

# Relatório do Trabalho Final de Programação Orientada a Objetos

---

Nome: Bernardo Nilson

Professor: Edson Moreno

## Desenvolvimento

### Classes e Aplicação

Dentro do diretório **src** estão todos os arquivos .java referentes ao programa:

- **App**: executa a Interface, somente. Nossa classe *main* está aqui.
- **Interface**: responsável por toda a parte visual do Sistema, além de controlar as ações e funções do nosso Programa.
- **Program**: principal classe, eu diria. Responsável por gerenciar as funções e objetos do nosso sistema.
- **Subject e Class**: Subject é a nossa disciplina, ao mesmo tempo que Class (Turma) herda seus atributos.
- **Person, Teacher e Student**: Person (Pessoa) mantém os atributos comuns entre Teacher (Professor) e Student (Aluno). Aqui existe uma relação de herança.

### Interface

A Interface gráfica do Programa é composta por três seções principais:

#### Cadastro:

- Cadastro de Aluno;
- Cadastro de Professor;
- Cadastro de Disciplina;

#### Visualização:

- Visualização de detalhes de Disciplina;
- Visualização de detalhes de Professor;
- Visualização de detalhes de Aluno;

#### Botões:

- Botão de Atualização;
- Botão de Export;
- Botão de Alocação de Turmas;

### Funcionalidades

Assim que o programa é executado, ele vai tentar ler os arquivos chamados **ExportStudents.txt** e **ExportTeachers.txt** do diretório atual e carregar suas informações no programa.

*Obs.* Vou deixar ambos os arquivos com algumas informações básicas, como solicitado.

Depois disso, você é capaz de cadastrar novos alunos, professores e disciplinas. Sendo necessário preencher todos os campos, com exceção da descrição da disciplina.

### Observações sobre os cadastros

Obs. Durante os cadastros dos professores e alunos, suas **carga-horárias já são definidas** por trás do programa (12h para professores e 28h para alunos).

Obs. Diferente de como acontece na PUCRS, **optei por manter IDs únicos para os identificadores de professor e alunos**, porém ainda é necessário informar manualmente, ou seja, não há incremento automático).

Obs. Durante o cadastro de disciplinas, o programa exibe todos os professores cadastrados no sistema, se desejar, pode atualizar a lista clicando no botão "Refresh" ao lado. É possível selecionar mais de um professor usando comandos de **ctrl** e **alt** do teclado.

Obs. Caso você tente cadastrar uma carga-horária semanal maior que **12h** para uma disciplina, o programa vai atribuir apenas 12h, visto que é o máximo que um professor consegue assumir semanalmente. Evitando possíveis problemas.

No momento de indicar as disciplinas que os alunos querem cursar, deve ser informado o número de matrícula do aluno e o código da disciplina.

**Não existe a possibilidade de exclusão e edição de dados do sistema, uma vez adicionados.**

### Visualização

Nos campos da esquerda existem as opções de alunos, professores e disciplinas cadastradas no sistema, você pode selecionar um por vez. Caso tenha cadastro com sucesso e não apareça, **lembre-se sempre de usar o botão de "Refresh"**. Com base no lado esquerda, ao clicar na sua respectiva visualização, aparecerão os seus detalhes, como especificado na descrição do trabalho.

### Exportar informações

Na última seção da Interface, existe um botão "**Exportar Dados**". Ele salva as informações de professores e alunos em dois arquivos .txt separados (um para cada).

ex. *ExportStudents.txt*:

```
Nome: Bernardo Nilson, Matrícula: 100101
Nome: Felipe Suarez, Matrícula: 100102
```

ex. *ExportTeachers.txt*:

```
Nome: Edson Moreno, Matrícula: 900201
Nome: Ana Maria Madalena, Matrícula: 900103
```

**Importante:** lembre-se que quando executado, o programa faz a leitura desses mesmos arquivos.

### Alocação de turmas

Na última seção, existe um botão "**Alocar em turmas**" para alocar todos os professores e alunos nas disciplinas cadastradas, conforme os critérios descritos no trabalho.

O programa **aceita clicar no botão mais de uma vez**, sem a necessidade de parar a execução da aplicação. Ao fazer isso, todas as informações referentes às turmas anteriores são **apagadas e reescritas** com os novos dados.

## Compilação e execução

Nesta seção, vou descrever o passo-a-passo até a execução do programa.

No computador da PUCRS, você vai utilizar o sistema operacional *Windows* e baixar o arquivo *.zip* do Moodle.

1. Abra o Gerenciador de Arquivos, navegue até o arquivo **BernardoNilson.zip** e descompacte-o na pasta atual.
2. Abra o aplicativo Visual Studio Code.
3. Clique em "**File**".
4. Clique em "**Open folder**" e selecione a pasta gerada pelo passo 1.
5. Na aba de extensões do VS Code, instale o kit de extensões para Java: "*Extension Pack for Java*" da própria Microsoft. Atalho: **Ctrl + Shift + X**
6. Clique no arquivo **App.java** da pasta ``src``
7. Na barra lateral esquerda, selecione a 4ª opção: "Run and Debug". Atalho: **Ctrl + Shift + D**
8. Clique no botão "**Run and Debug**" e pronto

O passo-a-passo foi testado nos computadores da PUCRS do *laboratório 401*, com as versões:

- java version "**1.8.0\_371**"
- Java(TM) SE Runtime Environment (build 1.8.0\_371-b11)
- Java HotSpot(TM) 64-Bit Server VM (build 25.371-b11, mixed mode)
- javac **17.0.9**

O programa também foi testado no Ubuntu 22.04, com as versões padrões do OpenJDK.

**Obrigado e boas correções!** 😊