

# Trabalho Final de POO - 2020.2

UFF - Universidade Federal Fluminense

IC - Instituto de Computação

Disciplina: Programação Orientada a Objetos - 2020.2

Professora: Vania De Oliveira Neves

Integrantes: Danilo Siqueira e Winne Domingues

Repositório: <https://github.com/igorcordeiro08/TrabalhoFinal> \*(verificar nota ao final)

## Relatório do Trabalho Final

### Descrição

Trata-se de um sistema de gerenciamento de orçamentos de uma construtora, a fim de receber pedidos de Clientes, entrar com as medidas do projeto e, considerando, dados de insumos necessários para 1 metro quadrado dos itens de construção padrão (Laje, Parede e Coluna), também os insumos que cada fornecedor dispõe, é construída a edificação desejada em composições de classes através dos módulos gerenciadores, a fim de realizar o dimensionamento dos materiais, comparar os fornecedores, tomando o que tem menor custo e seja capaz de fornecer os materiais e, enfim, gerando o projeto e o orçamento.

### Atribuições:

- Criação e Levantamento de Requisitos:** Danilo, Igor e Winne.
- Modelagem e Diagramação do Projeto:** Danilo, Igor e Winne.
- Classes de Domínio:** Danilo e Winne.
- helper (Enums):** Danilo.
- Helpers (métodos auxiliares):** Danilo e Winne.
- Classes de service:**

- CalculadoraOrcamento, GerenciadorProjeto, ValidadorProjeto, GerenciadorCliente, GerenciadorCatalogo, GerenciadorContrato:** Danilo.
- GerenciadorAndar, GerenciadorEdificacao:** Winne.
- mocks:** Danilo.
- Apresentação:** Winne.
- Revisão:** Danilo e Winne

Apesar das atribuições iniciais, houve jornadas de programação e revisões conjuntas.

## Itens de Java Utilizados

- o **Coleções utilizadas:** ArrayList, HashMap, Map
- o **Bibliotecas externas utilizadas:** Para a leitura dos JSONs utilizados para persistência de dados.

```
com.google.gson.Gson
```

- o **Condicionais:** for, switch ... case
- o **Interfaces:** PessoaFisica, PessoaJuridica, Interface Comparable
- o **Classes Abstratas:** Contrato, Pessoa
- o **Funções anônimas, Lambda expressions**
- o **Modificadores de de Acesso:** private, protected, public
- o **Interface Comparable e compareTo()** em Fornecedor
- o **BigDecimal:** para ter mais precisão
- o **Lançamento de exceções com throws**
- o **Tratamento de exceções**
- o **java.io.IOException**
- o **java.io.Reader**
- o **java.io.Writer**
- o **java.nio.file.Files**
- o **java.nio.file.Paths**
- o **Collectors**

- **Sobrecarga de métodos**
- **Sobrescrita de métodos**
- **Polimorfismo**
  
- JSON para armazenamento de instâncias de Clientes, Projetos, Funcionarios, Fornecedores e mocks de dados.
  
- **Enum:**
  - Enum\_mock
  - NumerosOrdinais
  - SituacaoProjeto, para representar os possíveis estados do Projeto
  
- Parâmetros no JSONS:
  - funcionariosTempo: representando quanto tempo leva 1 funcionario para realizar cada tarefa de construção.
  - tempoFuncionarios: representando quantos funcionarios são necessários para cada tarefa de construcao por unidade de tempo (hora).
  - metrosQuadrados: representando o consumo de insumos para cada a execução de cada item da construção
  
- **Helpers (métodos auxiliares criados - nomes bem descritivos):**

```
```java
String getOrdinal(int i, boolean isMasculino)
int validaInteiroPositivo()
int minimoDiasPorHora(int horas)
public static void clear() //limpa a tela de acordo com o S.O.
Object[] getSliceOfArray(Object[] arr, int start, int end)
```
```

## Arquitetura da Interface com o usuário (CLI):

Menu Principal:

1 - Cadastrar Projeto

Já existe Cliente Cadastrado?

S:

busca por nome ou cpf

Cadastra o Projeto

//Criação da edificação

Quantos andares da primeira edificação

o?

//Criação do Andar

Quantos m<sup>2</sup> de laje (chão e teto)?

Quantos m<sup>2</sup> de paredes?

Quantas colunas de Sustentação?

Se foi informado mais de um andar:

O próximo andar será igual à este?

S:

cria um novo igual e adiciona

a ao projeto?

N:

repete os passos de criação

do andar

Alguma Outra Edificação:

S:

Repete passos de criação da edificação

ão

N:

Salva o projeto e volta ao Menu Principal

cipal

N:

Cadastra o Cliente

Cadastra o Projeto, conforme passos acima

Digite o nome do Cliente:

Digite o CPF do Cliente:

### 3 - Selecionar Projeto

Escolha a situação do(s) projeto(s) para exibí-los:

#### 1 - Todos os Projetos

Selecione o projeto para avançar situação:

Lista Projeto

Gera orçamento

2 - Projeto aguardando liberação de insumos ou funcionários.

3 - Projeto pendente de aprovação do orçamento para iniciar.

4 - Orçamento esta sendo revisado.

5 - Projeto iniciado.

6 - Projeto finalizado.

//Fluxos internos de 2 a 6 com resultados semelhantes ao do item 1, variando apenas o filtro por estado do projeto

ENTER - Encerra

\*Obs: O repositório pertence ao Igor Cordeiro , que era integrante do grupo, contribuiu com a modelagem do Sistema, mas resolveu não continuar na disciplina neste momento.

## Relatório da Avaliação Individual

1. Altere seu programa para que **todos** os métodos que fazem uso de manipulação de arquivos ou banco de dados **lançam novas** exceções relacionadas ao tipo de


erro que pode ocorrer naquele método (por exemplo, em métodos que buscam um determinado objeto, lançar a exceção `ObjetoNaoEncontradoException`) ao invés de tratá-la nesse método. Essa exceção deverá ser tratada nas classes que interagem com o usuário em que deverão ser exibidas ao usuário mensagens correspondentes às exceções lançadas (pode ser via `System.out.println`)

## Solução:

Repositório utilizado:

[GitHub - wdomingues/TrabalhoFinalPOO-UFF](https://github.com/wdomingues/TrabalhoFinalPOO-UFF)

<https://github.com/wdomingues/TrabalhoFinalPOO-UFF>

 wdomingues/TrabalhoFinalPOO-UFF • github.com

Exceções personalizadas criadas em `com.company.exceptions`

```
-DocumentoInvalidoException - Utilizada no GerenciadorCliente  
para lançar exceção se a validação do documento falhar.  
-ObjetoNaoEncontradoException, usada ao tentar ao validar o ma  
pa gerado JSON através da biblioteca GSON, desde o Gerenciador  
de Projeto e o Gerenciador de Orçamento
```

Métodos que passaram a propagar `IOException` (por classe)

(Foram propagados em cascata até às suas chamadas originais na Main)

```
CalculadoraOrcamento
```

```
Orcamento calcula(Projeto projeto)  
Map getFuncionarioProduzEmHoras()  
Map getFuncionariosProduzirEmUmaHora()  
private static Map geraFuncionarioTempo() throws IOException{  
    MapgerarTempoFuncionarios()  
}
```

```
Map getFileMapStringDouble(String path)
Projeto[] salvaOrcamentoPendenteAprovacao(Orcamento orcamento)
Projeto[] salvaOrcamentoOrcamentoRevisao(Orcamento orcamento)
Projeto[] salvaOrcamentoOrcamentoRevisao(Orcamento orcamento)
Projeto[] salvaOrcamentoAprovado(Orcamento orcamento)
Orcamento[] salvaOrcamento(Orcamento orcamento, SituacaoProjeto situacao)
Orcamento usaOrcamento(String dadoBusca)
```

## GerenciadorCatalogo

```
Catalogo gerarCatalogo()
Funcionario[] identificarFuncionariosDisponiveis()
void vincularInsumoProjeto(Insumo insumo, Projeto projeto)
void vincularFuncionarioProjeto(Funcionario funcionario, Projeto projeto)
Catalogo atualizarCatalogo(Catalogo catalogo, Fornecedor fornecedorAlterado, Funcionario funcionarioAlterado)
```

## GerenciadorCliente

```
Cliente getCliente()
Cliente cadastrar()
Cliente[] salvaCliente(Cliente cliente)
Cliente usaCliente(String dadoBusca)
GerenciadorProjeto
Projeto[] recuperaProjetos()
Projeto[] salvaProjetoEmEspera(Projeto projeto)
Projeto[] salvaProjetoPendenteAprovacao(Projeto projeto)
```

```
Projeto[] salvaProjetoOrcamentoRevisao(Projeto projeto)
Projeto[] salvaProjetoAprovado(Projeto projeto)
Projeto[] finalizaProjeto(Projeto projeto)
Projeto[] salvaProjeto(Projeto projeto, SituacaoProjeto situacao)
```

Projeto

```
void setFuncionarios(ArrayList funcionarios)
```

Orcamento

```
void setFuncionarios(ArrayList funcionarios)
```

Main

```
Orcamento validaProjetoGeraOrcamento (Catalogo catalogo, GerenciadorProjeto gerenciadorProjeto, Projeto projeto)
```

2. O cliente da sua aplicação solicitou que, além de arquivos, também seja possível utilizar outros meios de persistência dos dados, como por exemplo, banco de dados. Além disso, mais um de sistema de banco de dados (MySQL, Oracle, MongoDB, etc) poderá ser utilizado. Dependendo do que será persistido, será instanciado um tipo de persistência adequado.

Todas as classes que realizam operações de persistência, devem possuir os mesmos métodos (por exemplo, inserir, remover, listar, buscar), mas a implementação de cada método é diferente para cada tipo de persistência utilizado.



a) Discuta como você poderá fazer essas alterações na sua aplicação. Discuta como a maneira que seu programa está implementado favorece ou desfavorece a implementação dessa solicitação. Poderia ter sido utilizada solução melhor? A sua solução já está adequada para atender essa solicitação da melhor maneira? Escreva como comentário na sua classe principal.

Realize a implementação da solicitação com alguns exemplos de classes de persistência. Os métodos não precisam ser implementados, basta colocar a assinatura. Exemplifique a maneira como salvar um objeto de diferentes maneiras como comentários na sua classe principal.

## SOLUÇÃO

a) Para persistir em bancos de dados, em mais de 1 SGBD:  
o que será persistido?

Os conteúdos que também são guardados nos arquivos chamados de mocks, os quais contém separadamente:

- Clientes
- Fornecedores
- Funcionários
- Orçamentos
- Projetos

- TIPO adequado de persistência

- Pela maneira como foi modelado o sistema, com geração de dados em Listas e Maps (Dicionários), em seguida convertidas em JSONs (persistência em arquivos), seria natural utilizar um banco de dados não relacional, noSQL, como o MongoDB, que persiste dados em forma de documentos, com suporte a JSON. Seria uma solução atual e objetiva.
- Também seria possível utilizar SGBDs relacionais, como o PostgreSQL. Para isso, seria necessário fazer a modelagem conceitual, identificando as entidades e relacionamentos, baseada no diagrama de classes (UML) deste projeto, a fim de fazer o projeto lógico no SGBD.
- Então, poderia ser, então feito o carregamento dos dados já existentes nos SGBDs.
- Em seguida, através da interface criada Persistencia (em com.company.helper), com a assinatura dos métodos de persistência em bancos de dados, podem ser implementadas classes de persistência para

cada SGBD com pelo menos os métodos CRUD criados na interface, sendo sobrescritos de acordo com as especificidades do SGBD.

- Em seguida, as instâncias de persistência, poderiam ser invocadas sequencialmente à persistência de arquivos previamente existente (dos JSONs), fazendo as devidas conexões com os SGBDs e executando as transações desejadas.

## b) Implementação

Foram criadas as classes de permanencia que implementan a interface Persistencia em com.company.helper:

-PersistenciaMongo

-PersistenciaPostgres

Após instância com criação de cada Conexão ao seu respectivo SGBD, os métodos tem que ser implementados a fim de realizar as operações CRUD especificadas.

Por exemplo, no caso do Postgres, como é relacional, o trabalho será maior, pois tem que ser selecionado atributo por atributo de cada registro, ou validar um registro completo, tendo o tratamento.

Cada método de interesse seria invocado nos classes Gerenciadoras, para inserir (salvar) paralelamente aos arquivos JSON, por exemplo.

Já onde houver consultas, os métodos de busca ou listagem são invocados.

A mesma analogia vai para a deleção e atualização.