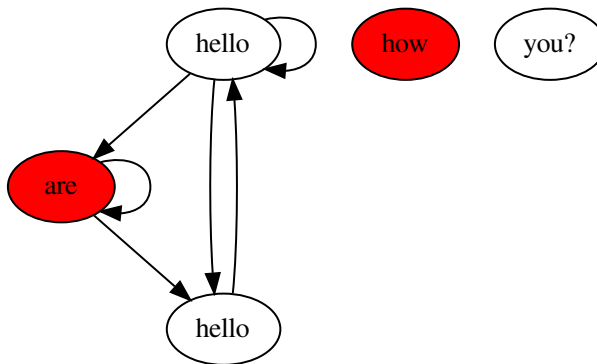


Esame di Programmazione II, 26 settembre 2019

Si crei un progetto Eclipse e, nella directory dei sorgenti, si crei il package `it.univr.graph`. Si copi al suo interno le classi `Graph.java` e `Main.java`.

Se si realizzano nuove classi, le si crei dentro il package `it.univr.graph`. Non si modifichi le dichiarazioni dei metodi. Si possono definire altri campi, metodi, costruttori e classi, ma devono essere **private**. La consegna fornita compila. Anche la soluzione che verrà consegnata dovrà compilare, altrimenti non verrà corretta.

Esercizio 1 [20 punti, si consegna `Graph.java`] Si completi la classe `Graph.java`, che implementa un grafo diretto. Un grafo contiene la lista dei suoi nodi e ciascun nodo contiene il suo valore (di tipo generico `E`) e la sua *stella uscente*, cioè la lista dei nodi a cui è connesso, senza ripetizioni. I nodi posso essere normali o colorati di rosso e questo nell'implementazione viene rappresentato dalle classi interne `Node` e `RedNode`. Un grafo nasce senza nodi, che si possono poi aggiungere con il metodo `add` (per i nodi normali) e `addRed` (per i nodi rossi). Archi fra nodi dello stesso grafo si possono aggiungere con il loro metodo `linkTo`. Un grafo è iterabile, quindi il suo metodo `iterator()` deve restituire un iteratore dei nodi del grafo. Infine, con il metodo `toString` è possibile avere una rappresentazione testuale del grafo in formato dot. Per esempio, il seguente grafo di stringhe:



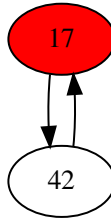
dovrebbe avere il seguente risultato per `toString`, che rispetta la sintassi del formato dot:

```
digraph {
  node0 [label="hello"]
  node0 -> node0
  node0 -> node2
  node0 -> node4
  node1 [label="how", style="filled", fillcolor="red"]
  node2 [label="are", style="filled", fillcolor="red"]
  node2 -> node2
  node2 -> node4
  node3 [label="you?"]
  node4 [label="hello"]
  node4 -> node0
}
```

Nel formato dot i nodi vengono dichiarati con nome e valore (`label`) e gli archi sono rappresentati come frecce. I nodi rossi hanno uno stile riempito (`filled`) in rosso. La scelta dei nomi dei nodi dentro la stampa in dot è irrilevante. L'importante è che i nomi siano diversi per nodi diversi. Nell'esempio sopra si è usato `node0`, `node1` ecc., ma ogni altra scelta è valida.

Esercizio 2 [11 punti, si consegna `Main.java`] Si completi la classe `Main.java` in modo che il suo metodo `main`

1. costruisca e stampi in dot il grafo di stringhe della pagina precedente;
2. costruisca e stampi in dot il seguente grafo di `java.lang.Integer`:



3. costruisca un grafo di stringhe contenente un unico nodo con valore `"hello"`, provando poi a legarlo a un nodo del primo grafo, il che dovrebbe generare un'eccezione.

*Se tutto è corretto, l'esecuzione del **Main** dovrebbe stampare:*

```
digraph {
  node0 [label="hello"]
  node0 -> node0
  node0 -> node2
  node0 -> node4
  node1 [label="how", style="filled", fillcolor="red"]
  node2 [label="are", style="filled", fillcolor="red"]
  node2 -> node2
  node2 -> node4
  node3 [label="you?"]
  node4 [label="hello"]
  node4 -> node0
}
```

```
digraph {
  node0 [label="17", style="filled", fillcolor="red"]
  node0 -> node1
  node1 [label="42"]
  node1 -> node0
}
```

Exception in thread "main" java.lang.IllegalArgumentException:
Cannot link nodes from distinct graphs