UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO
SCC0204/SCC0504 – Programação Orientada a Objetos
Prof. José Fernando Rodrigues Júnior

Lista 10 - Padrões de projeto

1. (Bridge) Implemente uma classe que representa dois vetores matemáticos em R^3 e, usando o padrão Bridge, defina 3 operações para objetos desta classe: adição, produto escalar, e produto vetorial. Exemplifique no método main.

2. (Decorator) Implemente uma classe Java chamada VogalCounterInputStream que herde de FilterInputStream e funcione como um decorador para contagem de vogais em fluxos de entrada de dados. O objetivo é que, ao ler os dados por meio de read(), a classe contabilize quantas vogais (letras a, e, i, o, u, maiúsculas ou minúsculas) foram lidas desde a criação do stream.

- A classe VogalCounterInputStream deve estender FilterInputStream, recebendo um InputStream no construtor.
- Sobrescreva o método int read() de modo que:
  - O byte lido seja passado normalmente para o código cliente.
  - Caso o caractere correspondente ao byte lido seja uma vogal (aAeEiIoOuU), incremente um contador interno.
- Implemente um método público int getTotalVogaisLidas() que retorne o total acumulado de vogais lidas até o momento.

- Crie uma classe principal (Main) e mostre dois exemplos de uso:
- Exemplo 1 – Leitura a partir de um arquivo .zip:
  - Use ZipInputStream para ler um arquivo .zip que contenha um único arquivo de texto.
  - Decore o ZipInputStream com seu VogalCounterInputStream.

```
FileInputStream theFile = new FileInputStream("ZIPFile.zip");
ZipInputStream zip = new ZipInputStream(theFile);
zip.getNextEntry(); /*set o 1o. arquivo dentro do zip como fonte de dados*/
VogalCounterInputStream vc = new VogalCounterInputStream(zip);
```

  - Leia e imprima o conteúdo do arquivo.

- Exemplo 2 – Leitura a partir do teclado (System.in):
  - Use System.in como fonte de dados e decore-o com VogalCounterInputStream.
  - Leia e imprima os caracteres digitados até o final da entrada (digite ‘x’ para interromper o input).

3. (Decorator) Escreva uma classe derivada de FilterInputStream que seja capaz de decorar o funcionamento de um BufferredReader de maneira que quando o método readLine for evocado, a linha retornada mostre um contador de quantos caracteres já foram lidos desde que o stream foi criado, junto com a linha que foi lida. Exemplifique.

4. (Factory + Facade)
Implemente uma aplicação de cálculo aritmético que utilize o padrão de projeto Facade para centralizar a aplicação das operações: adição, subtração, multiplicação e divisão, permitindo que o usuário informe expressões no formato "número operador número" (por exemplo: "3 + 4").

- Crie uma interface chamada Operacao, com um método double calcular(double a, double b);
- Implemente quatro classes concretas que representam cada operação: Soma, Subtracao, Multiplicacao e Divisao, todas implementando a interface Operacao;
- Crie uma classe OperacaoFactory para instanciar objetos da família de classes Operacao;
- Implemente uma classe chamada CalculadoraFacade que oferece dois métodos:

- double calcular(String operacao, double a, double b): recebe o símbolo da operação ("+", "-", "*", "/"), instancia a respectiva operação, e executa o cálculo correspondente;
- double calcularExpressao(String expressao): recebe uma expressão no formato "a operador b", separa os elementos e retorna o resultado do cálculo; para separar os elementos use

```
String[] elementos = expressao.trim().split(" ");
```

- Implemente uma classe principal chamada Main que instancia a CalculadoraFacade e executa expressões aritméticas.

⇨ A expressão deve conter exatamente dois operandos e um operador separados por espaços. Se a operação for inválida ou a divisão for por zero, a aplicação deve imprimir uma mensagem de erro e retornar -1 (ou lançar uma exceção e interromper).

5. (Singleton) Implemente um sistema de registro de eventos (classe Logger) que utilize o padrão de projeto Singleton para garantir uma única instância responsável por registrar mensagens. A classe deve conter um método registrar(String mensagem) que imprime a mensagem no console com um timestamp.

Em seguida, crie três classes chamadas ModuloUsuario, ModuloPagamento e ModuloRelatorio, cada uma com um método que simula uma operação (como login, pagamento ou geração de relatório) e registra uma mensagem usando o Logger. Na classe principal (Main), instancie os três módulos, chame seus métodos de operação e verifique se todas as mensagens foram registradas usando a mesma instância de Logger, imprimindo ao final uma verificação que confirma se a identidade das instâncias for de fato a mesma.

6. (UML) Escreva o diagrama UML dos projetos vistos em aula: Bridge, ChainOfResponsibilityTextParser, Facade, FactoryMethod, MediatorBuyer, e Observer.

7. (Observer) Verifique a implementação Java e implemente seu próprio padrão observador (sem se preocupar com questões de sincronização).

8. (Bridge e Factory)
a) Use o padrão Bridge para implementar uma classe MeusVetores que tem como propriedades dois conjuntos de inteiros positivos representados como arrays, e que recebe funcionalidades novas para serem executadas sobre estes dois conjuntos. Escreva 3 funcionalidades: união, intersecção, e diferença entre conjuntos. Após a execução, as funcionalidades deverão exibir os dados de cada um dos conjuntos e também o resultado do processamento. Exemplifique no método main.

b) Altere seu código para que as funcionalidades do item a) sejam criadas por intermédio de um Factory.

9. (Decorator) Implemente uma classe Divida e use o padrão Decorator para efetuar os seguintes cálculos sobre a dívida representada por objetos desta classe: juros, juros com desconto, e juros com desconto com acréscimo de taxa. Note que a assinatura do Decorator irá requerer, além do objeto sendo decorado, um parâmetro cujo significado muda de decorador para decorador – defina um único parâmetro na classe mãe dos decoradores.

10. (Proxy e Decorator) Considere um Sistema (classe) que é vulnerável toda vez que um de seus dois métodos (pense em métodos fictícios) recebe como parâmetro uma string “789”. Escreva a classe Sistema.

a) Use o padrão proxy para escrever uma classe que filtra chamadas de métodos cujos parâmetros coloquem em risco o sistema – veja o exemplo de ProxySeguranca no Tidia.

b) Use o padrão decorator para escrever uma classe FilteredSistema que, caso a string “789” seja detectada, ela seja convertida para “987”.

11. (Chain of Responsibility) Use o padrão chain of responsibility para processar números, sendo que as classes da cadeia deverão dividir o processamento da seguinte maneira: classe 1, números múltiplos de 2; classe 2, números múltiplos de 3; classe 3, demais números. Implemente o cálculo do quadrado de um número dado, e o cálculo da raiz quadrada de um número dado. Exemplifique no método main.

12. (Observer) Use o padrão Observer para escrever uma classe tal que toda vez que ela recebe uma nova string, observadores distintos realizam o seguinte processamento: transformar a string inteira em maiúscula, concatenar o ano atual ao fim da string, e concatenar o tamanho da string ao fim da própria string.