Universidade de Trás-os-Montes e Alto Douro
Departamento de Engenharias

utad

Mestrado em Engenharia Informática

Arquitetura e Padrões de Software Exercícios práticos

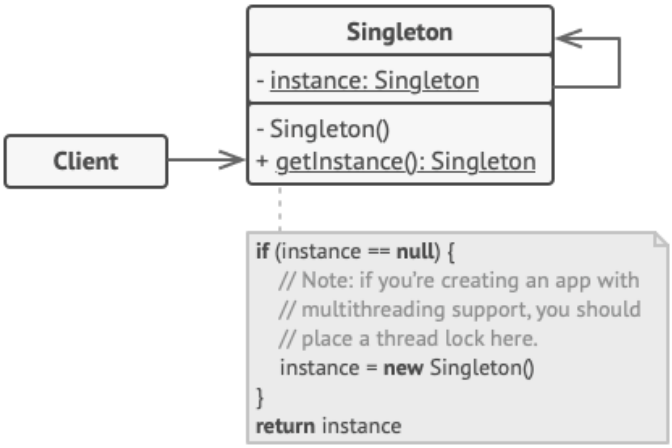
Background

1. Desenhe e codifique um conjunto de classes que permitam demonstrar as características da Programação Orientada a Objetos, nomeadamente: abstração, encapsulamento, herança, composição e polimorfismo.

2. Fazendo uso da API Reflect da biblioteca do Java, use técnicas de metaprogramação para:
   2.1. Consultar as definições de uma classe (atributos declarados, atributos públicos, superclasse, interfaces implementadas, exceções lançadas).
   2.2. Consultar os construtores de uma classe.
   2.3. Consultar os métodos de uma classe e correspondentes tipos dos atributos.
   2.4. Executar um dos métodos da classe passando-lhe o(s) parâmetro(s) necessário(s).
   2.5. Alterar o valor (estado) de um qualquer atributo de uma instância da classe.
   2.6. Instanciar um objeto de uma classe identificada pelo utilizador em *runtime*.
   2.7. Executar um método de uma classe, identificado pelo utilizador em *runtime*, devendo passar os parâmetros inseridos igualmente pelo utilizador.
   2.8. Repita o exercício anterior fazendo tratamento das exceções que o método possa gerar.

Padrões de criação

3. Singleton

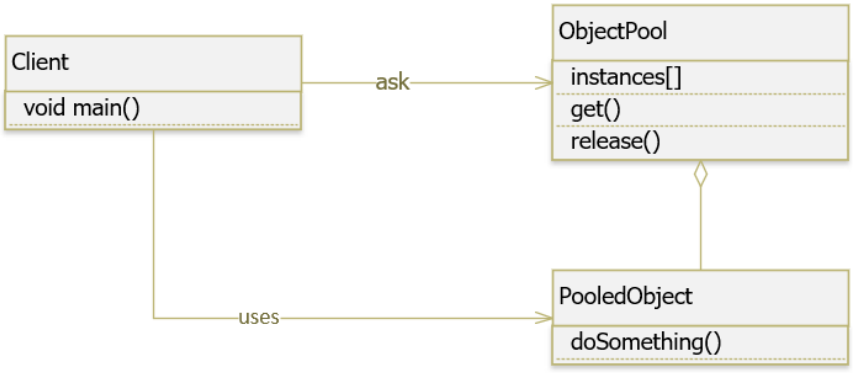


3.1. Demonstre a utilização do padrão Singleton;

3.2. Implemente um sistema de gestão do acesso a um repositório (base de dados) a ser usado num sistema informático. O sistema usa uma única conexão à base de dados e esta permite efetuar vários pedidos em simultâneo.

3.3. Implemente um sistema de *logging* a ser usado num sistema informático. O sistema permite registar mensagens em três níveis de importância (ALERTA, ERRO e FATAL) e para um dispositivo configurável (FILE, STDERR, TCP-PORT).

4. Object Pool



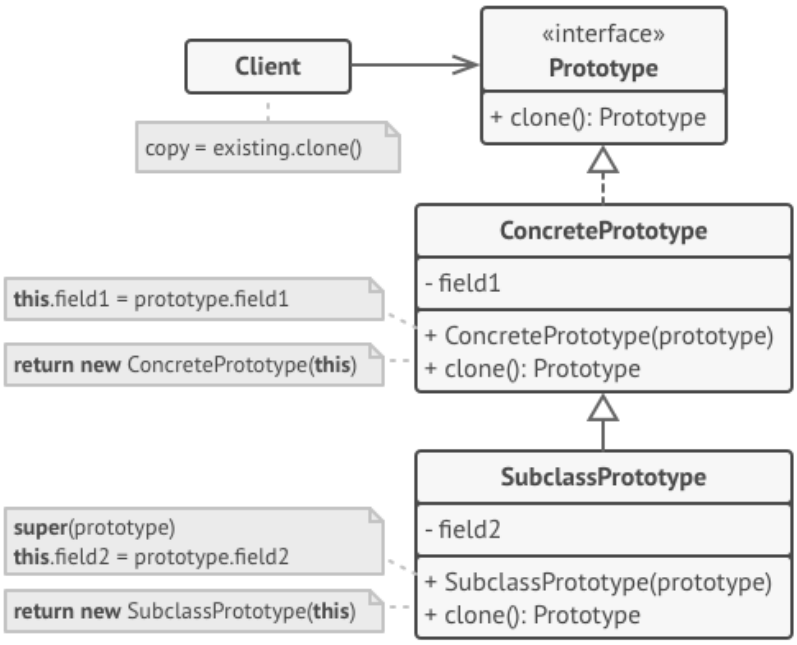
4.1. Implemente um sistema de impressão centralizado, composto por várias impressoras todas iguais. Cada impressora só pode ser usada por um cliente de cada vez. Para efetuar uma impressão, o cliente deve requisitar uma impressora e libertar a mesma após terminar.

4.2. Um sistema informático usa um repositório em base de dados. O acesso à base de dados é efetuado através de ligações num máximo de 4 em simultâneo.

Implemente um sistema de gestão de conexões de acesso à base de dados. Cada conexão só pode ser usada por um cliente de cada vez. A conexão deve ser requisitada para utilização e posteriormente e libertada quando não for mais necessária.

Se um cliente solicitar uma conexão e não houver disponíveis será reportado um erro ao mesmo.

5. Prototype



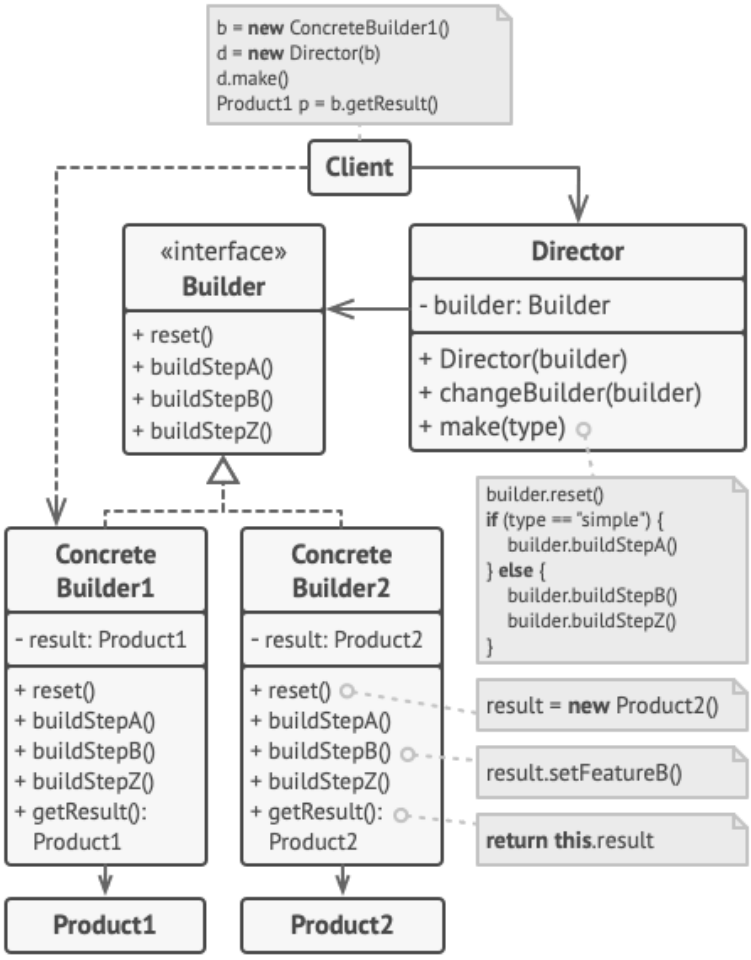
5.1. Demonstre a utilização do padrão Prototype confirmando a identicidade dos objetos obtidos por clonagem.

5.2. Um software de um jogo utiliza uma classe para caracterizar os jogadores *bot* inimigos. Esses jogadores possuem uma estrutura complexa que é constituída pelas características físicas de apresentação (vestimentas) e de comportamento (inteligência, deslocamento, etc.) que são parametrizadas aquando da sua criação.

Em cada nível do jogo, os inimigos surgem de forma contínua ao longo da linha temporal da cena e têm sempre as mesmas características.

Considerando a problemática da criação dos elementos descritos durante o decorrer do jogo, implemente uma solução para o problema apresentado.

6. Builder



6.1. Um software de gestão de informação necessita de proceder a consultas a dispositivos remotos diversos.

Independentemente do dispositivo consultado, para conseguir efetuar as consultas, estão previstos os seguintes passos:

1 - Abrir conexão;

2 - Efetuar pedido;

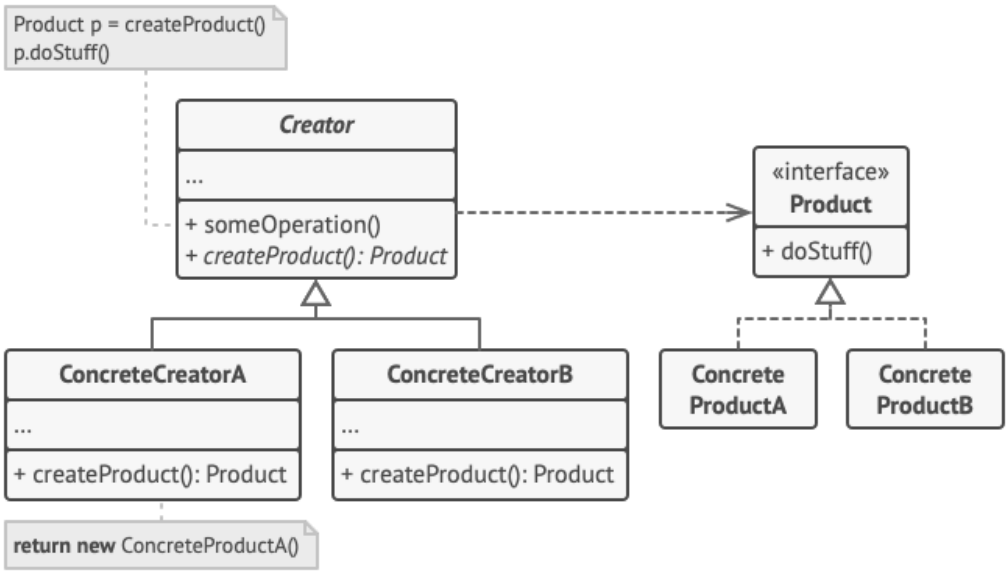
3 - Armazenar resposta;

4 - Fechar conexão;

As consultas podem ser realizadas em partes distintas do software e implicam a execução da totalidade dos passos.

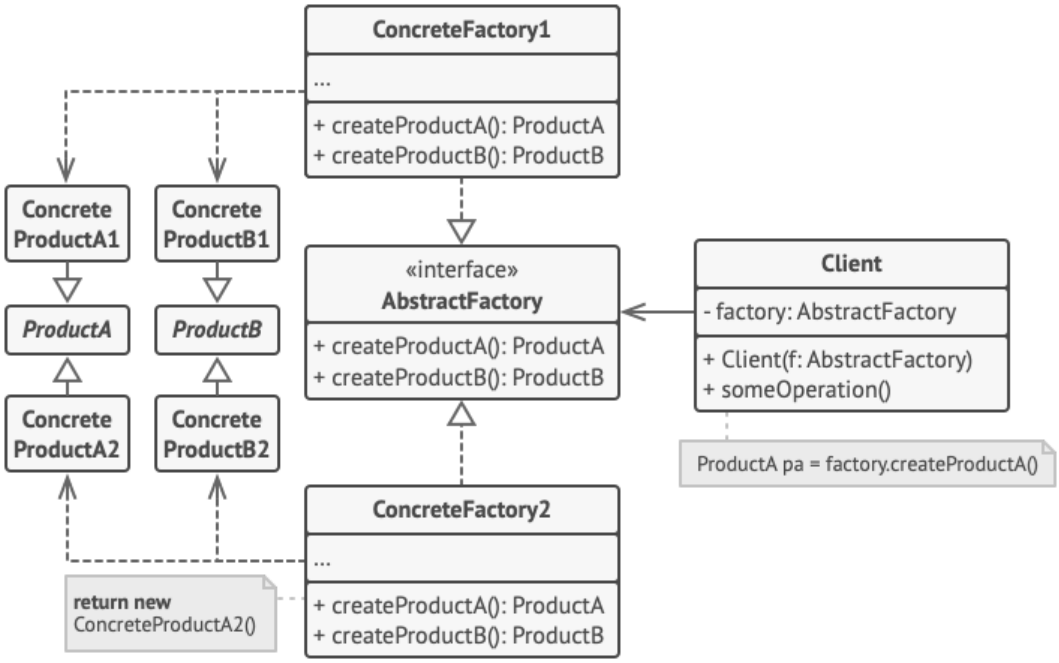
Implemente uma solução para o problema apresentado.

7. Factory Method



7.1. Demonstre a utilização do padrão Factory Method implementando métodos para a criação de veículos ligeiros e veículos pesados caracterizados por um motor que os diferencia (por exemplo, na potência).

8. Abstract Factory



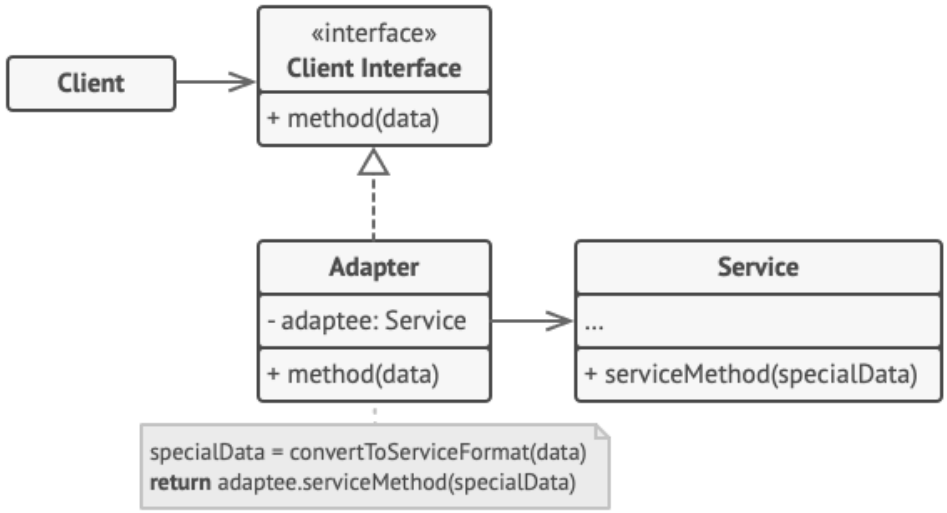
8.1. Demonstre a utilização do padrão Abstract Factory implementando, na sequência o exercício 7.1, fábricas para a criação de veículos com motor a combustão e veículos com motor elétrico.

8.2. Uma fábrica de automóveis requer o desenvolvimento de uma aplicação que permita gerir o processo de montagem dos automóveis. Cada automóvel necessita de pelo menos 3 componentes: Chassi (Citadino, SUV, Comercial), motor (gasolina, gasóleo, elétrico) e rodas (17 polegadas, 19 polegadas). A fábrica monta três modelos de carros: C1 (chassi Citadino, motor a gasolina e rodas 17 polegadas), Exclusive (chassi SUV, motor elétrico, rodas 19 polegadas) e Transport (chassi Comercial, motor a gasóleo e rodas 19 polegadas).

Implemente uma solução para o problema apresentado, identificando os participantes no padrão, que demonstre a criação dos dois exemplos descritos.

Padrões estruturais

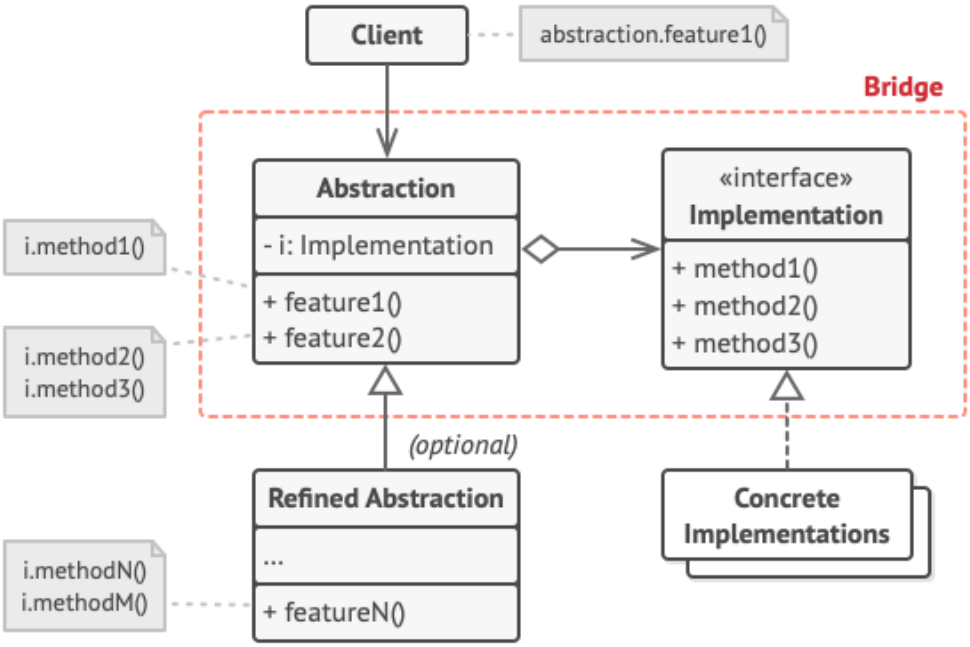
9. Adapter



9.1. Um software necessita de efetuar a ordenação dos dados que manuseia para a sua apresentação. Para efetuar a ordenação, recorre-se a bibliotecas externas que permitem a ordenação de dados numéricos recorrendo aos mais diversos algoritmos (Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, etc.). É pretendido que o software possa recorrer a qualquer uma das soluções de ordenação existentes.
Implemente uma solução para o problema apresentado, identificando os participantes no padrão.

9.2. Pretende-se implementar um sistema de pagamento numa aplicação de comércio eletrónico com faturação. O sistema deve incorporar as operações VendaCredito(quantia) e VendaDebito(quantia). Pretende-se que o sistema possa integrar várias plataformas de pagamento, nomeadamente: Cartão de Débito, Cartão de Crédito, AliPay, PayPal, entre outros.
Implemente uma solução para o problema apresentado, identificando os participantes no padrão.

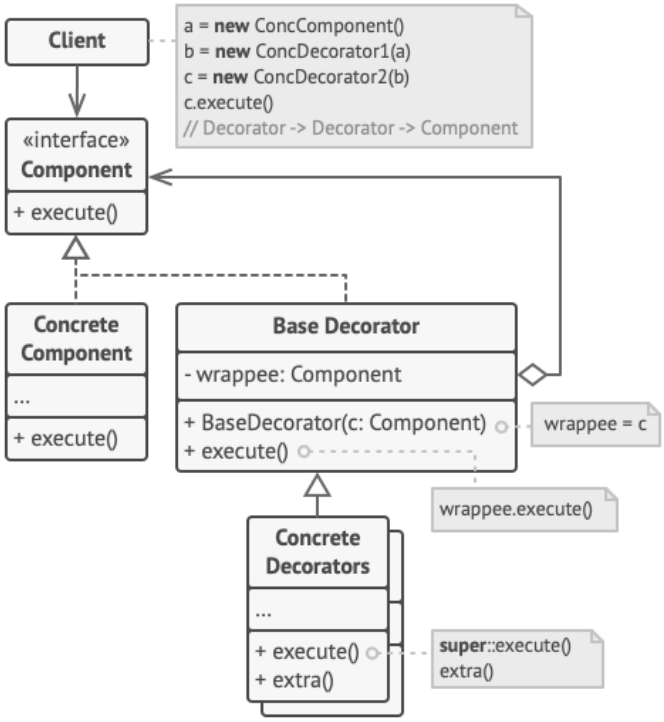
10.Bridge



10.1. Considere uma empresa que possui programadores e testadores, em que ambos são colaboradores e recebem tarefas de acordo com a área em que atuam (criação de rotinas e criação de testes, respetivamente). Suponha também que a depender de fatores externos possam vir a receber tarefas diferenciadas. Por exemplo, um programador pode criar rotinas em Java ou C# e se estiver com poucos projetos pode criar testes automatizados ou manuais. Um Testador geralmente cria testes manuais ou automatizados, mas em situações de exceção pode vir a criar rotinas Java ou C#.

Implemente uma solução para o problema apresentado, identificando os participantes no padrão.

11.Decorator



11.1. Uma loja gráfica faz estampagem em objetos que pode ser de diversos tipos (por exemplo: camisolas, canecas, etc.). Em cada um dos objetos a estampagem pode ser efetuada a uma só cor (preto) ou a várias cores (RGB).

Cada produto impresso tem um custo associado que depende do objeto estampado e do número de cores usadas na estampagem.

Desenvolva uma implementação funcional de referência para o problema apresentado e demonstre a sua utilização.

11.2. Um sistema tem um objeto que permite escrever no disco um documento. Por questões de segurança foi decidido adicionar a capacidade de encriptar o ficheiro. Por questões de espaço ocupado foi decidido comprimir o ficheiro.

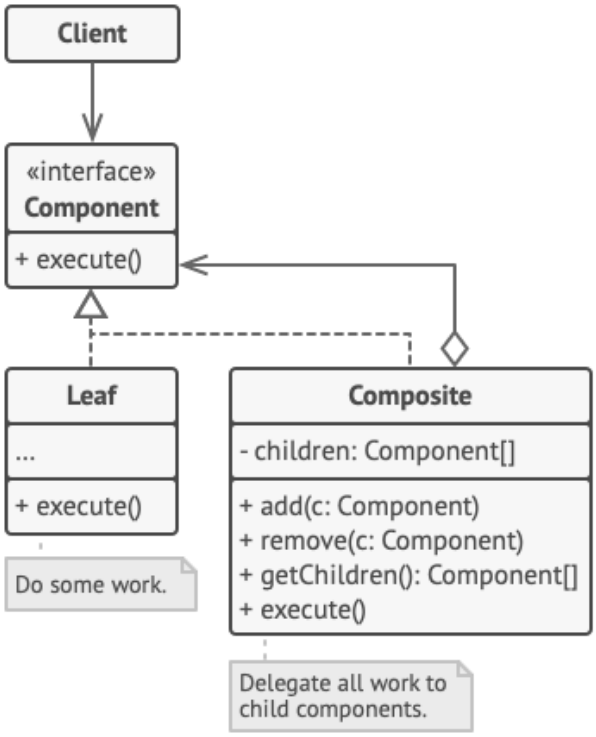
Desenvolva uma implementação funcional de referência para o problema apresentado.

11.3. Considere uma biblioteca que permite o acesso a um repositório de dados de funcionários de uma empresa através da classe seguinte:

```
public class FuncionarioStorage {
    public int Numero;
    public String Nome;
    public String Username;
    public String Password;
    public String Morada;
}
```

A implementação de uma nova aplicação na empresa necessita de fazer uso do mesmo repositório de informação referente aos funcionários, de onde pretende reutilizar os campos Número, Nome Username e Password.
Para além desta informação, a nova aplicação pretende associar a cada funcionário um conjunto de contactos diversos (email, telefone, etc.).
Implemente uma solução para o problema apresentado.

12.Composite

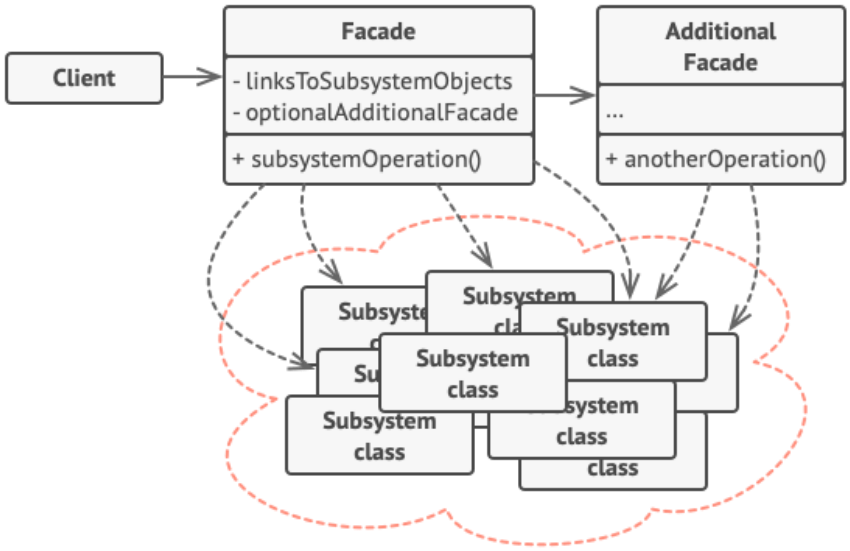


12.1. Um diagrama é uma estrutura que constituída por objetos como Círculo, Linhas, Triângulo, etc.. Quando desenhamos o diagrama com uma cor (digamos, vermelho), essa mesma cor também é aplicada a todos os objetos usado no diagrama.

Implemente uma solução para o problema apresentado.

.

13.Facade



13.1. Comando Simplificado

Implemente um comando universal que permita simplificar as operações sobre um sistema multimédia composto pelos seguintes componentes:

- Rádio (On, Off, Set AM, Set FM, Set Frequency);
- Vídeo DVD (On, Off, Play; Pause, Load, Eject, Stop);
- Amplificador (On, Off, Set Input, Volume Up, Volume Down);
- TV (On, Off, Set Input, Volume Up, Volume Down, Set Canal);

No comando defina as operações:

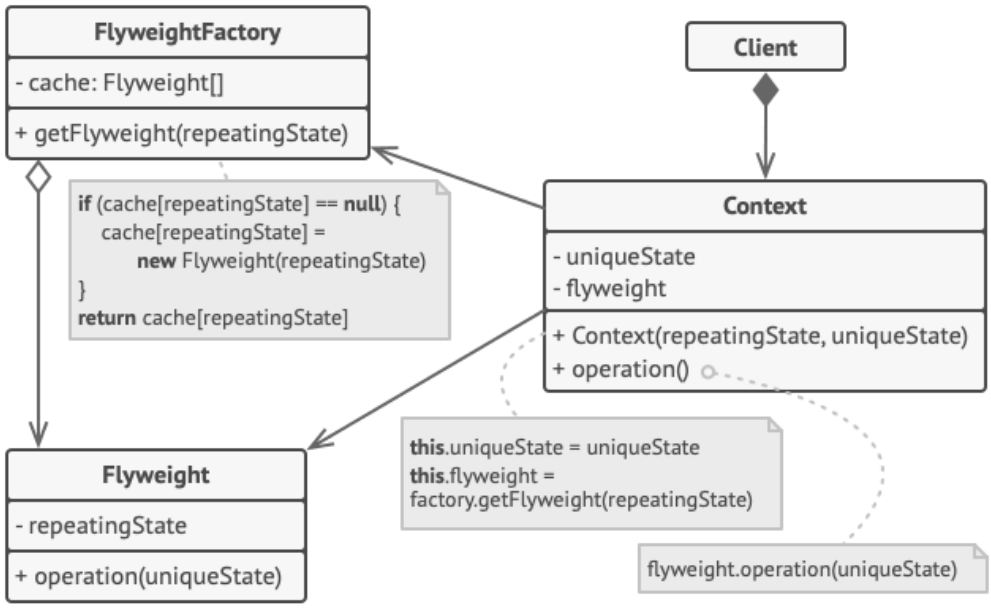
- Ver filme DVD;
- Ver canal TV;
- Ouvir Rádio;
- …

13.2. Método SendEmailMessage

Implemente uma classe que sirva de interface para o envio de uma mensagem de email, em ambos os formatos texto plain e html, para um endereço concreto.

Dica: faça uso das classes **Properties**, **Message**, **MimeMessage**, **InternetAddress**, **MimeBodyPart** e **Transport** na sua implementação.

14.Flyweight



14.1. Desenho de objetos repetidos

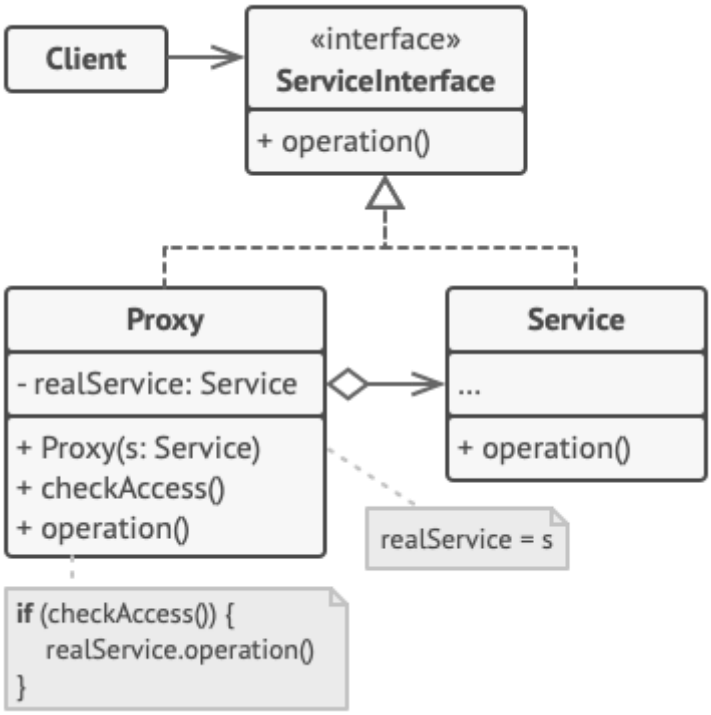
Implemente um gerador de círculos coloridos, desenháveis com dimensões e em posicionamentos diferentes. Considere os atributos intrínsecos a cor do objeto que podem ter valores Vermelho, Verde, Azul, Branco e Preto.

Simule a geração aleatória e apresentação no ecrã das características de 10 círculos.

14.2. Animação de personagens de jogo

Num jogo são usadas imagens para a renderização dos atores de um nível (jogador e inimigos). Considerando que cada ator tem 4 imagens que caraterizam a animação do seu movimento em [X,Y] dentro da cena (por exemplo: jogador1.png, jogador2.png, jogador3.png, jogador4.png, inimigo1.png, inimigo2.png, inimigo3.png, etc.) e que no mesmo nível podem estar 1 jogador e 3 inimigos, simule a movimentação das personagens dentro da cena. Para isso use um posicionamento inicial [X,Y] aleatório para cada um deles e o seu movimento nos dois eixos seja igualmente aleatório com amplitude entre -2 e 2 ().

15.Proxy



15.1. Lista de contactos

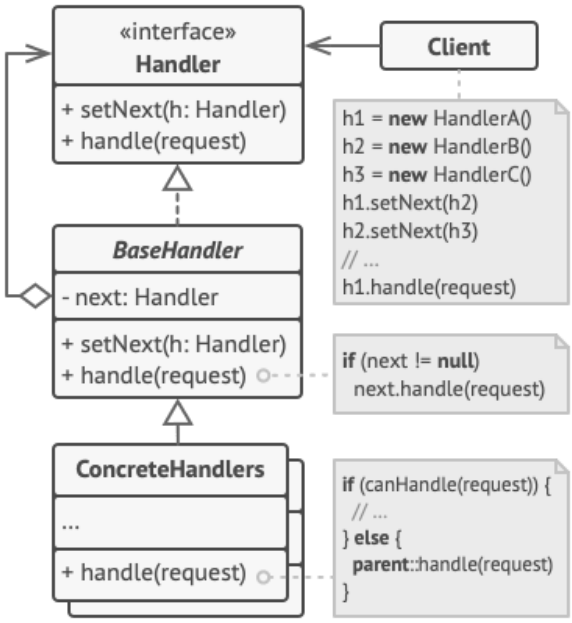
Um sistema de gestão de lista de contactos disponibiliza a consulta dos contactos armazenados através do recurso a um processo complexo em termos operacionais e tempo de execução.

Para agilizar o processo de consulta dos contactos, pretende-se implementar um sistema de cache que permita acelerar a obtenção de valores com maior solicitação.

Para a implementação do sistema, assuma uma capacidade de cache para os últimos 5 valores transacionados.

Padrões comportamentais

16.Chain of Responsability



16.1. Dispensador de trocos

Implemente um dispensador de trocos que permita, a partir de uma quantia numerária, indicar, de forma otimizada, a quantidade de notas e moedas que compõe essa quantia.

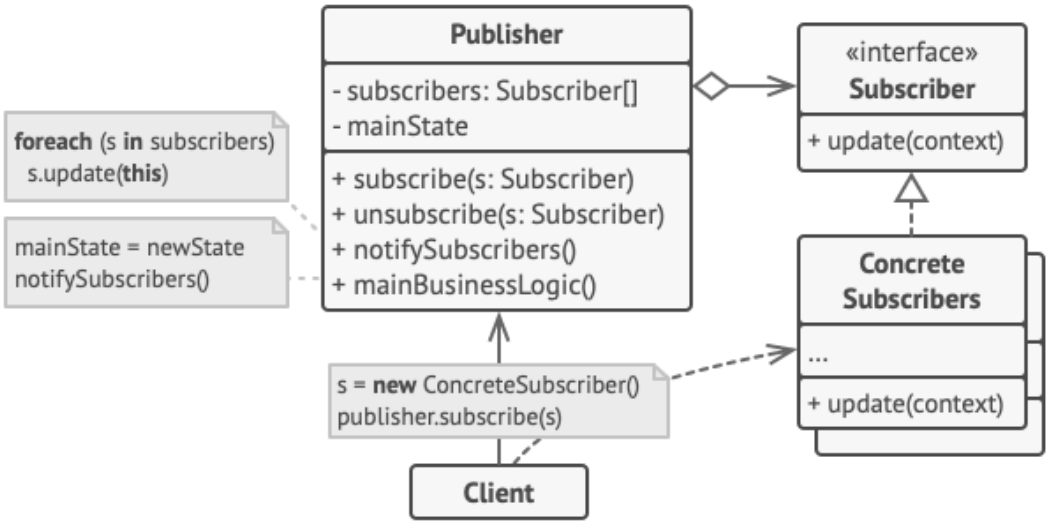
Considere as seguintes quantias de moedas e notas (0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500).

16.2. Sistema de Logging

Implemente um sistema de logging parametrizado com um nível de registo (INFO=0, DEBUG=1 e ERROR=2). O sistema suporta vários tipos de loggers, associados a dispositivos diferentes (Console, Error, File, etc.), cada um deles caracterizado por um dos níveis previstos.

Cada um dos loggers verifica o nível da mensagem recebida e, caso seja o seu, efetua o log. Posteriormente passa a mensagem para seu próximo logger caso este exista.

17.Observer



17.1. Alerta de Stock

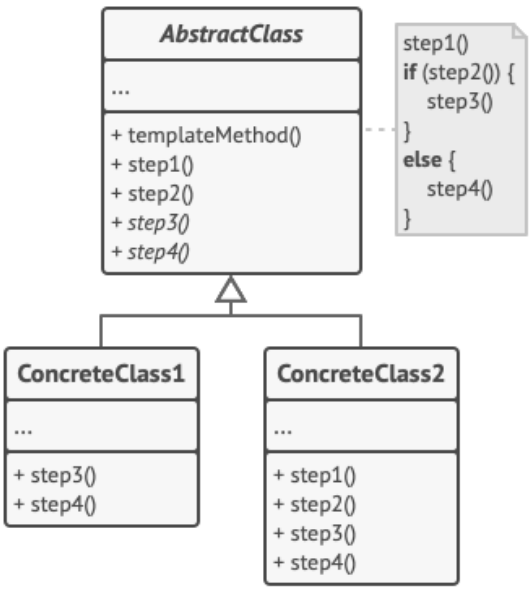
Implemente um sistema de uma loja de comércio que contenha um mecanismo de alertas de reposição de stock para produtos esgotados. A loja, quando não dispõe em armazém um qualquer produto existente em catálogo, permite aos clientes subscrever uma notificação de reposição de stock desse produto.

17.2. Central de Mensagens temáticas

Desenvolva um sistema que implemente uma central de partilha de mensagens temáticas. Os clientes da central podem enviar mensagens para os canais definidos na central, cada um deles associado a uma temática concreta e também pode ler todas as mensagens enviadas para a central.

Os clientes podem optar por serem notificados se a central receber mensagens em canais específicos. Para isso necessitam subscrever o canal em questão, podendo subscrever vários.

18.Template Method



18.1. Football Supporter

Implemente um sistema que modele o comportamento de um adepto de futebol que assiste aos jogos no estádio.

As ações obrigatórias em dia de jogo são:

- void ComprarBilhete ()
- void IrAoEstadio()
- void VerJogo()
- void VoltarParaCasa()

Poderão haver ações complementares antes de IrAoEstadio e depois de VoltarParaCasa.

18.2. Jogos de Tabuleiro

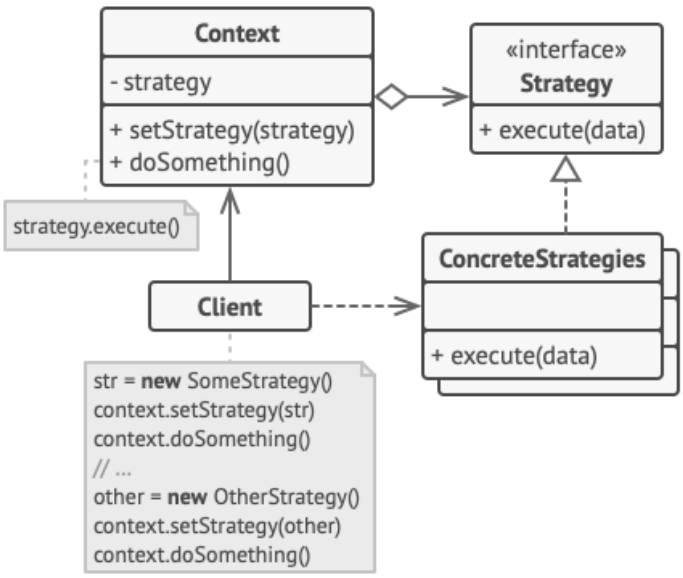
Uma empresa de jogos de tabuleiro decidiu uma app móvel. A empresa tem cinco jogos de tabuleiro: Xadrez, Damas, Gamão, Mastermind e Ludo. A empresa quer desenvolver todos estes jogos numa única aplicação móvel.

A lista de métodos obrigatórios para cada jogo são:

- void iniciarJogo()
- void escolherPeça()
- void escolherNumerodeJogadores(List<String> listaJogadores)
- String jogar(String jogador, String jogada)

- String mostraPontuacao(List<String> listaJogadores, double tempoAtualEmSegundos, pontuacaoAtual)
- void fimJogo()
- void voltarAJogar()

19.Strategy



19.1. Ordenação dos clientes

Uma empresa financeira necessita de uma solução para os gestores de conta analisarem a carteira de clientes mediante critérios diferentes de ordenação.

Os critérios podem variar entre maior quantia para investimento e maior quantia investida.

A ordenação deve permitir identificar os clientes com melhores condições para investimento.

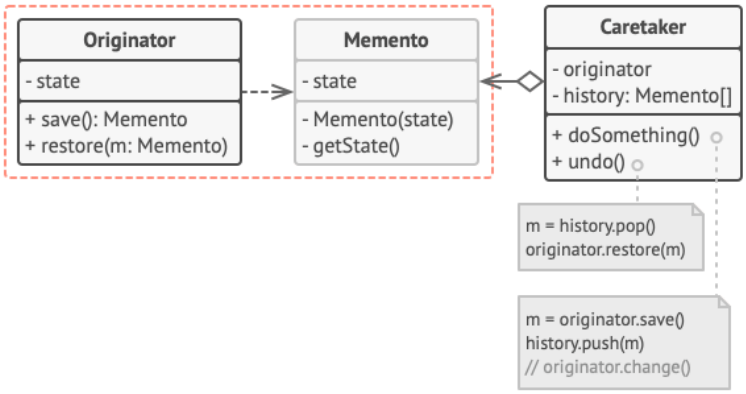
19.2. Livraria

A livraria Lélé vende diversos artigos do tipo Livros, Revistas e Jogos. Há livros e revistas em suporte papel e em suporte digital. A livraria possui Jogos de Tabuleiro e Jogos Eletrónicos, sendo o último o jogo físico ou digital.

No pagamento à vista, tem a seguinte política de descontos:

- Livros, Revistas Físicos e Jogos de Tabuleiro: 30% de desconto;
- Livros e Revistas Digitais: 15% de desconto;
- Não há desconto para Jogos de Vídeo Game;
    - Para além disto, a livraria pode ter promoções especiais durante o ano, de modo que os descontos dos produtos podem mudar.

20.Memento



20.1. Histórico de operações

Uma calculadora de números inteiros implementa o cálculo das operações de adição ('+'), subtração ('-'), multiplicação ('*') e divisão ('/') de dois números de forma acumulada.

No seu funcionamento, permite a leitura de um número, seguido da leitura da operação e da leitura do segundo número. Após esta terceira leitura, o valor da operação é calculado e assumido como sendo o primeiro valor da operação seguinte.

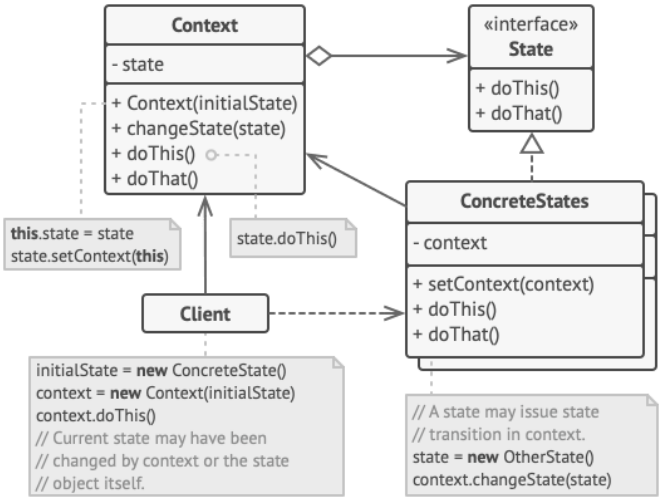
A cada operação calculada é guardado o estado da calculadora de forma a permitir efetuar o restauro (undo). Esta ação de restauro é efetuada quando a operação inserida for o Undo ('U' ou 'u').

A operação Sair ('S' ou 's') abandona a calculadora.

20.2. Histórico de documentos

Um software de edição de documentos necessita de um sistema de versionamento dos documentos editados. O sistema guarda uma nova versão a cada ação de "Guardar" executada no editor. Por uma questão de eficiência o sistema apenas guarda as 10 últimas versões do documento.

21.State



21.1. Registo de assiduidade

Considere um *software* para registo de presenças em sala de aula baseado na leitura de um cartão RFID. A entrada e saída da sala é efetuada por torniquetes que são desbloqueados através da leitura do cartão. A sala tem uma capacidade que não pode ser excedida.

O sistema reconhece dois tipos de utilizadores, o docente e o aluno, e apenas permite entrar alunos na sala após um docente ter entrado. Se o docente sair da sala o sistema permite apenas a saída de alunos.

Considere que apenas pode estar um docente dentro da sala.