INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO MARANHÃO
CAMPUS SÃO LUÍS – MONTE CASTELO
DIRETORIA DE ENSINO SUPERIOR
DEPARTAMENTO DE COMPUTAÇÃO

Curso de Sistemas de Informação

Disciplina: Padrões de Software
Profº João Carlos Pinheiro
Data de Entrega: 13/05/2022

Etapa 01 – Padrões Template Method & Builder

1. Quando devemos aplicar o padrão Template Method?

2. Será que podemos misturar os padrões de projeto? Será que o Template Method poderia ser usado em conjunto com o Chain of Responsibility? Como?

3. Relatórios são muito parecidos: todos eles contêm cabeçalho, corpo, e rodapé. Existem dois tipos de relatórios: simples e complexos. As diferenças são sutis: relatórios simples possuem cabeçalhos e rodapés de uma linha, apenas com o nome do banco e telefone, respectivamente; relatórios complexos possuem cabeçalhos que informam o nome do banco, endereço, telefone, e rodapés que informam e-mail, e a data atual. Além disso, dada uma lista de contas, um relatório simples apenas imprime titular e saldo da conta. O relatório complexo exibe titular, agência, número da conta, e saldo. Use Template Method, e implemente o mecanismo de relatórios. Use dados falsos para os dados do banco.

4. Por que ao invés do Builder, não usar um conjunto de construtores nesse objeto, que fariam a mesma coisa que o Builder? Discuta vantagens e desvantagens de se usar diversos construtores ao invés do Builder?

5. Quando podemos aplicar o padrão Builder? E quando ele mais atrapalhará do que ajudará?

6. Compare os padrões Builder e StepBuilder. Descreva um cenário em que seria mais vantajoso usar o padrão StepBuilder

7. Escreva classes para satisfazer os seguintes papéis do padrão Builder:

- **Client**: recebe como parâmetros o nome, endereço, telefone e e-mail de uma pessoa, solicita ao director que construa informações de contato, recupera a informação do builder e imprime;

- **Director**: recebe como parâmetro o builder a ser utilizado e os dados de contato. Manda o builder construir o contato;

- **Builder**: constrói o contato. Existem três tipos de contato e um builder para cada tipo:

    - ContatoInternet: armazena nome e e-mail;
    - ContatoTelefone: armazena nome e telefone;
    - ContatoCompleto: armazena nome, endereço, telefone e e-mail.

A classe que representa o papel client deve ter o método main() que irá criar um director e um builder de cada tipo. Em seguida, deve pedir ao director que crie um contato de cada tipo e imprimi-los (use o toString() da classe que representa a informação de contato)

8. Considere a classe Livro apresentada a seguir.

```
public class Livro {
  private String nomeNacional;
  private int ano;
  private List<String> autores;
  private int edicao;
  private String cidade;
  private String editora;
  private String nomeOriginal;
  private List<String> tradutores;
  private int paginas;
  private long isbn;

  public Livro(String nomeNacional, int ano, List<String> autores,
               int edicao, String cidade, String editora, String nomeOriginal,
               List<String> tradutores, int paginas, long isbn) {
    this.nomeNacional = nomeNacional;
    this.ano = ano;
    this.autores = autores;
    this.edicao = edicao;
    this.cidade = cidade;
    this.editora = editora;
    this.nomeOriginal = nomeOriginal;
    this.tradutores = tradutores;
    this.paginas = paginas;
    this.isbn = isbn;
  }

  // getters e setters omitidos
}
```

Observe o construtor da classe Livro. Veja como fica complicado escrever tantos parâmetros. Uma solução seria a utilização do padrão de projeto Builder, que seria responsável pela construção do objeto, como por exemplo:

```
Livro livro = new Livro.LivroBuilder("Java, como programar")
        .publicadoEm(2003)
        .dosAutores("H. M. Deitel", "P. J. Deitel")
        .edicao(4)
        .cidade("Porto Alegre")
        .editora("Bookman")
        .nomeOriginal("Java How to Program")
        .tradutores("Carlos Arthur Lang Lisbôa")
        .paginas(1386)
        .isbn(9788536301235L)
        .build();
```

Desenvolva a solução requisitada, utilizando o padrão de projeto *Builder* e *fluent interface*.