

### Atividade prática – Reuso com Python (Associação de classes, Herança, Polimorfismo)

Implemente um sistema simples de gerenciamento de partições e sistemas operacionais. O sistema deve permitir criar partições em um gerenciador e associá-las a sistemas operacionais. Cada sistema operacional possui um gerenciador de arquivos estruturado de forma diferente.

#### Estrutura das classes:

- **GerenciadorDeParticoes:** Pode conter múltiplas partições (associação 1:N).
- **Particao:** Pode ter um sistema operacional associado (associação 0:1).
- **SistemaOperacional:** Superclasse que representa um sistema operacional genérico, podendo conter múltiplos arquivos (associação 1:N).
  - Métodos:
    - `listar_arquivos()`: Exibe a lista de arquivos no sistema.
- **Windows e Linux:** Subclasses de SistemaOperacional, cada uma com um atributo `gerenciador_de_arquivos` específico.
  - **Windows:** O gerenciador de arquivos é representado por uma lista de tuplas, onde:
    - O primeiro elemento da tupla representa o nome da pasta.
    - O segundo elemento é uma lista de arquivos contidos na pasta.
  - **Linux:** O gerenciador de arquivos é representado por um dicionário, onde:
    - A chave representa o nome da pasta.
    - O valor é uma lista de arquivos contidos na pasta.
- **Arquivo:** Representa um arquivo armazenado em um sistema operacional.

#### Métodos principais:

1. **GerenciadorDeParticoes:**
  - Criar uma nova partição.
  - Listar partições existentes.
2. **Particao:**
  - Associar um sistema operacional à partição.
3. **SistemaOperacional:**
  - Criar uma pasta no gerenciador de arquivos.

- Adicionar arquivos em uma pasta existente.
- Listar arquivos armazenados.

#### 4. **Windows e Linux:**

- Cada classe deve sobrescrever o método `listar_arquivos()` para exibir arquivos com um formato específico.

**Implementar um método polimórfico para Imprimir os arquivos de um determinado sistema operacional em uma determinada partição e o total ocupado em bytes.**

**Criar um arquivo de testes utilizando a biblioteca `pytest`. Fazer todos os testes possíveis.**

**Utilizar exceções em python quando necessário.**

**Exemplo de saída esperada:**

Criando partição "C:"...

Associando Windows à partição "C:"...

Criando pasta "Documentos" no Windows...

Adicionando arquivo "trabalho.docx" à pasta "Documentos"...

Criando pasta "Imagens" no Windows...

Adicionando arquivo "foto.jpg" à pasta "Imagens"...

Listando arquivos no Windows:

- Pasta: Documentos -> ['trabalho.docx']

- Pasta: Imagens -> ['foto.jpg']

Criando partição "/"home"...

Associando Linux à partição "/"home"...

Criando pasta "Projetos" no Linux...

Adicionando arquivo "codigo.py" à pasta "Projetos"...

Criando pasta "Downloads" no Linux...

Adicionando arquivo "musica.mp3" à pasta "Downloads"...

Listando arquivos no Linux:

- Projetos: ['codigo.py']

- Downloads: ['musica.mp3']