Esercizio SOLID

Obiettivo:

Estendere le classi create nel precedente esercizio (*Cerchio, Rettangolo, Triangolo, interfaccia FormaGeometrica* e classe astratta Figura) aggiungendo nuove funzionalità avanzate, facendo leva sul polimorfismo e applicando esplicitamente i principi SOLID.

Requisiti

Utilizzando come punto di partenza l'output del precedente esercizio, questo esercizio mira ad approfondire l'applicazione pratica dei principi SOLID, con particolare attenzione ai seguenti principi:

- **Single Responsibility Principle (SRP)**
- **Open/Closed Principle (OCP)**
- **Liskov Substitution Principle (LSP)**
- **Interface Segregation Principle (ISP)**
- **Dependency Inversion Principle (DIP)**

1. Creare un'interfaccia `Disegnabile` che definisca il metodo astratto:

   ```
   void disegna();
   ```

   (Applicazione di ISP e DIP)

2. Creare un'interfaccia `Comparabile` che definisca il metodo astratto:

   ```
   int compara(Figura altraFigura);
   ```

   La comparazione deve avvenire in base all'area della figura.
   (Applicazione di ISP e DIP)

3. Modificare la classe astratta `Figura` in modo che implementi l'interfaccia `Comparabile`, lasciando il metodo astratto `compara(Figura altraFigura)` da implementare nelle classi concrete. (Applicazione di OCP)

4. Estendere le classi `Cerchio`, `Rettangolo` e `Triangolo` implementando l'interfaccia `Disegnabile` e il metodo `compara()` ereditato dalla classe astratta. Ogni figura dovrà implementare:

   - il metodo `disegna()` mostrando a video una rappresentazione testuale (ASCII art molto semplice) della forma geometrica.
   - il metodo `compara()` confrontando le figure in base all'area, restituendo -1 se più piccola, 0 se uguale, e 1 se maggiore.
   - Per il **Cerchio**, mostrare un testo che includa il raggio.
   - Per il **Rettangolo**, mostrare un testo con base e altezza.
   - Per il **Triangolo**, mostrare un testo con i tre lati.

5. Creare una classe concreta `Quadrato` che estenda direttamente la classe astratta `Figura` (non Rettangolo), implementando le interfacce `FormaGeometrica`, `Disegnabile` e il metodo `compara()`. Questa classe deve contenere un attributo `lato`, implementando il calcolo di area e perimetro e i metodi richiesti.

   (Applicazione di LSP)

6. Realizzare una classe `GestoreForme` con responsabilità esclusiva della gestione delle figure (applicazione di SRP), che mantenga un array o una lista di oggetti del tipo `Figura` e includa:
   - `void aggiungiFigura(Figura f)` per aggiungere una figura.
   - `void mostraTutteLeFigure()` per stampare le informazioni e disegnare tutte le figure.
   - `double calcolaAreaTotale()` per la somma delle aree.
   - `Figura trovaFiguraMassima()` per individuare la figura con area maggiore usando il metodo `compara()`.
7. Creare una classe di test `TestEsercizioAvanzato` che dipenda solo da astrazioni e non da implementazioni concrete (applicazione di DIP), che:
   - Istanzi almeno un oggetto per ciascuna classe (`Cerchio`, `Rettangolo`, `Triangolo`, `Quadrato`).
   - Aggiunga questi oggetti al `GestoreForme`.
   - Visualizzi tutte le informazioni e i disegni delle figure.
   - Stampi l'area totale e la figura con l'area maggiore.

Istruzioni per lo Svolgimento:

1. Riutilizzare le classi Java create nell'esercizio precedente.
2. Implementare le nuove richieste sopra descritte, prestando particolare attenzione all'applicazione dei principi SOLID.
3. Eseguire e testare accuratamente il programma.
4. Zippare tutte le classi modificate e quelle nuove in un file `.zip`.
5. Caricare il file zip nella cartella `EserciziStudenti`.
6. Naming Convention: `EsercizioSOLID - Gruppo.zip`.