Salvataggio dati sicuro

Pattern decorator

Descrizione

- Si vuole implementare un sistema di salvataggio dati dove la sorgente originale è un semplice testo.

- A seconda della sensibilità dei dati o della destinazione, il sistema deve poter applicare diversi "layer" di trasformazione:
  - Base64 Encoding (per il trasporto)
  - Criptazione semplice (per la sicurezza)
  - Compressione (per risparmiare spazio).

Componenti

- Interfaccia: **`DataSource`** che definisce come i dati vengono manipolati:
  - **`void writeData(String data)`**
  - **`String readData()`**
- Il componente concreto: **`FileDataSource`** che simula la scrittura su un supporto fisico (semplicemente legge e scrive un attributo String content)
- Il Decoratore astratto: **`DataSourceDecorator`** che implementa **`DataSource`** e possiede una proprietà **`DataSource`**.
- I Decoratori concreti (3 classi) implementano la logica di trasformazione che si vuole applicare:
  - Encoding
  - Criptazione
  - Compressione

Decoratori

- EncryptionDecorator:
  - writeData: Deve "criptare" la stringa (usa cifrario di Cesare → sposta ogni carattere di +1 nella scala ASCII) prima di passarla al componente successivo.
  - readData: Deve "decriptare" (sposta di -1) dopo aver letto dal componente.
- CompressionDecorator:
  - writeData: Deve rimuovere gli spazi bianchi e convertire tutto in minuscolo (simulazione di compressione/normalizzazione).
  - readData: Restituisce il dato così com'è (la compressione è distruttiva in questo esercizio semplificato).
- Base64Decorator:
  - writeData: Utilizza **java.util.Base64** per codificare la stringa.
  - readData: Decodifica la stringa Base64 ricevuta.

Esempio d’uso di encoding

- **Base64 NON è cifratura, è** un **sistema di codifica** che trasforma dati binari in testo leggibile ASCII usando solo 64 caratteri (A-Z , a-z, 0-9, + /=)

```
String nome = "Decorator";
for(byte b : nome.getBytes()) {
    System.out.println(b);
}
String codificato = Base64.getEncoder().encodeToString(nome.getBytes());
System.out.println(codificato);
```

```
68
101
99
111
114
97
116
111
114
RGVjb3JhdG9y
```

L’output sarà la sequenza di byte che corrisponde al codice ASCII delle lettere che compongono la stringa nome

Infine il metodo **`encodeToString`** trasforma la sequenza di bytes in una stringa in base64