SÃOJUDAS UNIVERSIDADE	FTCE Faculdade de Tecnologia e Ciências Exatas

USJT – 2019 - ArqDeSisMulti – Arquitetura e Desenvolvimento de Sistemas Multicamadas

Professor: Rodrigo

Aula: 4
Assunto: Patterns Estruturais e Comportamentais

Conteúdo

Veja a apresentação Aula04-Teoria-USJT-2019-ArqDeSisMulti-Patterns Estrut Comp-adap-faca-stra-templ.pdf

Os exercícios abaixo se referem apenas aos patterns cujo código exemplo foi discutido em aula:

4. Adapter [JOSHI 16, pags. 5 a 12]
5. Facade [JOSHI 16, pags. 13 a 16]
6. Strategy [JOSHI 16, pags. 133 a 136]
7. Template Method [JOSHI 16, pags. 118 a 124]

Exercício do Laboratório

4. Adapter [SOUZA 05b]

4.1. Exercício:

A classe java.util.Map da API de coleções de Java permite que sejam armazenados pares de objetos (chave e valor) em uma de suas implementações (as mais conhecidas são HashMap e TreeMap). No entanto, estas classes não possuem um construtor que receba como parâmetro uma matriz de duas linhas e que monte o mapa usando a primeira linha como chave e a segunda como coluna. Crie um adaptador (dica: use Adapter de classe) que tenha este construtor.

4.2. Exercício:

Abaixo estão os códigos fonte de um cliente, uma interface para um somador que ele espera utilizar e uma classe concreta que implementa uma soma, mas não da maneira esperada pelo cliente. Como você pode ver abaixo, o cliente espera usar uma classe que soma inteiros em um vetor, mas a classe pronta soma inteiros em uma lista. Crie um adaptador (dica: use Adapter de objeto) para resolver esta situação.

```
public class Cliente{
  private SomadorEsperado somador;
```

```
  private Cliente(SomadorEsperado somador) {
    this.somador =somador;
  }

  public void executar() {

    int[] vetor = new int[] {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int soma =somador.somaVetor(vetor);
    System.out.println("Resultado: " +soma); }
  }

public interface SomadorEsperado{
  int somaVetor(int[] vetor);
}


import java.util.List;

public class SomadorExistente{

  public int somaLista(List<Integer> lista){
    int resultado = 0;
    for(int i: lista)
      resultado +=i;

  return resultado;
  }
}
```

5. Facade

5.1. Exercício:

Frequentemente precisamos ocultar um conjunto de classes com uma interface complexa por trás de uma classe mais simples. Isso acontece, por exemplo, ao reutilizar código escrito por outra equipe de programadores.

Escreva uma classe Java chamada IOFacade que simplifique o uso dos recursos de I/O do Java com métodos para abrir arquivos texto, arquivos binários e objetos.

6. Strategy

6.1. Exercício:

Escreva um programa que exiba uma mensagem diferente para cada dia da semana usando o padrão Strategy.

6.2. Exercício:

Abaixo estão implementadas quatro formas bastante conhecidas de ordenação: bubble sort, insertion sort, selection sort e quick sort. Coloque-as no padrão Strategy e escreva um cliente que alterna de estratégia de ordenação livremente. Se estiver curioso, cronometre a

execução de cada método para verificar qual é o mais eficiente (deve ser usada uma quantidade grande de números no vetor para perceber a diferença).

```
public class Ordem {

  /*
   * Bubble sort
   */
  public static int[] bolha(int v[]) {

    for (int i = v.length - 1; i > 0; i--) {
      for (int j = 0; j <= i - 1; j++) {
        if (v[j] > v[j + 1]) {
          int aux = v[j + 1];
          v[j + 1] = v[j];
          v[j] = aux;
        }
      }
    }
    return v;
  }

  /*
   * Insertion sort
   */
  public static int[] insercao(int v[]) {
    int i, j, x;

    for (j = 1; j < v.length; ++j) {
      x = v[j];
      for (i = j - 1; i >= 0 && v[i] > x; --i) {
        v[i + 1] = v[i];
      }
      v[i + 1] = x;
    }
    return v;
  }

  /*
   * Selection sort
   */
  public static int[] selecao(int v[]) {
    int i, j, min, x;

    for (i = 0; i < v.length - 1; ++i) {
      min = i;
      for (j = i + 1; j < v.length; ++j) {
        if (v[j] < v[min])
          min = j;
      }
      x = v[i];
      v[i] = v[min];
      v[min] = x;
    }
    return v;
  }

  /*
   * Quicksort - particionamento
```

```
   */
  private int separa(int v[], int p, int r) {
    int c = v[p + ((int) (Math.random() *
      (1 + r - p) * 1000) % (1 + r - p))];
    int i = p + 1, j = r, t;
    while (true) {
      while (i <= r && v[i] <= c)
        ++i;
      while (c < v[j])
        --j;
      if (i >= j)
        break;
      t = v[i];
      v[i] = v[j];
      v[j] = t;
      ++i;
      --j;
    }
    v[p] = v[j];
    v[j] = c;
    return j;
  }

  /*
   * Quicksort - principal (chama o metodo separa)
   */
  public void quicksort(int v[], int p, int r) {
    int j;
    if (p < r) {
      j = separa(v, p, r);

      quicksort(v, p, j - 1);
      quicksort(v, j + 1, r);
    }
  }
}
```

7. Template Method

7.1. Exercício:

Exercite o padrão Template Method criando uma classe abstrata que lẽ uma String do console, transforma-a e imprime-a transformada. A transformação é delegada às subclasses. Implemente quatro subclasses, uma que transforme a string toda para maiúsculo, outra que transforme em tudo minúsculo, uma que duplique a string e a ultima que inverta a string.

7.2. Exercício:

Os Comparators de Java podem ser considerados uma variação do Template Method, apesar de não serem feitos via herança. Monte um vetor de doubles e escreva um comparator que compare os números de ponto-flutuante pelo valor decimal (desconsidere o valor antes da virgula). Em seguida, use Arrays.sort() para ordenar o vetor e Arrays.toString() para imprimi-lo.

Tarefa para Entrega

1. Termine em casa o exercício do lab.
2. Entregue a tarefa enviando via GitHub.

Bibliografia

[JOSHI 16] JOSHI, R.; Java Design Patterns: reusable solutions to common problems. Java Code Geeks. 2016.

[GAMMA 00] GAMMA et al.; Padrões de Projeto: soluções reutilizáveis de software orientado a objetos. 1a Edição. Bookman. 2000.

[FREEMAN 04] FREEMAN et al.; Head First Design Patterns. 1a Edição. O'Reilly. 2004.

[SOUZA 05a] SOUZA, V.; Padrões de Projeto em Java: exercícios - padrões de criação; disponível online em <http://www.inf.ufes.br/~vitorsouza/wp-content/uploads/teaching-br-padroesdeprojeto-exercicios01.pdf>. Acessado em 06/11/2016

[SOUZA 05b] SOUZA, V.; Padrões de Projeto em Java: exercícios - padrões de estrutura; disponível online em <http://www.inf.ufes.br/~vitorsouza/wp-content/uploads/teaching-br-padroesdeprojeto-exercicios02.pdf>. Acessado em 06/11/2016

[SOUZA 05c] SOUZA, V.; Padrões de Projeto em Java: exercícios - padrões de comportamento; disponível online em <http://www.inf.ufes.br/~vitorsouza/wp-content/uploads/teaching-br-padroesdeprojeto-exercicios03.pdf>. Acessado em 06/11/2016