# 📋 RELATÓRIO DE IMPLEMENTAÇÃO — DESAFIO PRÁTICO JAVA

**Desenvolvido por:** Sandro Luis de Paula Junior
**Cargo pretendido:** Desenvolvedor de Software (Atendimento Técnico) — Júnior
**Empresa:** Projedata Informática
**Data:** 05/04/2025
**GitHub:** github.com/Cor4l92/desafio-java-projedata-funcionarios
**E-mail:** sandro.sd.luis@gmail.com
**WhatsApp:** +55 32 9 9106-3549

## 🎯 OBJETIVO

Desenvolver uma aplicação Java que gerencie uma lista de funcionários de uma indústria, realizando operações como inserção, remoção, formatação, cálculos, agrupamentos e ordenações, conforme especificado no desafio prático da Projedata Informática.

Este projeto foi desenvolvido como parte do processo seletivo para a vaga de **Desenvolvedor de Software (Atendimento Técnico) — Júnior**, demonstrando habilidades em lógica de programação, manipulação de dados, uso de coleções, formatação e boas práticas de organização de código.

## ✅ REQUISITOS IMPLEMENTADOS

### 1. Classe `Pessoa`

**Atributos:**

- `nome` (String)
- `dataNascimento` (LocalDate)

```
public class Pessoa {
    private String nome;
    private LocalDate dataNascimento;

    public Pessoa(String nome, LocalDate dataNascimento) {
        this.nome = nome;
```

```
        this.dataNascimento = dataNascimento;
    }

    // Getters e Setters
}
```

✅ **Status: Implementado com sucesso.**

## 2. Classe `Funcionario` (herda de `Pessoa`)

**Atributos adicionais:**

- `salario` (BigDecimal)
- `funcao` (String)

```
public class Funcionario extends Pessoa {
    private BigDecimal salario;
    private String funcao;

    public Funcionario(String nome, LocalDate dataNascimento, BigDecimal salario, St
        super(nome, dataNascimento);
        this.salario = salario;
        this.funcao = funcao;
    }

    // Getters, Setters e toString()
}
```

✅ **Status: Implementado com sucesso.**

## 3. Classe `Principal` — Execução das ações

### 3.1 – Inserir todos os funcionários

```
funcionarios.add(new Funcionario("Maria", LocalDate.of(2000, 10, 10), new BigDecima
funcionarios.add(new Funcionario("João", LocalDate.of(1990, 5, 12), new BigDecimal('
// ... demais funcionários
```

✅ **Status: Lista criada com todos os 10 funcionários, na ordem da tabela.**

### 3.2 – Remover o funcionário “João”

```
funcionarios.removeIf(f -> f.getNome().equals("João"));
```

✅ **Status: Funcionário "João" removido com sucesso. Restaram 9 funcionários.**

### 3.3 – Imprimir todos os funcionários com formatação

**Formatação aplicada:**

- Data: `dd/MM/yyyy` (ex: 10/10/2000)
- Salário: Separador de milhar como ponto, decimal como vírgula (ex: R$ 2.009,44)

```
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
String salarioFormatado = String.format("%.2f", f.getSalario().doubleValue()).replac
```

✅ **Status: Todos os dados exibidos corretamente conforme especificação.**

### 3.4 – Aplicar aumento de 10% no salário

```
for (Funcionario f : funcionarios) {
    BigDecimal novoSalario = f.getSalario().multiply(new BigDecimal("1.10"));
    f.setSalario(novoSalario);
}
```

✅ **Status: Aumento aplicado a todos os funcionários. Valores atualizados.**

### 3.5 – Agrupar funcionários por função em um Map

**Estrutura:**

- Chave: `String` (função)
- Valor: `List`

```
Map> mapaPorFuncao = new HashMap<>();
for (Funcionario f : funcionarios) {
    mapaPorFuncao.computeIfAbsent(f.getFuncao(), k -> new ArrayList<>()).add(f);
}
```

✅ **Status: Agrupamento realizado com sucesso.**

### 3.6 – Imprimir funcionários agrupados por função

```
Operador:
- Maria | 10/10/2000 | R$ 2.210,38
- Heitor | 19/11/1999 | R$ 1.718,99
```

✅ **Status: Impressão realizada conforme agrupamento.**

### 3.8 – Imprimir aniversariantes dos meses 10 e 12

```
if (mes == 10 || mes == 12) { ... }
```

✅ **Status: Funcionários "Maria" (10/10) e "Miguel" (14/10) exibidos corretamente.**

### 3.9 – Imprimir funcionário com maior idade

**Lógica:**

- Comparar pela data de nascimento (menor data = mais velho)
- Calcular idade: `anoAtual - anoNascimento`

```
Funcionario maisVelho = funcionarios.stream()
        .min(Comparator.comparing(Pessoa::getDataNascimento))
        .orElse(null);
int idade = Year.now().getValue() - maisVelho.getDataNascimento().getYear();
```

✅ **Status: "Caio" identificado como mais velho, com 64 anos.**

### 3.10 – Imprimir lista por ordem alfabética

```
funcionarios.sort(Comparator.comparing(Funcionario::getNome));
```

✅ **Status: Lista exibida em ordem alfabética (Alice, Arthur, Caio, ...).**

### 3.11 – Imprimir total dos salários

```
BigDecimal totalSalarios = funcionarios.stream()
        .map(Funcionario::getSalario)
        .reduce(BigDecimal.ZERO, BigDecimal::add);
```

✅ **Status: Total calculado e exibido: R$ 50.867,22 (após aumento de 10%).**

### 3.12 – Quantos salários mínimos cada funcionário ganha

**Salário mínimo considerado:** R$ 1212,00

```
double qtd = f.getSalario().doubleValue() / 1212.00;
```

✅ **Status: Todos os funcionários exibidos com quantidade de salários mínimos (ex: Miguel = 17,35).**

## 🛠️ FERRAMENTAS UTILIZADAS

- **IDE:** Visual Studio Code
- **Linguagem:** Java 17
- **Extensões:** Java Extension Pack, Code Runner
- **Terminal:** Integrado ao VS Code (Windows)
- **Sistema Operacional:** Windows 10/11

## 📁 ESTRUTURA DO PROJETO

```
ProjetoFuncionarios/
├── src/
│   └── main/
│       ├── Pessoa.java
│       ├── Funcionario.java
│       └── Principal.java
├── README.md
└── RELATORIO.md
```

## ▶️ COMO EXECUTAR O PROJETO

1. Abra o terminal na raiz do projeto.
2. Compile os arquivos:

   ```
   javac -d . src/main/*.java
   ```

3. Execute o programa:

```
java main.Principal
```

## 📤 ENTREGA

- Projeto exportado como `.zip`
- Relatório técnico (este documento em HTML, PDF e Word)
- Código-fonte organizado e comentado
- Saída do console validada e funcional
- Repositório público no GitHub: github.com/Cor4l92/desafio-java-projedata-funcionarios

## 💬 CONSIDERAÇÕES FINAIS

Este desafio foi uma excelente oportunidade para aplicar conceitos fundamentais da programação em Java, como:

- Orientação a Objetos (herança, encapsulamento)
- Manipulação de Collections (List, Map, Stream API)
- Tratamento de datas com `LocalDate`
- Formatação de saída para atender requisitos específicos
- Lógica de negócios e resolução de problemas práticos

Todas as funcionalalidades foram implementadas conforme especificado, e o programa foi testado com sucesso, gerando a saída esperada em todas as etapas.

Estou muito motivado pela oportunidade de participar do processo seletivo da **Projedata Informática** e coloco-me à disposição para eventuais esclarecimentos, entrevistas técnicas ou novos desafios.

Agradeço pela atenção!

### 👨‍💻 CONTATO

**Sandro Luis de Paula Junior**
📧 sandro.sd.luis@gmail.com
💻 GitHub
+55 32 9 9106-3549 (WhatsApp)

Relatório gerado em 05/04/2025 — Desafio Prático Java — Projedata Informática