

	Serviço Nacional de Aprendizagem Comercial	
EXERCÍCIO: Curso Técnico em Desenvolvimento de sistemas		Código em Python

#### Código estrutura:

- **CLASSES:** (ALUNO, DISCIPLINA, PROFESSOR, MATRICULA, NOTA) COM ATRIBUTOS E MÉTODOS CORRESPONDENTES:
- Aluno: Gerencia matrículas e relatórios.
- Disciplina: Controla vagas e alunos matriculados.
- Professor: Atribui disciplinas e insere notas.
- Matricula: Associa aluno, disciplina e nota.
- Nota: Armazena valor e observação.
- SistemaEscolar: Centraliza a lógica do sistema, simulando o fluxo do diagrama de sequência.

#### Fluxo do Diagrama de Sequência

O método SistemaEscolar.registrar\_em\_disciplina implementa o fluxo do diagrama de sequência:

1. Aluno solicita registro em disciplina.
2. Sistema verifica disponibilidade (vagas na disciplina).
3. Se houver vagas, cria uma matrícula.
4. Simula aprovação do administrador (impressão no console).
5. Confirma ou rejeita o registro.

#### Casos de Uso Suportados

- **Matricular Aluno:** SistemaEscolar.matricular\_aluno.
- **Registrar em Disciplina:** Aluno.matricular e SistemaEscolar.registrar\_em\_disciplina.
- **Atribuir Professor:** Professor.atribuir\_disciplina.
- **Inserir Notas:** Professor.inserir\_nota.
- **Visualizar Relatório:** Aluno.visualizar\_relatorio e Professor.visualizar\_relatorio.



```
from datetime import datetime
from typing import List

class Nota:
    def __init__(self, valor: float, observacao: str = ""):
        self.valor = valor
        self.observacao = observacao

    def __str__(self):
        return f"Nota: {self.valor} ({self.observacao})"

class Matricula:
    def __init__(self, id_matricula: int, aluno, disciplina, data_matricula: datetime):
        self.id = id_matricula
        self.aluno = aluno
        self.disciplina = disciplina
        self.data_matricula = data_matricula
        self.nota = None

    def associar_nota(self, nota: Nota):
        self.nota = nota
        print(f"Nota associada à matrícula {self.id} na disciplina {self.disciplina.nome}")

    def __str__(self):
        return f"Matrícula {self.id}: {self.aluno.nome} em {self.disciplina.nome} ({self.data_matricula.strftime('%Y-%m-%d')})"

class Disciplina:
    def __init__(self, nome: str, codigo: int, vagas: int = 10):
        self.nome = nome
        self.codigo = codigo
        self.vagas = vagas
        self.alunos_matriculados: List[Matricula] = []
        self.professor = None

    def registrar_aluno(self, matricula: Matricula) -> bool:
        if self.vagas > len(self.alunos_matriculados):
            self.alunos_matriculados.append(matricula)
            print(f"Aluno {matricula.aluno.nome} registrado na disciplina {self.nome}")
            return True
        else:
            print(f"Sem vagas na disciplina {self.nome}")
            return False
```



```
def atribuir_professor(self, professor):
    self.professor = professor
    print(f"Professor {professor.nome} atribuído à disciplina {self.nome}")

def __str__(self):
    return f"Disciplina: {self.nome} (Código: {self.codigo}, Vagas: {self.vagas})"

class Aluno:
    def __init__(self, nome: str, id_aluno: int):
        self.nome = nome
        self.id = id_aluno
        self.matriculas: List[Matricula] = []

    def matricular(self, disciplina: Disciplina, sistema) -> bool:
        return sistema.registrar_em_disciplina(self, disciplina)

    def visualizar_relatorio(self) -> str:
        relatorio = f"Relatório de {self.nome}:\n"
        for matricula in self.matriculas:
            nota_str = matricula.nota.valor if matricula.nota else "Sem nota"
            relatorio += f"- {matricula.disciplina.nome}: {nota_str}\n"
        return relatorio

class Professor:
    def __init__(self, nome: str, id_professor: int):
        self.nome = nome
        self.id = id_professor
        self.disciplinas: List[Disciplina] = []

    def atribuir_disciplina(self, disciplina: Disciplina):
        disciplina.atribuir_professor(self)
        self.disciplinas.append(disciplina)

    def inserir_nota(self, matricula: Matricula, valor: float, observacao: str = ""):
        nota = Nota(valor, observacao)
        matricula.associar_nota(nota)

    def visualizar_relatorio(self) -> str:
        relatorio = f"Relatório do Professor {self.nome}:\n"
        for disciplina in self.disciplinas:
            relatorio += f"- Disciplina: {disciplina.nome}\n"
            for matricula in disciplina.alunos_matriculados:
                nota_str = matricula.nota.valor if matricula.nota else "Sem nota"
                relatorio += f"  - {matricula.aluno.nome}: {nota_str}\n"
        return relatorio
```

```
class SistemaEscolar:
    def __init__(self):
        self.alunos: List[Aluno] = []
        self.disciplinas: List[Disciplina] = []
        self.professores: List[Professor] = []
        self.matriculas: List[Matricula] = []
        self.contador_matriculas = 0

    def matricular_aluno(self, nome: str, id_aluno: int) -> Aluno:
        aluno = Aluno(nome, id_aluno)
        self.alunos.append(aluno)
        print(f"Aluno {nome} matriculado no sistema")
        return aluno

    def registrar_em_disciplina(self, aluno: Aluno, disciplina: Disciplina) -> bool:
        if disciplina not in self.disciplinas:
            print(f"Disciplina {disciplina.nome} não encontrada")
            return False

        self.contador_matriculas += 1
        matricula = Matricula(self.contador_matriculas, aluno, disciplina, datetime.now())
        if disciplina.registrar_aluno(matricula):
            aluno.matriculas.append(matricula)
            self.matriculas.append(matricula)
            # Simula aprovação do administrador
            print(f"Administrador aprovou matrícula {matricula.id}")
            return True
        return False

    def adicionar_disciplina(self, nome: str, codigo: int, vagas: int = 10) -> Disciplina:
        disciplina = Disciplina(nome, codigo, vagas)
        self.disciplinas.append(disciplina)
        return disciplina

    def adicionar_professor(self, nome: str, id_professor: int) -> Professor:
        professor = Professor(nome, id_professor)
        self.professores.append(professor)
        return professor

# Exemplo de uso
def main():
    sistema = SistemaEscolar()

    # Criar disciplinas
    matematica = sistema.adicionar_disciplina("Matemática", 101, vagas=2)
    fisica = sistema.adicionar_disciplina("Física", 102, vagas=1)

    # Criar alunos
    aluno1 = sistema.matricular_aluno("João", 1)
    aluno2 = sistema.matricular_aluno("Maria", 2)

    # Criar professor
    professor = sistema.adicionar_professor("Dr. Silva", 101)

    # Atribuir professor
    professor.atribuir_disciplina(matematica)
```

```
        return False

    def adicionar_disciplina(self, nome: str, codigo: int, vagas: int = 10) -> Disciplina:
        disciplina = Disciplina(nome, codigo, vagas)
        self.disciplinas.append(disciplina)
        return disciplina

    def adicionar_professor(self, nome: str, id_professor: int) -> Professor:
        professor = Professor(nome, id_professor)
        self.professores.append(professor)
        return professor

# Exemplo de uso
def main():
    sistema = SistemaEscolar()

    # Criar disciplinas
    matematica = sistema.adicionar_disciplina("Matemática", 101, vagas=2)
    fisica = sistema.adicionar_disciplina("Física", 102, vagas=1)

    # Criar alunos
    aluno1 = sistema.matricular_aluno("João", 1)
    aluno2 = sistema.matricular_aluno("Maria", 2)

    # Criar professor
    professor = sistema.adicionar_professor("Dr. Silva", 101)

    # Atribuir professor
    professor.atribuir_disciplina(matematica)

    # Registrar alunos em disciplinas
    aluno1.matricular(matematica, sistema) # Deve registrar
    aluno2.matricular(matematica, sistema) # Deve registrar
    aluno2.matricular(fisica, sistema)      # Deve falhar (sem vagas)

    # Inserir notas
    for matricula in matematica.alunos_matriculados:
        professor.inserir_nota(matricula, 8.5, "Bom desempenho")

    # Visualizar relatórios
    print("\nRelatório do Aluno:")
    print(aluno1.visualizar_relatorio())
    print("\nRelatório do Professor:")
    print(professor.visualizar_relatorio())

if __name__ == "__main__":
    main()
```

### Notas Adicionais

- **Simplicidade:** O sistema é minimalista, mas cobre os principais pontos dos diagramas. Pode ser expandido para incluir validações (ex.: **aluno já matriculado**) ou persistência (ex: **banco de dados**).
- **Integração com diagramas:** O código reflete diretamente as estruturas e fluxos dos diagramas.
- **Extensões possíveis:**
- Adicionar interface gráfica (ex: com **Tkinter**).



Serviço Nacional de Aprendizagem Comercial

**EXERCÍCIO: Curso Técnico em Desenvolvimento de sistemas**

**Código em Python**

- Incluir persistência com **SQLite**.
- Implementar mais casos de erro (ex.: **professor não atribuído**).