Dipartimento di Informatica Sistemistica e Comunicazione
Corso di Laurea in Informatica

Programmazione 2

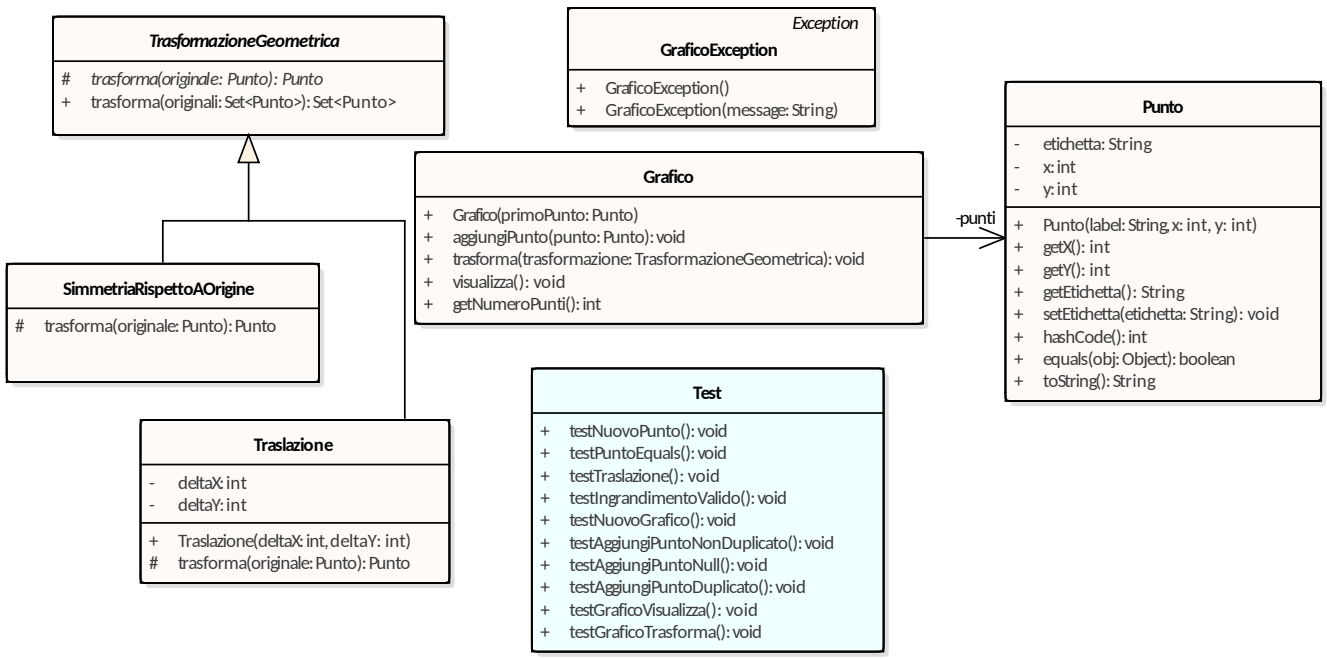
13 Giugno 2017 – Secondo Compitino

Testo parte di pratica

Si consideri un sistema di visualizzazione di grafici. Ogni Grafico include un insieme di Punti, ognuno rappresentato con coordinate bidimensionali e un'etichetta che sarà visualizzata nel grafico. Il sistema deve garantire che ogni grafico non possa contenere più punti con le medesime coordinate. Sui Grafici è possibile attuare delle trasformazioni geometriche, ovvero Traslazioni e SimmetriaRispettoAOrigine.

Implementare le classi come rappresentate dal seguente diagramma UML.

La classe `Test` (già fornita) contiene un insieme di casi di test che devono essere fatti girare di volta in volta in modo da verificare la corretta realizzazione del software. **Come requisito *minimo* per ottenere una valutazione positiva, lo studente deve garantire che la sua implementazione non presenti errori di compilazione e superi almeno 3 casi di test fra quelli dati.**



Classe `GraficoException`: rappresenta l'eccezione di tipo *checked* che viene prodotta quando un oggetto di tipo `Grafico` viene utilizzato erroneamente.

Classe `Punto`:

- ✓ rappresenta uno punto bidimensionale, caratterizzato dalle coordinate `x` e `y` (accessibili in sola lettura) e da un'`etichetta` (accessibile sia in lettura che in scrittura)
- ✓ definisce un costruttore che inizializza gli attributi. Il costruttore garantisce che, se l'`etichetta` in input è `null` o una stringa vuota, l'`etichetta` del punto viene inizializzata al valore di default `"UNDEF"`
- ✓ due `Punti` sono uguali se hanno le medesime coordinate, indipendentemente dall'`etichetta`.
- ✓ Il metodo `hashCode` ritorna l'hashcode dell'oggetto.
- ✓ il metodo `toString` restituisce una stringa che include le coordinate e l'`etichetta` del punto. Ad esempio, per il punto di coordinate `x`=2, `y`=5 e `etichetta` ="`prova`", il metodo restituisce `(prova: 2, 5)`.

Classe `Grafico`:

- ✓ rappresenta un grafico caratterizzato da un insieme (`HashSet`) di `punti` che ne fanno parte.

- ✓ definisce un costruttore che inizializza un `Grafico` contenente inizialmente un unico `Punto` (passato in input al costruttore) o nessun punto se il parametro è `null`.
- ✓ il metodo `aggiungiPunto` permette di aggiungere un `Punto` al `Grafico`, ma, nel caso in cui tale `Punto` sia uguale (medesime coordinate) ad uno già presente nel `Grafico`, il metodo solleva un'eccezione `GraficoException`. Il metodo non fa nulla se il punto passato come parametro è `null`.
- ✓ il metodo `visualizza` stampa a video i punti contenuti nel `Grafico`.
- ✓ il metodo `trasforma` modifica tutti i punti del `Grafico` in base alla `TrasformazioneGeometrica` in input, o non fa nulla se il parametro è `null`.
- ✓ il metodo `getNumeroPunti` restituisce il numero di punti contenuti nel `Grafico`.

Classi `TrasformazioneGeometrica, Traslazione, SimmetriaRispettoAOrigine`:

- ✓ la classe astratta `TrasformazioneGeometrica` e le corrispondenti classi concrete `Traslazione` e `SimmetriaRispettoAOrigine` permettono di attuare trasformazioni geometriche delle coordinate dei punti di un grafico.
- ✓ I metodi `trasforma(Punto)` e `trasforma(Set<Punto>)` attuano la trasformazione geometrica rispettivamente per un singolo punto o per un insieme di punti, restituendo un nuovo punto o un insieme di nuovi punti con le coordinate modificate. Si assuma che i parametri di tali metodi siano sempre valori diversi da `null`.
- ✓ La classe `Traslazione` è caratterizzata dagli spostamenti, `deltaX` e `deltaY`, che vanno sommati alle coordinate `x` e `y` per attuare la traslazione di un `Punto`. I valori di `deltaX` e `deltaY` si impostano con il costruttore e possono essere sia positivi, che nulli, che negativi.
- ✓ La classe `SimmetriaRispettoAOrigine` attua la trasformazione di ogni `Punto` invertendo il segno di ogni coordinata.