INSTITUTO FEDERAL Ceará	**CURSO** **TÉCNICO SUBSEQUENTE EM INFORMÁTICA PARA INTERNET** **PROJETO FINAL**	DATA: 2º semestre Turno: Noturno
	DISCIPLINA: **Programação Orientada à Objetos**	**Nota:**
	Professor: **Kaio Jonathas Alencar Gurgel**	
Aluno (a):		

Atenção: o trabalho deve ser realizado em TRIOS.

Projeto: Sistema de Gestão Acadêmica

🧩 Parte 1 – Estrutura de Classes

🎯 Objetivo:

Criar a base das classes usando **herança** e teoria de classes abstratas e iniciar a modelagem do domínio.

📌 Instruções:

- Crie uma **classe abstrata** `Pessoa`, com os atributos:
  - `nome`
  - `cpf` (privado)
  - `data_nascimento`
  - Um método abstrato `exibir_dados()`.
- Crie a classe `Aluno`, filha de `Pessoa`, com:
  - Atributo `matricula` (privado)
  - `notas` (lista de notas – privadas)
  - `disciplinas` (lista de objetos da classe `Disciplina`)
- Crie a classe `Professor`, filha de `Pessoa`, com:
  - Atributo `siape` (privado)
  - `disciplinas` (lista de objetos da classe `Disciplina`)
- Crie a classe `Disciplina`, com:
  - `codigo` (string)
  - `nome` (string)
  - `professor_responsavel` (objeto da classe `Professor`)
  - `alunos_matriculados` (lista de objetos `Aluno`)
- Implemente o método `exibir_dados()` de forma **polimórfica** nas subclasses `Aluno` e `Professor`.
- Implemente método para exibir informações das disciplinas também.

🧩 Parte 2 – Encapsulamento e Métodos

🎯 Objetivo:

Aplicar **encapsulamento**, **métodos de instância**, **classe** e **estáticos**.

📌 Instruções:

- Torne os atributos sensíveis (`cpf`, `matricula`, `notas`, `siape`) **privados** (__).
- Utilize `@property` e `@atributo.setter` para controlar acesso.

➕ Métodos de instância:

- `Aluno`:
  - `adicionar_nota(disciplina, nota)`
  - `calcular_media(disciplina)`
  - `matricular_em_disciplina(disciplina)`
  - `desmatricular_disciplina(codigo_disciplina)`
- `Professor`:
  - `adicionar_disciplina(disciplina)`
  - `remover_disciplina(codigo_disciplina)`
- `Disciplina`:
  - `Adicionar_professor(professor)`

🏛️ Atributos e Métodos de Classe:

- Atributo de classe: `total_alunos` e `total_professores`
- Método de classe: `exibir_total_cadastrados()`

🧩 Métodos Estáticos:

- `validar_cpf(cpf)`
- `validar_data(data_nascimento)`

🧩 Parte 3 – Sistema de Cadastro

🎯 Objetivo:

Instanciar objetos, trabalhar com listas de objetos e aplicar **polimorfismo** com `exibir_dados()`.

📌 Instruções:

- Permita cadastrar múltiplos alunos, professores e disciplinas.
- Armazene-os em listas separadas.
- Crie funções para:
  - Cadastrar novas entidades.
  - Listar todos os alunos, professores e disciplinas.

- Buscar aluno por matrícula, professor por SIAPE e disciplina por código.

🧩 Parte 4 – Persistência em Arquivos

🎯 Objetivo:

Implementar **leitura e escrita de arquivos**, com **tratamento de exceções**.

📌 Instruções:

- Carregue os dados dos arquivos na inicialização e faça as instanciações devidas.
- Salve os dados em arquivos `.txt` ao cadastrar novos objetos.
- Use blocos `try`, `except`, `finally` para tratar:
  - Arquivos inexistentes
  - Dados inválidos ou corrompidos
  - Duplicidade de registros
  - Matrícula em disciplina inexistente
  - Inserção de nota em disciplina não cursada

🧩 Parte 5 – Relatórios e Estatísticas

🎯 Objetivo:

Consolidar o sistema com geração de **relatórios**.

📌 Instruções:

- Relatórios:
  - Alunos aprovados (média >= 7), lembrando que os alunos no .txt possuem 3 notas?: n1,n2,média
  - Alunos reprovados
  - Professores com mais de X alunos nas disciplinas que ministram
  - Estatísticas gerais:
    - Número total de alunos, professores e disciplinas
    - Média geral de notas por disciplina

🧩 Parte 6 – Menus Interativos

🎯 Objetivo:

Criar a interface textual para navegação no sistema.

📌 Instruções:

📋 Menu Principal

```
1. Menu de Alunos
2. Menu de Professores
3. Menu de Disciplinas
4. Menu de Relatórios
5. Sair
```

👨‍🎓 Menu de Alunos

```
1. Cadastrar novo aluno
2. Listar alunos
3. Buscar aluno por matrícula
4. Matricular aluno em disciplina
5. Adicionar nota
6. Desmatricular disciplina
7. Voltar
```

👩‍🏫 Menu de Professores

```
1. Cadastrar novo professor
2. Listar professores
3. Buscar professor por SIAPE
4. Atribuir disciplina ao professor
5. Remover disciplina do professor
6. Voltar
```

📚 Menu de Disciplinas

```
1. Criar nova disciplina
2. Listar disciplinas
3. Buscar disciplina por código
4. Ver alunos matriculados na disciplina
5. Voltar
```

📈 Menu de Relatórios

```
1. Alunos aprovados
2. Alunos reprovados
3. Professores com muitos alunos
4. Estatísticas gerais
5. Voltar
```

📁 Estrutura modular sugerida:

Arquivo / Módulo	Conteúdo principal	Estimativa de linhas
`pessoa.py`	Classe abstrata `Pessoa`, `Aluno`, `Professor` com métodos e atributos	120–150
`disciplina.py`	Classe `Disciplina`, relação com `Professor` e `Aluno`	70–100
`persistencia.py`	Funções de leitura e escrita para os arquivos `.txt`	80–100

Arquivo / Módulo	Conteúdo principal	Estimativa de linhas
`cadastro.py`	Funções para cadastrar alunos, professores, disciplinas	80–120
`relatorios.py`	Funções de geração de relatórios diversos	70–100
`menus.py`	Menus principais e específicos de cada entidade	120–160
`main.py`	Execução do sistema, chamadas de menus	60–80

📊 Total estimado (com organização modular):

≈ 640 a 770 linhas

Detalhamento de Cada Módulo da estrutura (7 módulos)

---

1. `pessoa.py` – Classes `Pessoa`, `Aluno` e `Professor`

- **Objetivo**: Criar uma classe base abstrata `Pessoa` com os atributos comuns `nome`, `cpf`, `data_nascimento`, e um método abstrato `exibir_dados()`. As classes filhas `Aluno` e `Professor` herdarão de `Pessoa` e terão seus próprios atributos específicos.
- **Conteúdo**:
  - Classe abstrata `Pessoa` com atributos e métodos comuns.
  - Métodos estáticos para validação e formatação de CPF.
  - Classes `Aluno` e `Professor` com atributos específicos e implementação do método `exibir_dados()`.

---

2. `disciplina.py` – Classe `Disciplina`

- **Objetivo**: Criar a classe `Disciplina`, que irá armazenar informações sobre a disciplina (nome, código, professor e alunos associados).
- **Conteúdo**:
  - Atributos da disciplina (nome, código, professor e lista de alunos).
  - Métodos para adicionar alunos e professor à disciplina.
  - Métodos de exibição de informações da disciplina.

---

3. `persistencia.py` – Funções de Leitura e Escrita em Arquivo

- **Objetivo**: Criar funções para salvar e carregar os dados de `Aluno`, `Professor` e `Disciplina` de/para arquivos `.txt`.
- **Conteúdo**:
  - Funções para salvar dados em arquivos `.txt`.
  - Funções para carregar dados de arquivos `.txt`.
  - Utilização de `try...except` para tratamento de exceções, como falhas de leitura e escrita.

---

4. `cadastro.py` – Funções de Cadastro

- **Objetivo**: Criar funções para cadastrar novos `Aluno`, `Professor` e `Disciplina`.
- **Conteúdo**:
  - Funções para criar novos objetos e adicionar à lista de objetos de cada entidade.
  - Garantir que os dados inseridos sejam válidos, como CPF.
  - Salvar automaticamente os dados no arquivo correspondente.

---

5. `relatorios.py` – Geração de Relatórios

- **Objetivo**: Criar funções para gerar relatórios sobre os alunos e professores.
- **Conteúdo**:
  - Relatório de alunos aprovados e reprovados.
  - Relatório de professores com mais de um número determinado de alunos.
  - Estatísticas gerais, como a média das notas dos alunos, número total de alunos e professores, etc.

---

6. `menus.py` – Menus Principais e Específicos

- **Objetivo**: Criar os menus interativos para os usuários interagirem com o sistema.
- **Conteúdo**:
  - Menu principal, onde o usuário pode acessar as opções para alunos, professores e disciplinas.
  - Menus específicos para cadastrar e listar alunos, professores e disciplinas.
  - Menus para editar e excluir disciplinas, notas dos alunos, etc.
  - Menu de geração de relatórios.

---

7. `main.py` – Execução do Sistema

- **Objetivo**: Inicializar o sistema e chamar os menus apropriados com base na escolha do usuário.
- **Conteúdo**:
  - Execução do programa com o carregamento inicial dos dados.
  - Exibição do menu principal e interação com o usuário.
  - Chamada de funções dos outros módulos com base nas escolhas dos menus.

Três arquivos .txt (alunos.txt, professores.txt e disciplinas.txt) estão sendo disponibilizados para que sirvam como ponto de partida para o projeto.