# 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 clase *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` 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*. Assim como acontece na PUCRS, **optei por manter os identificadores de professor e alunos separados**, ou seja, o programa aceita caso tenha IDs iguais: aluno e professor com o mesmo ID. Minha motivação foi evitar duplicar os dados de duas listas diferentes (mas que poderiam usar Person).

> *Obs*. A metodologia para disciplica poder "selecionar" os professores que podem ministrar ela foi por meio de formações específicas, em qual área aquele professor é formado.

> *Obs*. Todas as formações estão disponíveis pelo programa e não podem ser alteradas pela Interface.

> *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.

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.**

**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 inforações 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, Formação: ORIENTACAO_A_OBJETOS
```

**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 *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 "Arquivo".
4. Clique em "Abrir nova pasta" e selecione a pasta gerada pelo passo 1.
5. Na barra da lateral esquerda, selecione a 4ª opção: "Executar e Debugar".
6. Clique no botão "Executar e Debugar".

O passo-a-passo foi testado nos computadores da PUCRS do *laboratório 40X*, com as versões:

- Java Runtime Enrironment: *X*
- Java Development Kit: *X*

O programa também foi testado no Ubuntu 22.04.

## Obrigado e boas correções! 😀