Soluzione Gestione Accessori Aspirapolvere 🧹

📝 Descrizione

Questo progetto implementa un sistema per gestire gli accessori specifici per diversi modelli di aspirapolvere utilizzando l' Abstract Factory. Ogni modello di aspirapolvere è dotato di tre accessori dedicati, ovverospazzola, filtro e sacchetto.

📐 Struttura del Progetto

```
aspirapolvere_project/
├── factories/
│   ├── abc456_factory.py     # Factory concreta per il modello ABC456
│   ├── accessorio_factory.py # Interfaccia astratta delle factory
│   └── xyz123_factory.py     # Factory concreta per il modello XYZ123
├── models/
│   ├── aspirapolvere.py      # Classe principale Aspirapolvere
│   ├── accessori/
│   │   ├── filtro.py        # Classi per i filtri
│   │   ├── sacchetto.py     # Classi per i sacchetti
│   │   ├── spazzola.py      # Classi per le spazzole
│   │   └── __init__.py      # File package accessori
│   └── __init__.py          # File package models
└── main.py                   # Punto di ingresso dell'applicazione
```

⚙ Requisiti di Sistema

- Python 3.7 o versioni successive
- Ambiente di sviluppo con pip configurato

🚀 Avvio del Progetto

1. Aprire il terminale
2. Navigare alla directory del progetto
3. Eseguire:

```
python main.py
```

📋 Output di Esempio

```
Aspirapolvere: SuperClean XYZ123
✓ Spazzola compatibile con il modello XYZ123
✓ Filtro compatibile con il modello XYZ123
✓ Sacchetto compatibile con il modello XYZ123

Aspirapolvere: PowerVac ABC456
✓ Spazzola compatibile con il modello ABC456
✓ Filtro compatibile con il modello ABC456
✓ Sacchetto compatibile con il modello ABC456
```

🔧 Design Pattern

- Abstract Factory Pattern per la creazione di famiglie di accessori compatibili

📚 Note Aggiuntive

Il progetto è strutturato per essere facilmente estendibile con nuovi modelli di aspirapolvere e relativi accessori.

Sviluppato come esempio di implementazione del pattern Abstract Factory

🧠 Ragionamento sulla Scelta del Pattern

Perché Abstract Factory?

L'Abstract Factory è stato scelto come pattern ottimale per questo scenario per i seguenti motivi:

1. **Famiglie di Prodotti Correlati**:
   - Ogni modello di aspirapolvere richiede una famiglia specifica di accessori (spazzola, filtro, sacchetto)
   - Gli accessori devono essere compatibili tra loro all'interno dello stesso modello
2. **Incapsulamento della Creazione**:
   - Nasconde i dettagli di implementazione delle classi concrete
   - Centralizza la logica di creazione degli accessori per modello
3. **Garanzia di Compatibilità**:

- Assicura che vengano creati solo accessori compatibili per ciascun modello
- Previene errori di abbinamento tra accessori e modelli

Vantaggi

1. **Manutenibilità**:
   - Facile aggiungere nuovi modelli di aspirapolvere
   - Modifiche localizzate nelle factory concrete
2. **Estensibilità**:
   - Semplice introduzione di nuovi tipi di accessori
   - Supporto per nuovi modelli senza modificare il codice esistente
3. **Coerenza**:
   - Garantisce la creazione di set completi di accessori compatibili
   - Riduce gli errori di configurazione

Svantaggi

1. **Complessità**:
   - Richiede la creazione di molte interfacce e classi
   - Può risultare eccessivo per sistemi semplici
2. **Rigidità della Struttura**:
   - L'aggiunta di nuovi tipi di accessori richiede modifiche all'interfaccia della factory
   - Tutte le factory concrete devono implementare i nuovi metodi