

Esercitazione 4: oggetti composti

Esercizio 1. Utilizzando la classe `MutablePoint` realizzata due settimane fa (il punto del piano, versione mutabile), realizzare una classe `MutableTriangle` per rappresentare triangoli secondo il seguente diagramma UML:

MutableTriangle
<ul style="list-style-type: none">– a : MutablePoint– b : MutablePoint– c : MutablePoint
<ul style="list-style-type: none">+ MutableTriangle(a : MutablePoint, b : MutablePoint, c : MutablePoint)+ getA() : MutablePoint+ getB() : MutablePoint+ getC() : MutablePoint+ translate(dx : double, dy : double) : void+ rotate(angle : double) : void+ perimeter() : double+ area() : double

Scrivere un semplice programma di prova che verifichi il corretto funzionamento della classe. Per la formula che determina l'area di un triangolo note le coordinate dei suoi vertici, e per verificare il corretto funzionamento del metodo `area`, può essere utile fare riferimento a questo sito Web: <http://www.mathopenref.com/coordtrianglearea.html>.

Esercizio 2. Determinare il diagramma UML della classe `ImmutableTriangle` per rappresentare **triangoli immutabili**, i cui metodi `translate` e `rotate` producono un nuovo oggetto triangolo invece di modificare l'oggetto ricevente. Per questa classe usare punti immutabili. Implementare in Java la classe `ImmutableTriangle` e verificarne il corretto funzionamento.

Esercizio 3. Data una istanza `t` di `MutableTriangle`, è possibile **modificare** il triangolo rappresentato da `t` senza invocare i metodi `translate` e `rotate` di `t`? Argomentare la risposta ed eventualmente fornire un esempio. Ripetere l'esercizio assumendo che `t` sia istanza di `ImmutableTriangle`.