

Prima parte

Si deve sviluppare in Java una gerarchia di classi per il supporto di rete di Petri. Ad es. sia i posti che le transizioni hanno un nome. Si può introdurre una classe base (astratta) Entità che possiede un nome e definisce l'eguaglianza, l'hashCode e il toString() di una qualunque entità di rete (posto o transizione). La classe Posto eredita da Entità e introduce in più la marcatura del posto (ossia il numero dei token presenti), unitamente a metodi per conoscere la marcatura o per cambiare la marcatura. Due costruttori di Posto consentono di specificare la marcatura iniziale del posto o nessuna marcatura, nel qual caso inizialmente nel posto si pongono 0 token. La classe Transizione eredita anch'essa da Entità. Una classe base Arco descrive genericamente un arco tra un posto e una transizione o viceversa. Memorizza l'identità del posto nonché il peso dell'arco, unitamente a metodi per conoscere peso (non modificabile dinamicamente) etc. Classi eredi sono ArcoIn e ArcoOut. In un arco di ingresso, il posto è in input alla transizione. In un arco di uscita, il posto è in output alla transizione. Ogni transizione si caratterizza per la lista degli archi di ingresso (*preset*) e la lista degli archi di uscita (*postset*). Come caso particolare, un postset potrebbe essere vuoto: la transizione, in queste situazioni, consuma token ma non ne genera. Un preset vuoto potrebbe anche sussistere, per denotare una transizione sempre abilitata. Oltre al costruttore che riceve la lista preset e la lista postset, due metodi fondamentali di Transizione sono:

boolean abilitata()

void sparo()

che consentono, rispettivamente, di conoscere lo stato corrente di abilitazione o meno della transizione, e poterla eventualmente farla scattare (evento di sparo).

Scrivere una classe Main che nel suo metodo main() costruisca la rete di esempio. È conveniente memorizzare i posti in una mappa $M <String, Posto>$, e le transizioni in una LinkedList T di oggetti Transizione. Il main deve mostrare su output il contenuto di M ed il contenuto di T.

Si ricorda che è possibile costruire "al volo" una lista di archi, usando il metodo `java.util.Arrays.asList(...)` che ammette un vararg di oggetti.

Seconda parte

Si deve scrivere nello stesso package `poo.pn` una classe `PN` per l'esecuzione di una rete di Petri. La classe `PN` riceve a tempo di costruzione, la mappa `M` dei posti e la linked list delle transizioni del modello `PN`.

Scopo della classe `PN` è consentire, in modo controllato, di far evolvere la rete, sparando, se ce ne sono, una transizione alla volta. Si chiede di rendere disponibili due metodi:

```
void singleStep()
```

```
void multipleSteps(int n)
```

Il metodo `singleStep()` fa sparare una transizione e subito dopo mostra il contenuto della marcatura `M`. Se non ci sono transizioni abilitate, l'invocazione di `singleStep()` si deve concludere scrivendo "Deadlock!".

Il metodo `multipleSteps(int n)` spara in successione `n` transizioni e subito dopo ogni sparo visualizza la marcatura `M`. Ovviamente, prima di ogni step, se non ci sono transizioni abilitate, si scrive "Deadlock!" e si arrestano i passi.

Si suggerisce di mantenere due liste di transizioni, inizializzate nel costruttore di `PN`: abilitate e disabilitate. Quando si deve far sparare una transizione, prima si fa shuffling() della lista di transizioni abilitate:

```
Collections.shuffle(abilitate)
```

quindi si estrae la prima transizione da abilitate (rimuovendola), diciamola `t`. Si fa sparare `t` e si propagano gli effetti dovuti allo sparo di `t`. Prima di tutto si pone `t` tra le disabilitate (pessimismo). Subito dopo si eliminano dalla lista abilitate tutte quelle transizioni che prima erano abilitate ma che hanno perso l'abilitazione a causa dello sparo di `t`. Tutte queste transizioni vanno rimosse da abilitate ed inserite in disabilitate. Infine, si analizzano le transizioni presenti in disabilitate, e tutte quelle che risultano ora abilitate, si rimuovono da disabilitate e si aggiungono alla lista abilitate.

Modificare il metodo `main()` della classe `Main`, in modo da far sparare in successione 5 transizioni (se possibile) della rete di esempio.