

Sistemi per l'elaborazione dell'informazione 2

Collezioni, classi generiche e grafi

Un grafo è una coppia (V, A) dove V è l'insieme dei *vertici* e $A \subseteq V^2$ è l'insieme degli *archi*. Se $(x, y) \in A$, diciamo che c'è un arco da x a y . È tipico rappresentare i grafi disegnando un simbolo per ogni vertice e una freccia tra i vertici collegati da archi.

Dovete implementare una classe generica `Graph<T>` in cui la natura degli elementi in V è una classe generica. La struttura del grafo dovrà essere implementata come una lista di adiacenza: per ogni vertice $x \in V$ verrà memorizzato l'insieme dei *successori* di x , cioè l'insieme

$$\{y \mid (x, y) \in A\}.$$

Per la lista d'adiacenza usate un'implementazione di `Map`. Notate che l'implementazione dovrà essere istanziata utilizzando come parametri generici `T` e `Set<T>`.

La classe `Graph<T>` conterrà un costruttore vuoto e i metodi `addVertex()` e `addArc()` per aggiungere rispettivamente un vertice e un arco al grafo. Per semplicità, se il vertice è già presente ignorate la richiesta, mentre se la sorgente o la destinazione del grafo non sono presenti inseritele nel grafo.

Oltre ai metodi di costruzione, la classe `Graph` disporrà anche di un metodo `vertices()` che restituisce l'insieme dei vertici e un metodo `successors()` che restituisce quello dei successori di un nodo (il tipo restituito sarà quindi `Set<T>`). Implementate `toString()` in modo che il grafo venga rappresentato adeguatamente.

Per verificare che l'implementazione sia corretta potete aggiungere alla classe un metodo `main` che crei un grafo i cui vertici sono le stringhe corrispondenti ai numeri interi da 0 a una costante `MAX_VERTICES`, aggiungendo un arco tra il vertice i e il vertice j se j è un divisore di i , visualizzando poi il grafico invocando `toString`.

Tipi enumerati e grafi etichettati

Estendete la classe appena implementata per ottenere dei grafi etichettati, ovvero dei grafi i cui vertici hanno un'etichetta scelta in un insieme finito di valori (un tipo enumerativo). La classe avrà ora tipo generico `<T,E extends Enum<?>>`. Scrivete un nuovo metodo `addVertex()` che riceve, oltre a un vertice, la sua etichetta, e reimplementate `addArc()` in maniera che lanci un'eccezione se uno dei due vertici non è già stato inserito. Anche in questo caso, sovrascrivete il metodo `toString()` in modo che vengano visualizzate anche le etichette dei vertici.

Visualizzatore di grafi

Create una classe (derivata da `Shell`) il cui `main()` apre una finestra che contiene solo un `Canvas` che la occupa interamente. Il `main()` deve creare un grafo etichettato, per poi visualizzarlo in modo ragionevole (per esempio, disponendo i vertici secondo un poligono regolare e poi collegandoli; divertitevi a fare delle frecce decenti). Nella scrittura del `PaintListener` vi possono tornare utili `gc.drawOval()` e

`gc.drawLine()`. I più arditi possono inserire il **Canvas** all'interno di uno **ScrollComposite**, che permette di aggiungere automaticamente delle barre di scorrimento alla finestra nel caso in cui quest'ultima sia troppo piccola per visualizzare interamente il grafo.