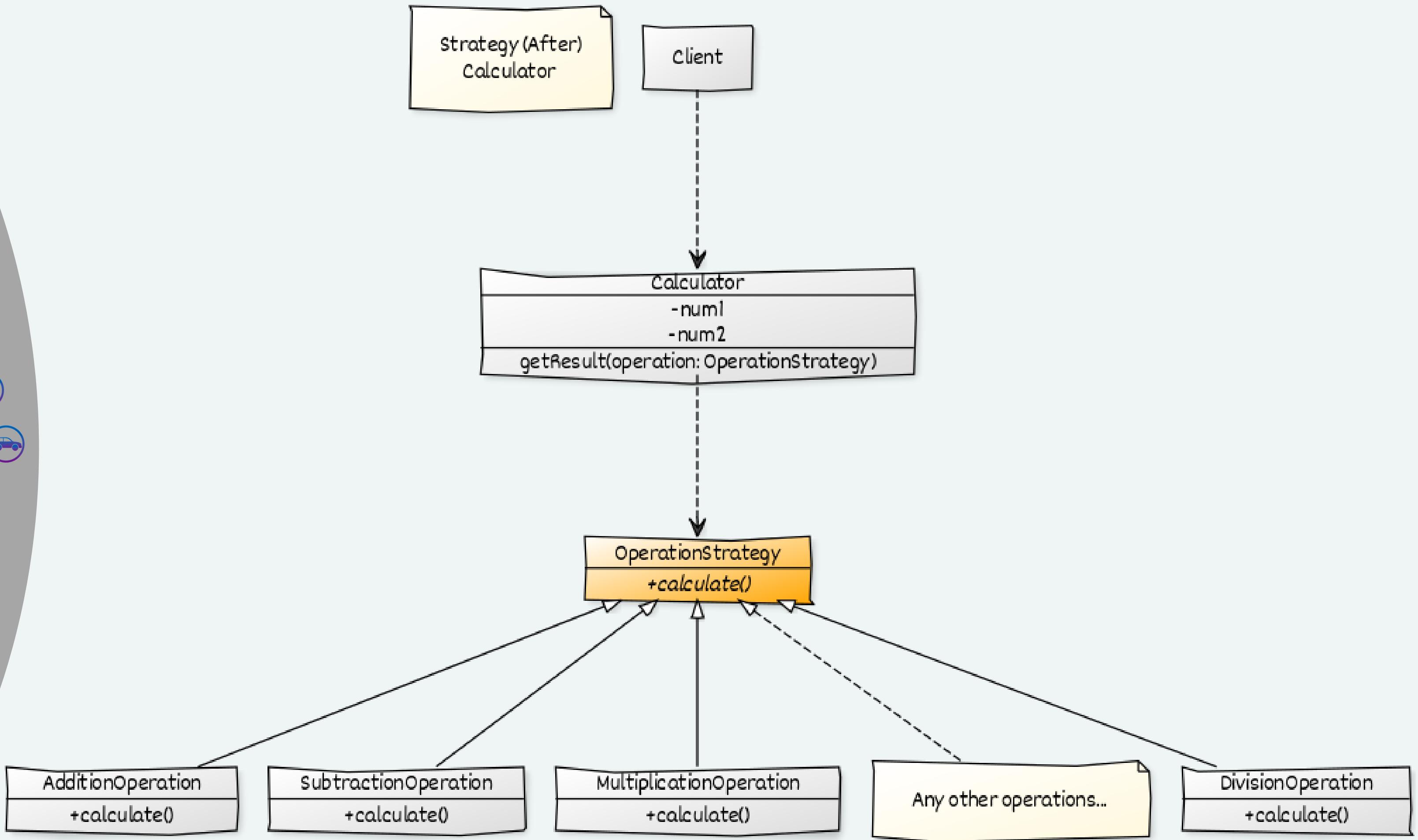
Definir uma família de algoritmos, encapsular cada um, e fazê-los intercambiáveis. Strategy permite que algoritmos mudem independentemente entre clientes que os utilizam.

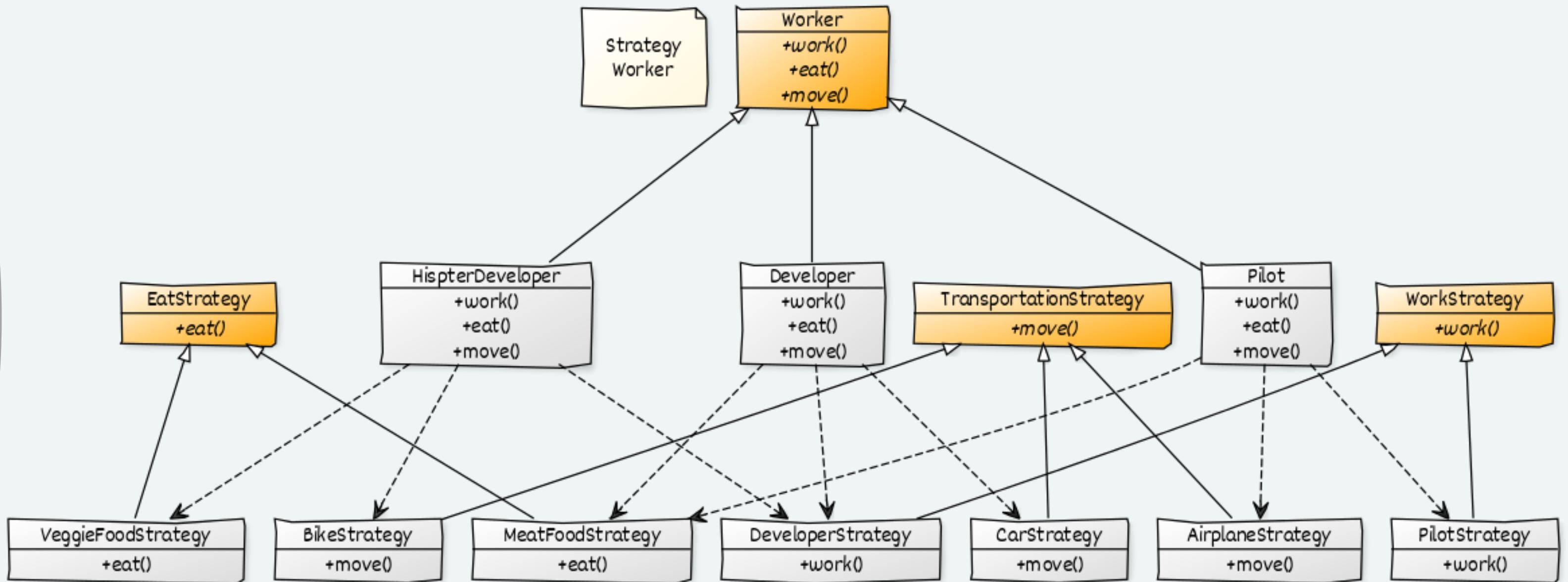
GOF





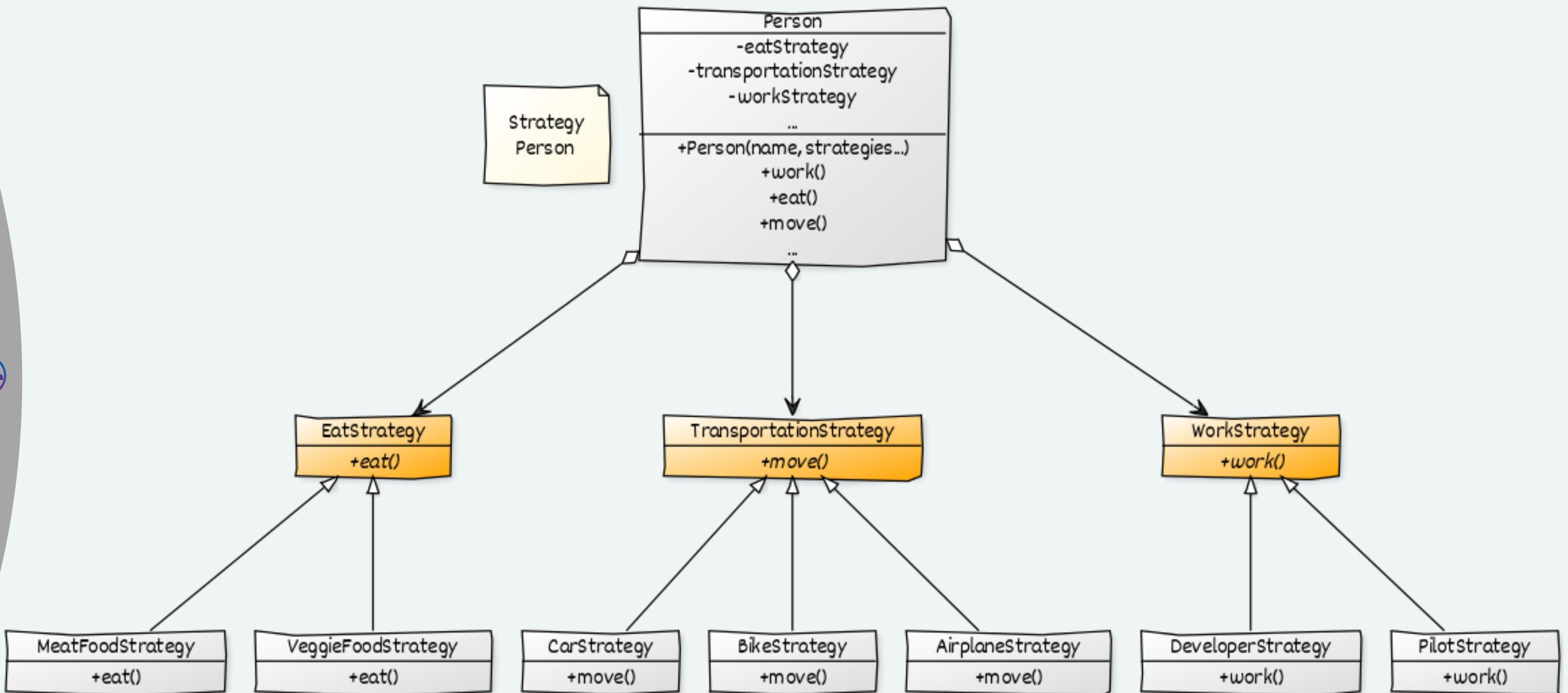
CREATED WITH YUML





CREATED WITH YUML

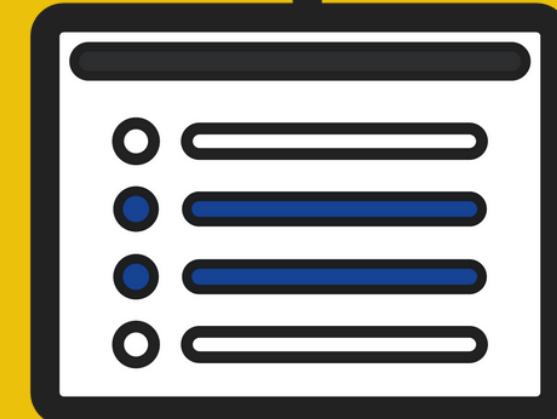
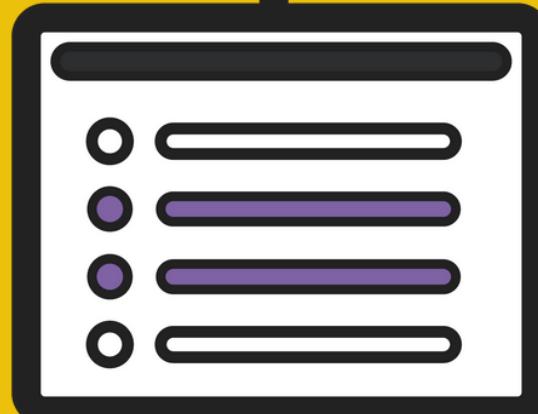
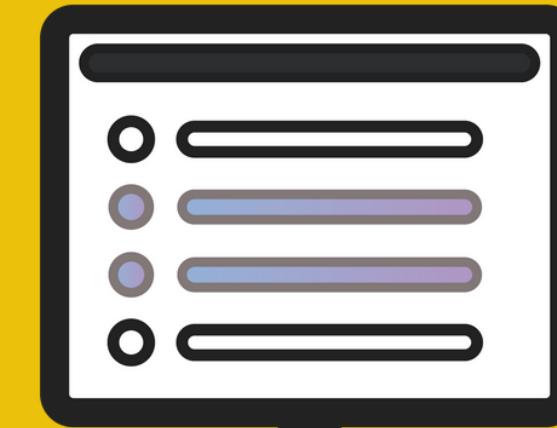


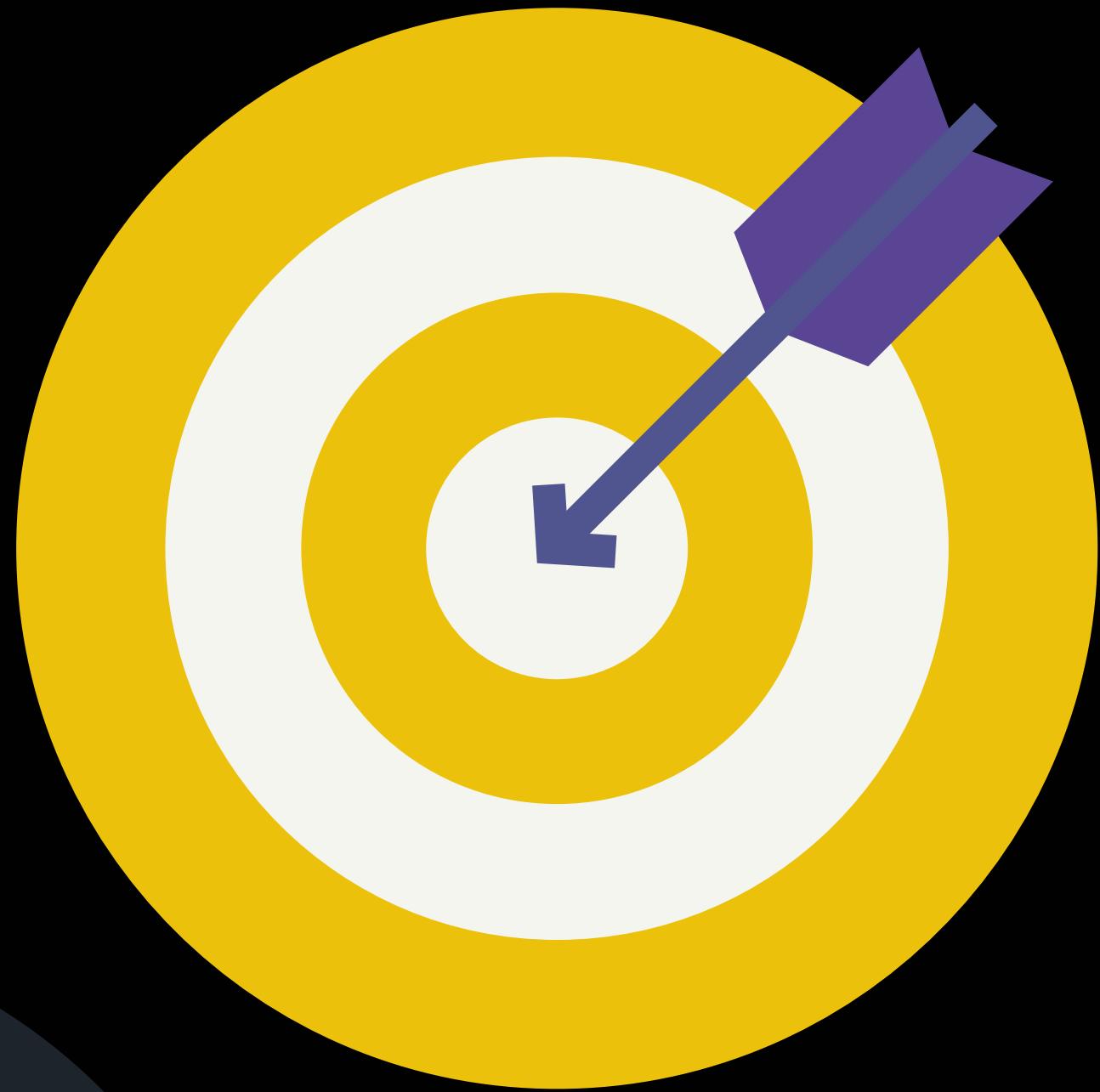


CREATED WITH YUML

AULA TEMPLATE METHOD

NÃO SEI SE PERCEBEU, MAS ISSO AQUI É UM
TEMPLATE...





OBJETIVOS

- APRESENTAR O PROBLEMA GERAL
- APRESENTAR UMA SOLUÇÃO UTILIZANDO O TEMPLATE METHOD
- APRESENTAR UMA FORMA DE MODELAR LÓGICAS COMPLEXAS REAPROVEITANDO MUITO CÓDIGO

PROBLEMAS

- COMO POSSO UNIR PARTES DE UM CÓDIGO QUE NÃO VARIAM COM PARTES VARIÁVEIS?
- COMO POSSO ALTERAR CERTOS PONTOS DO CÓDIGO MANTENDO UMA ESTRUTURA GERAL?



SOLUÇÃO

- DEFINIR UMA ABSTRAÇÃO COM TODAS OS PONTOS QUE PODEM SER VARIADOS
- CRIAR UMA TEMPLATE QUE CONTENHA AS PARTES FIXAS E POSSUA PONTOS DE CHAMADAS PARA AS PARTES VARIÁVEIS

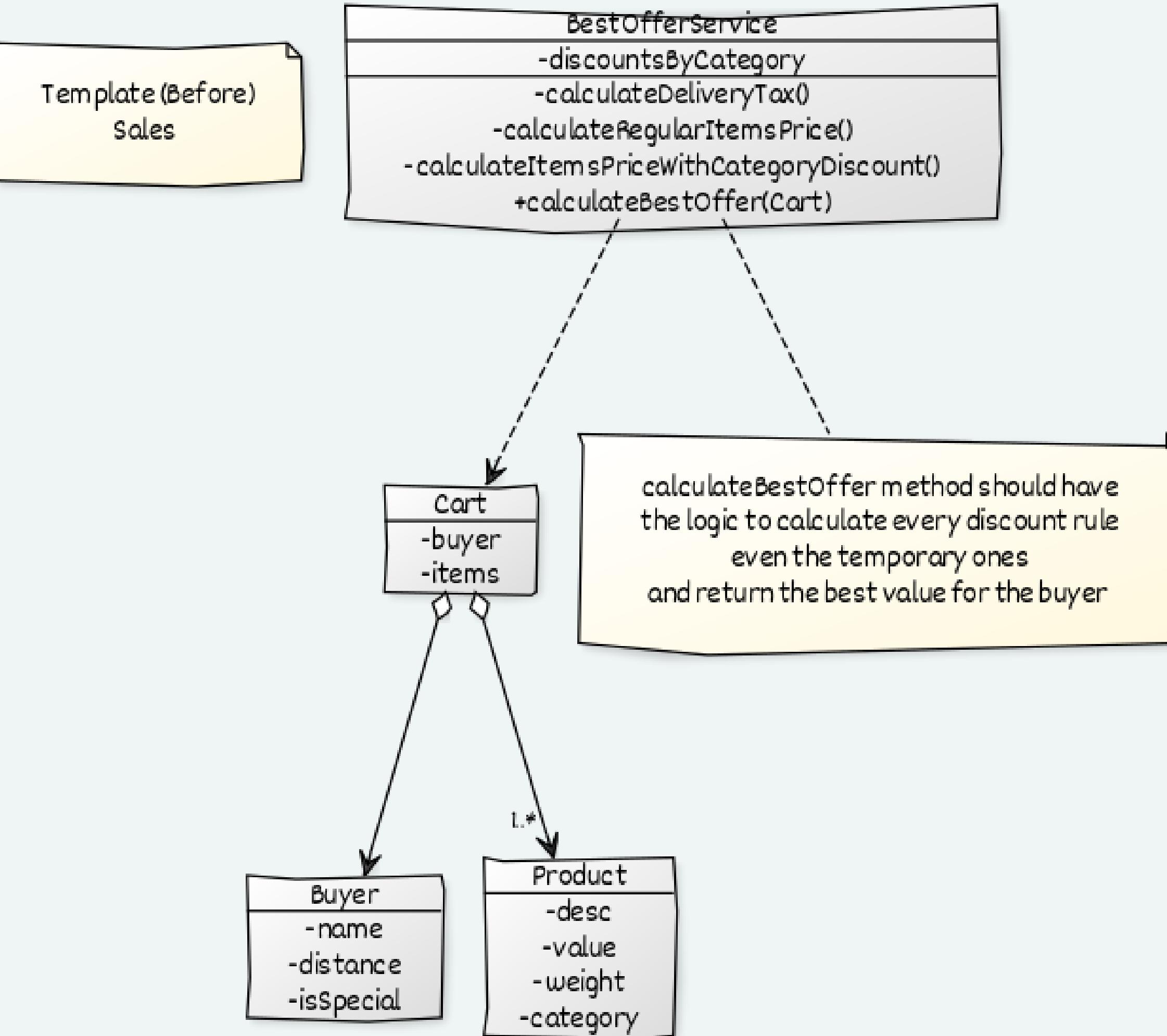




Definir o esqueleto de um algoritmo dentro de uma operação, deixando alguns passos a serem preenchidos pelas subclasses. Template Method permite que suas subclasses redefinam certos passos de um algoritmo sem mudar sua estrutura.

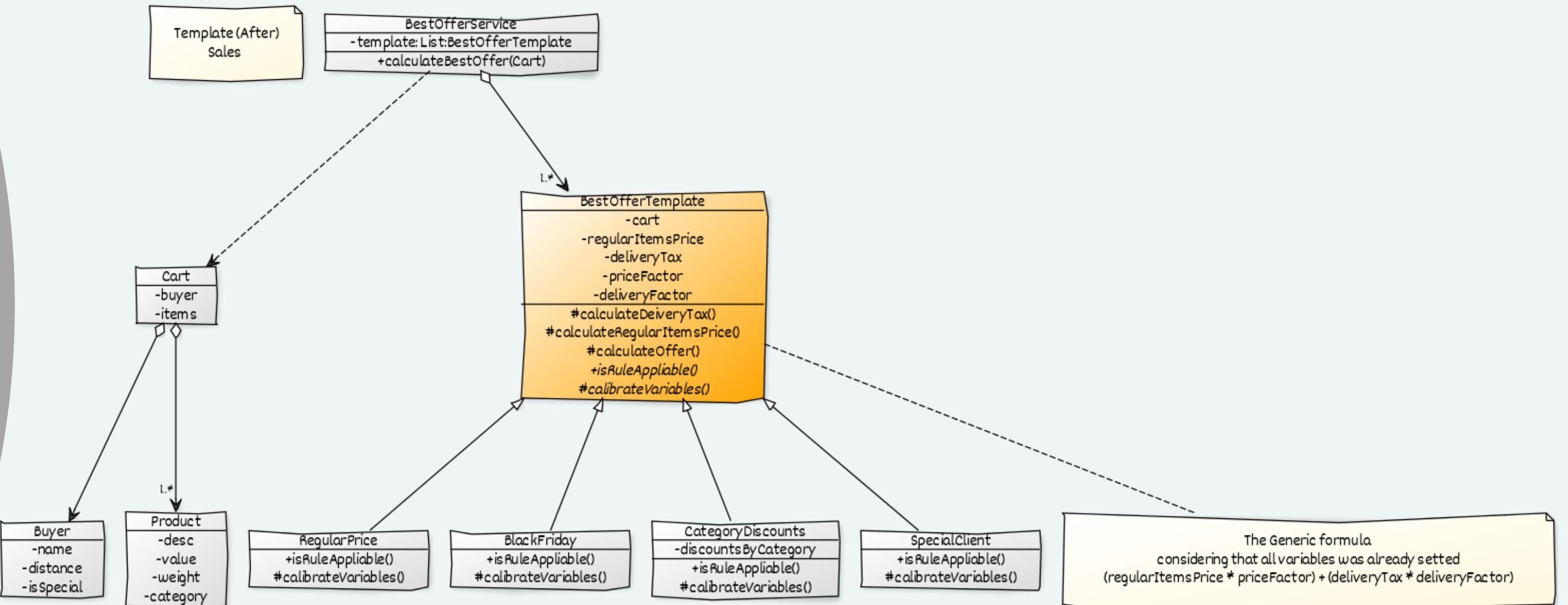


GOF

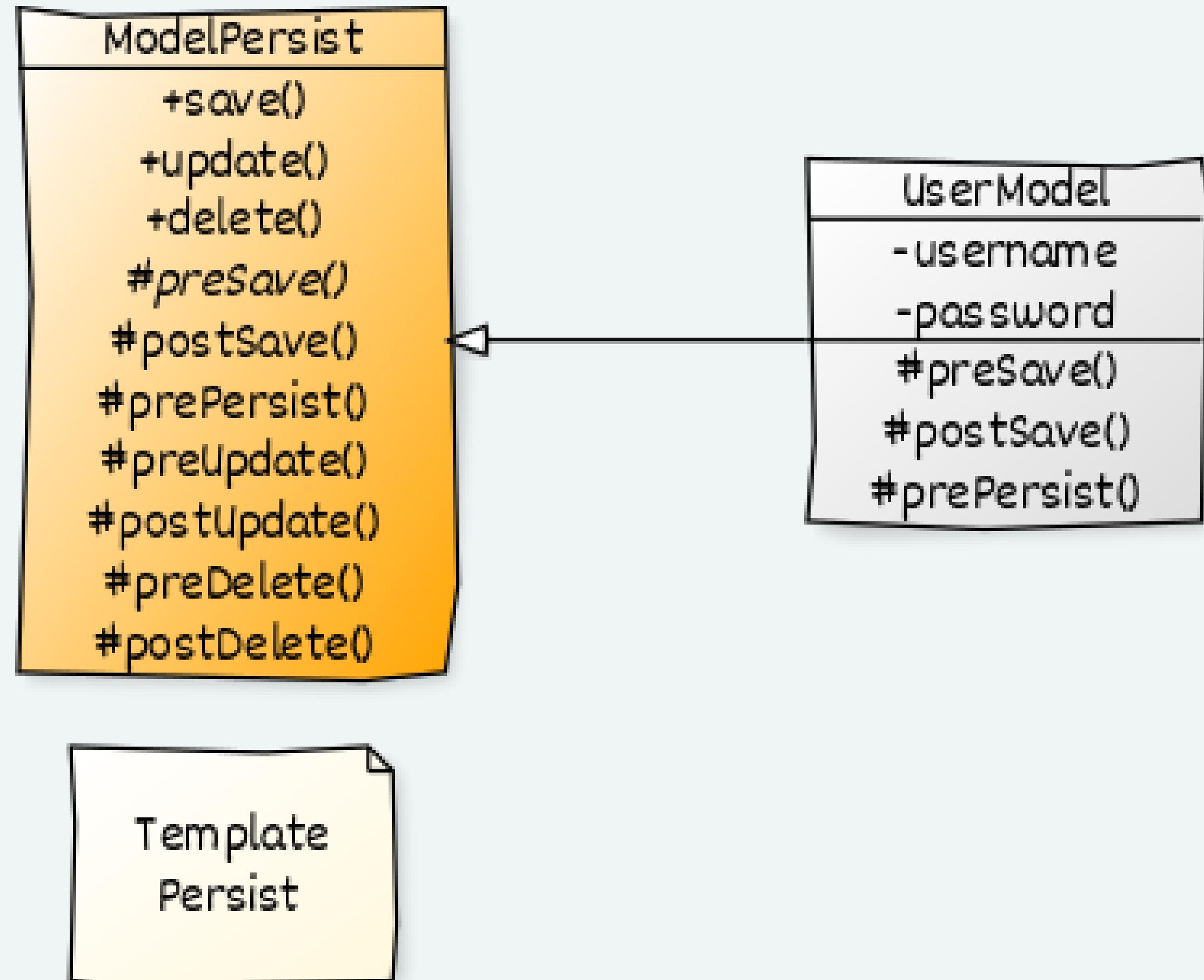


Template (Before)
Sales





CREATED WITH YUML



OBRIGADO!

CHEGAMOS NO FINAL DO NOSSO
CURSO, UHUUUU!

LEMBRE-SE DE EMITIR O
CERTIFICADO DO CURSO E SE
AINDA NÃO AVALIOU, ESTE É UM
BOM MOMENTO.

