

UNIVERSITÀ DEGLI STUDI DI FIRENZE
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
TESI DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

Analisi e Sviluppo di un Componente Java per la Simulazione
Interattiva di Reti di Petri Stocastiche

Candidato
Tommaso Scarlatti

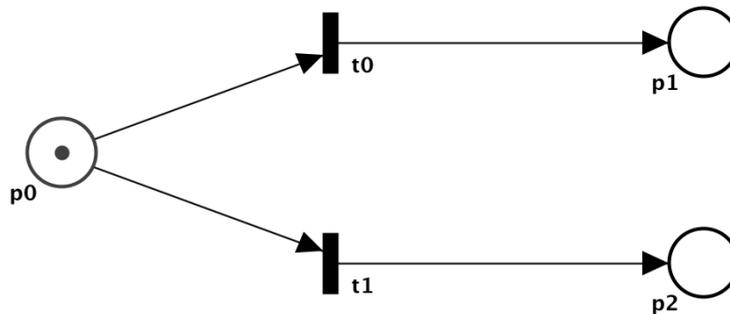


Relatore
Prof. Enrico Vicario

Correlatore
Ing. Marco Biagi

Anno Accademico 2016-2017

- ▶ Analisi e sviluppo di un simulatore interattivo di reti di Petri stocastiche
- ▶ Componente Java integrato nel tool ORIS



Rete di Petri

- ▶ Formalismo grafico/matematico per la modellazione di sistemi dinamici ad eventi discreti
- ▶ E' un grafo bipartito: posti e transizioni

Aree di applicazione

- ▶ Sviluppo di sistemi concorrenti
- ▶ Business Process Modeling
- ▶ ...

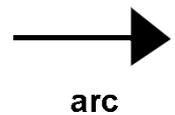
Semantica del formalismo

- ▶ **Marking:** distribuzione dei token nei posti
- ▶ **Firing rule:** regola che governa la rete.
Determina quando una transizione è abilitata
- ▶ Transizioni abilitate → una transizione scatta →
si modifica il marking

Estensioni

Esistono molte estensioni del formalismo base posto transizione (P/T):

- ▶ Reti di Petri Temporizzate
- ▶ Reti di Petri Stocastiche
- ▶ ...



Analisi dei requisiti

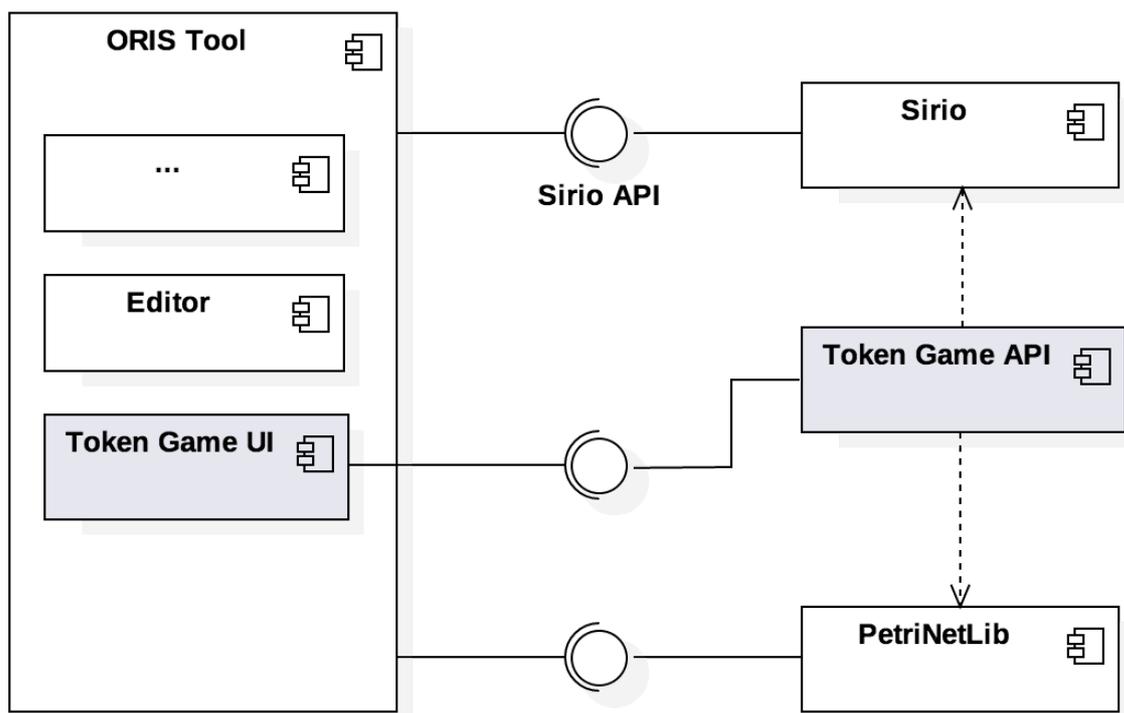
Requisiti funzionali

Sviluppare un **token game**, ossia un componente Java per la simulazione interattiva di reti di Petri stocastiche

Requisiti architetturali

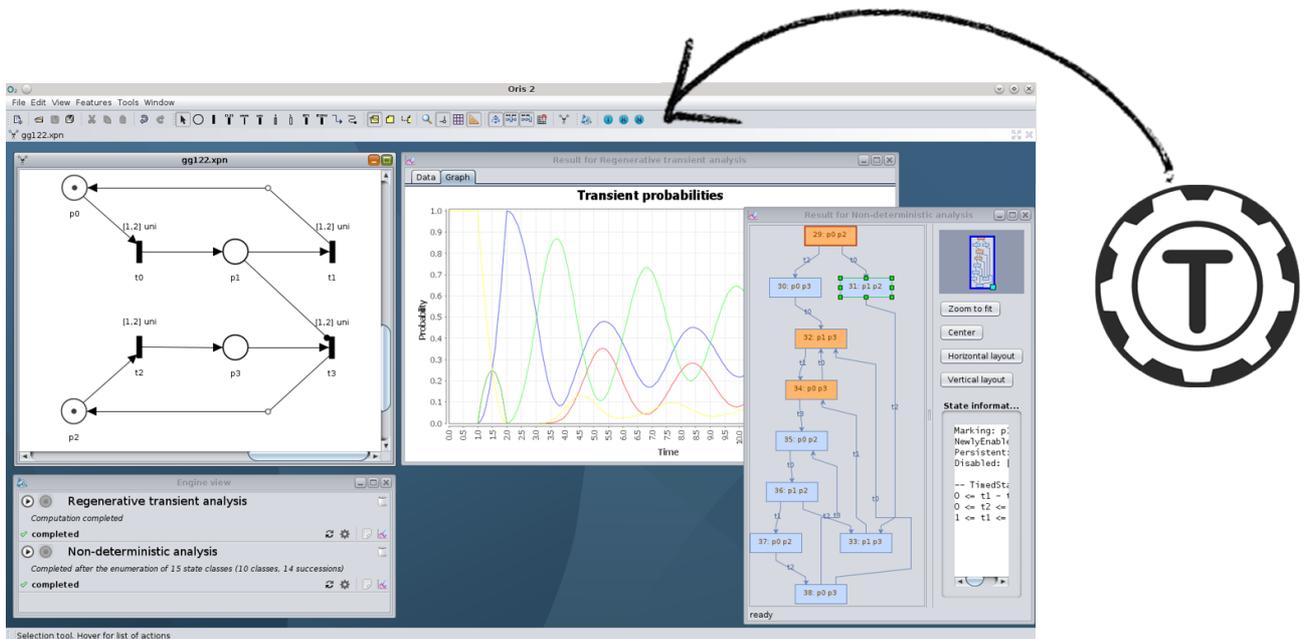
Integrare il componente all'interno del **tool ORIS**
(www.oristool.org)

- ▶ **ORIS** è un tool sviluppato dal *Software Technologies Lab* (STLAB) che permette la modellazione e l'analisi di sistemi reattivi temporizzati basati su varie classi di reti di Petri.



Interfaccia di ORIS

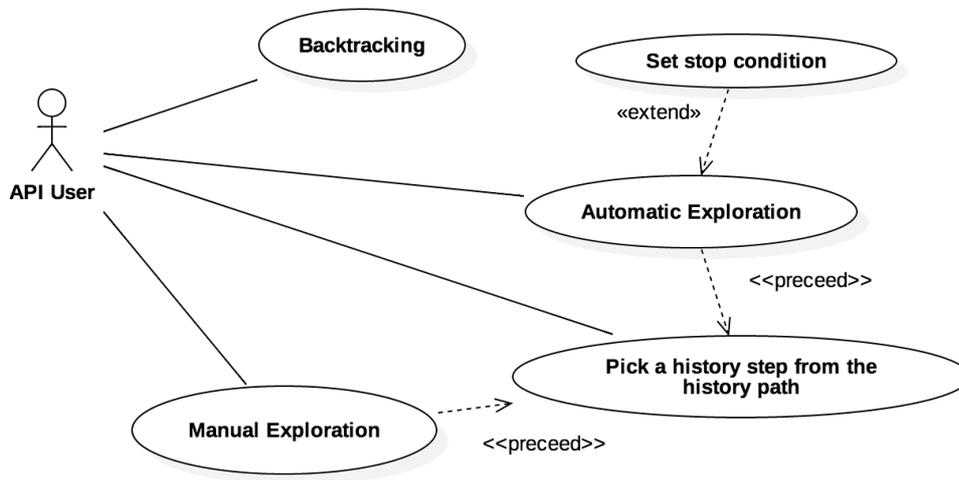
- ▶ Due viste: Editor View, Engine View
- ▶ **Token Game View** integrata nel tool Oris



Metodologia di sviluppo

- ▶ Studio di fattibilità: tempo, conoscenze da integrare
- ▶ Separare la logica di dominio del componente dalla sua rappresentazione grafica:
 1. API (Application Program Interface)
 2. GUI (Graphic User Interface)
- ▶ Test di unità e di integrazione (JUnit)

API Use Case Diagram



- ▶ Esplorazione Manuale/Automatica
- ▶ Backtracking
- ▶ Selezione di un passo del cammino

Token Game

Selected State			
Enabled Transition	Time Range	Firing Probability	Next Marking
t0	(0, 10)	0.4	2ab
t1	(3, 12)	0.6	a2b

Backtracking Steps: ⏪

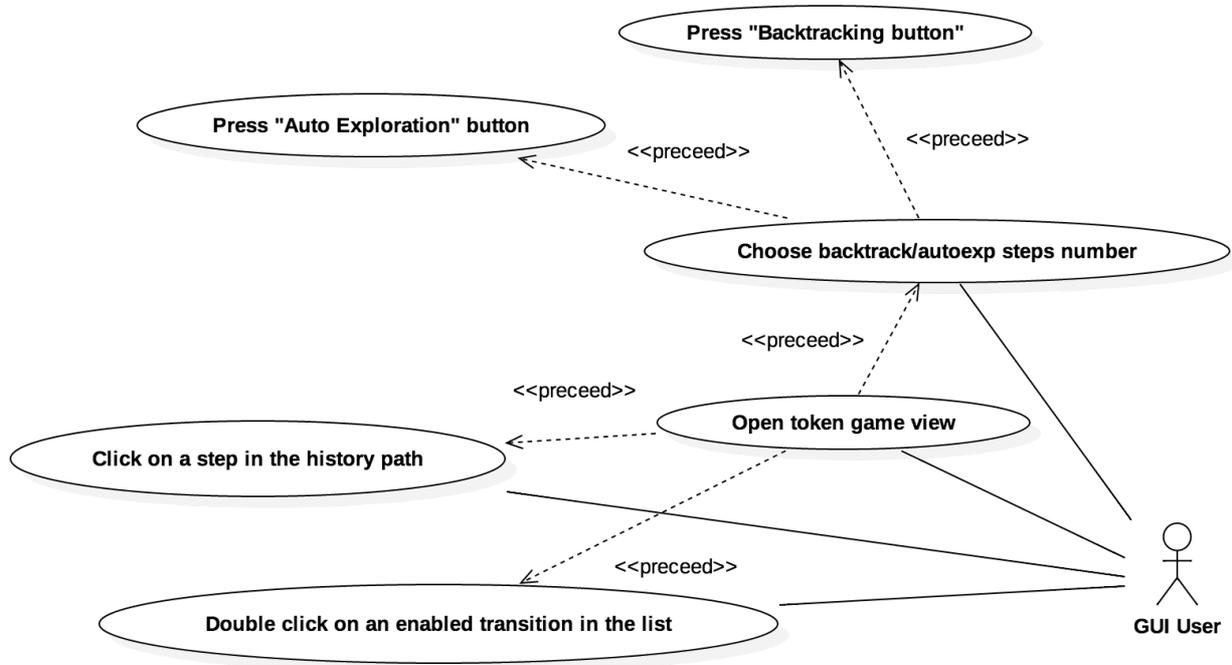
Exploration Steps: ↻ ➡

Stop condition:

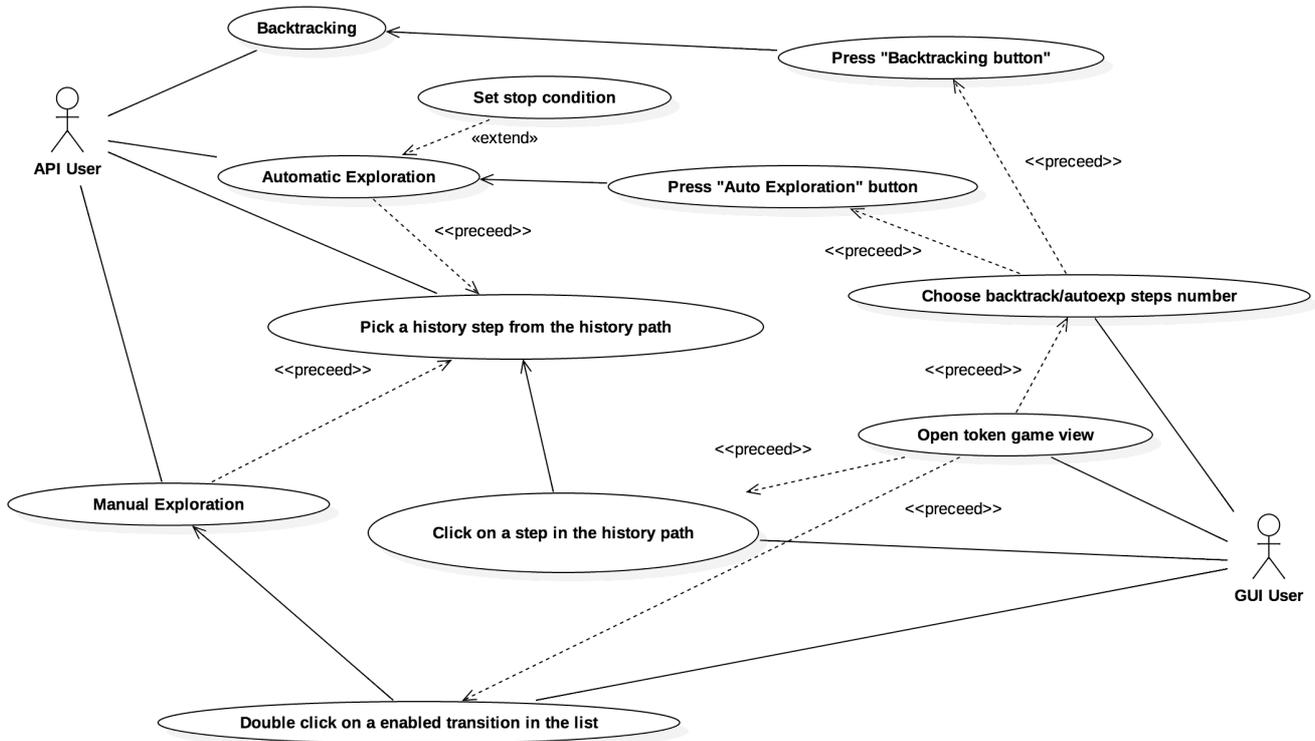
Path history				
Step	Fired Transition	Probability	Time Range	Marking
1	t1	0.6	(0, 12)	2ab
2	t3	0.8	(0, 1)	a
3	t2	0.3	(1, 8)	2b
4	t4	0.5	(0, 12)	2a
5	t2	0.6	(0, 7)	2ab

- ▶ Exploration
- ▶ Path History
- ▶ Petri Net

GUI Use Case Diagram

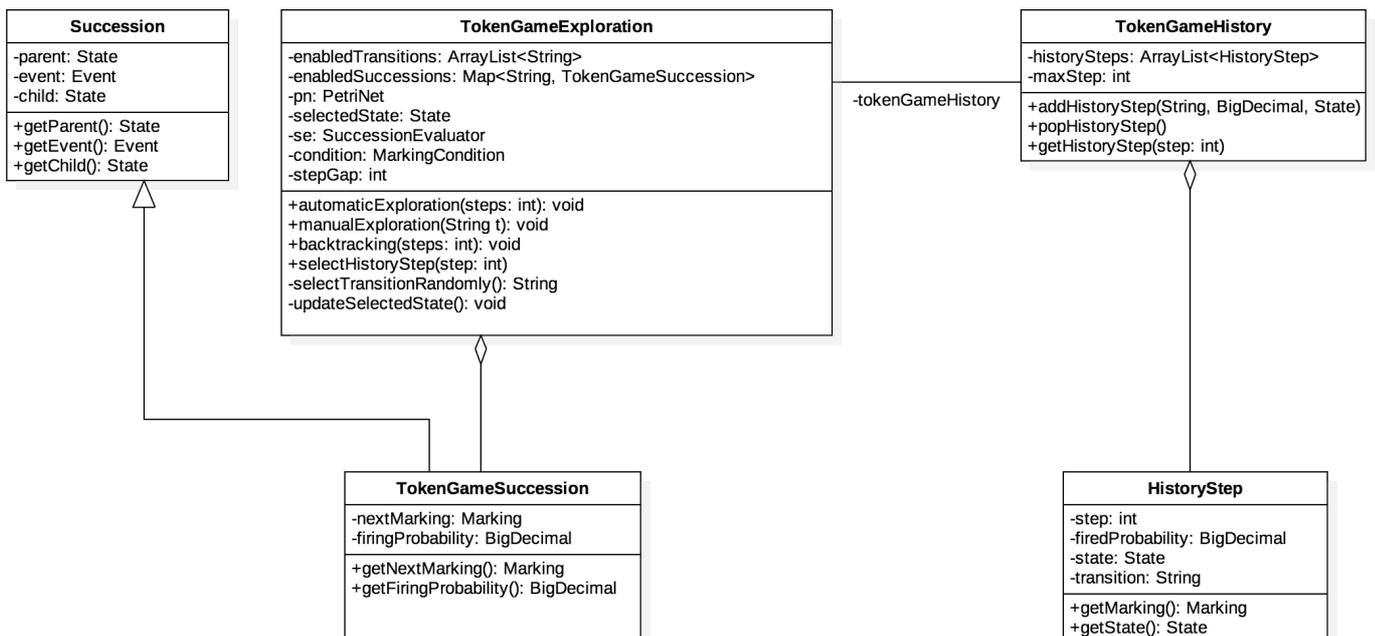


Tying pieces together..



Metodologia

- ▶ Implementare tutti i requisiti emersi nella fase di analisi
- ▶ Class diagram per catturare l'organizzazione delle classi mediante l'allocazione delle responsabilità tra esse



TokenGameExploration

- ▶ Classe centrale del progetto che espone metodi pubblici che soddisfano i requisiti funzionali

TokenGameHistory

- ▶ Rappresenta il container dei passi del cammino di esplorazione
- ▶ Modellata come una pila a stack (LIFO)

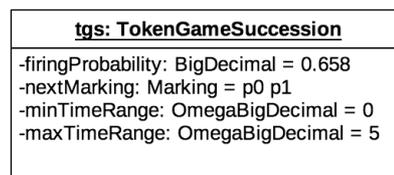
enabledTransitions

ArrayList <String>



enabledSuccession

Map <String, TokenGameSuccession>



- ▶ Java Swing + Window Builder
- ▶ *Listener* in ascolto degli eventi generati dall'utente
- ▶ Cinque possibili interazioni

The screenshot shows the Oris Tool interface for a Stochastic Petri Net. The main window displays a Petri net diagram with places p0, p1, p2, p3 and transitions t0, t1, t2, t3, t4. The interface includes a menu bar (File, Edit, View, Features, Tools, Window, Help), a toolbar, and several panels:

- Selected state: p3** panel:

Enabled transition	Time range	Firing probability	Next marking
t2	(0, 0)	0.5000	p2
t3	(0, 0)	0.5000	p0
- Path history** panel:

Step	Fired transition	Probability	Time range	Marking
1	t1	0.6321	(0, 1)	p0
2	t3	0.5000	(0, 0)	p3
3	t1	0.6321	(0, 1)	p0
- Backtracking** panel: Steps: [Backtrack]
- Exploration** panel: Steps: Stop condition: [Explore] [Load]

The Petri net diagram shows place p0 with 1 token, p1 with 1 token, p2 with 1 token, and p3 with 1 token. Transitions t0, t1, t2, t3, and t4 are connected to the places. Transition t1 has a uniform distribution [0,1] uni.

- ▶ Java Swing + Window Builder
- ▶ *Listener* in ascolto degli eventi generati dall'utente
- ▶ Cinque possibili interazioni

The screenshot displays the Oris Tool interface for a Stochastic Petri Net. The main window contains several panels:

- Selected state: p3**: A table showing enabled transitions and their properties.

Enabled transition	Time range	Firing probability	Next marking
t2	(0, 0)	0.5000	p2
t3	(0, 0)	0.5000	p0
- Path history**: A table showing the sequence of fired transitions and resulting markings.

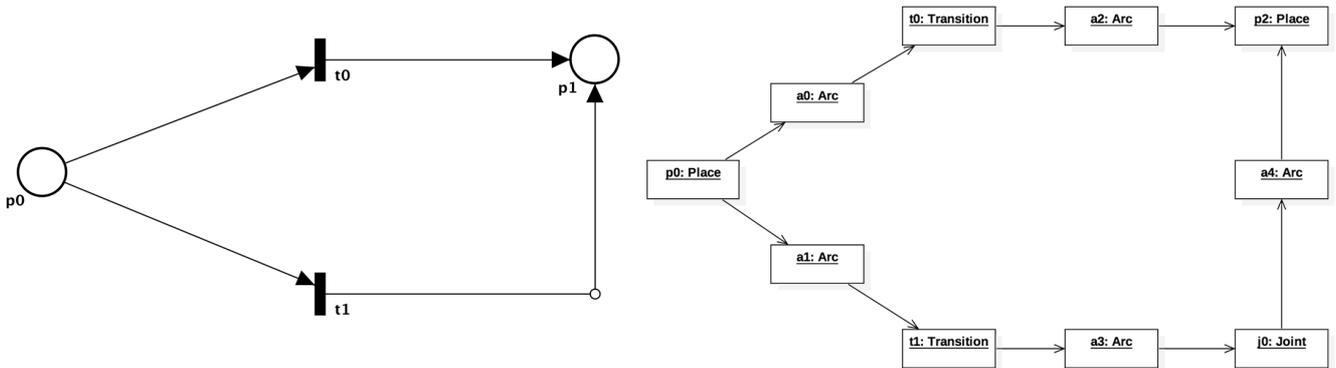
Step	Fired transition	Probability	Time range	Marking
1	t1	0.6321	(0, 1)	p0
2	t3	0.5000	(0, 0)	p3
3	t1	0.6321	(0, 1)	p0
- Backtracking**: A panel with a 'Steps' input field and a 'Backtrack' button.
- Exploration**: A panel with 'Steps' and 'Stop condition' input fields, and 'Explore' and 'Load' buttons.

The central area displays a Petri net diagram with places p0, p1, p2, p3 and transitions t0, t1, t2, t3, t4. The initial marking is at p0. The diagram shows a cycle of transitions and places, with a self-loop on p3.

Editor View VS Token Game View

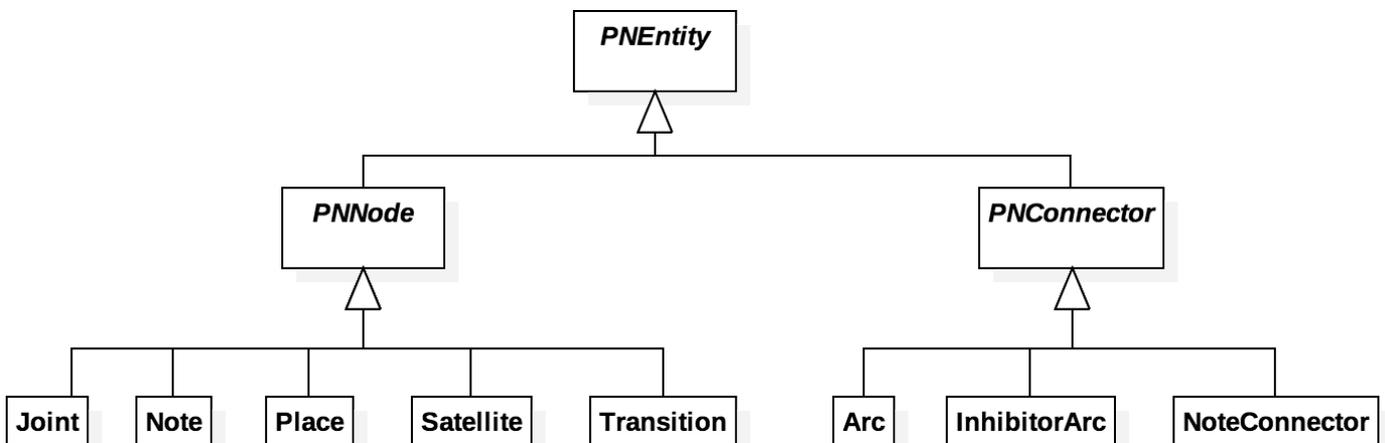
- ▶ Permettere alle reti delle due viste di evolvere in maniera indipendente l'una dall'altra
- ▶ **Deep copy** delle entità che formano la rete di Petri
 - ▶ *Shallow copy*: copia della struttura dati
 - ▶ *Deep copy*: copia dei singoli elementi della struttura

```
private Map<String, PNEntity> entities = new LinkedHashMap<>();
```



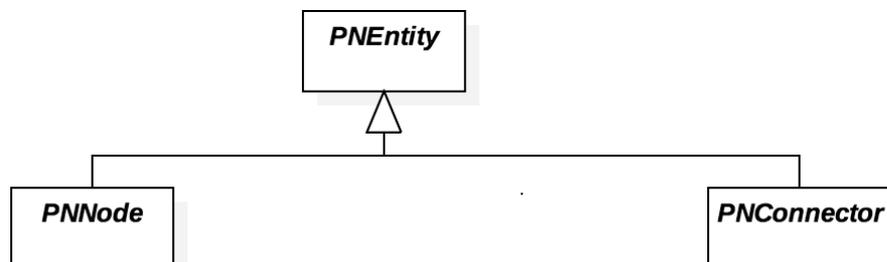
Problema

- ▶ Si ha a disposizione una collezione di **entità astratte**.
Non si può sapere a priori il tipo concreto
- ▶ Le entità formano complessivamente un **grafo orientato**.
Non se ne ha una rappresentazione esplicita



Problema

- ▶ Si ha a disposizione una collezione di **entità astratte**.
Non si può sapere a priori il tipo concreto
- ▶ Le entità formano complessivamente un **grafo orientato**.
Non se ne ha una rappresentazione esplicita

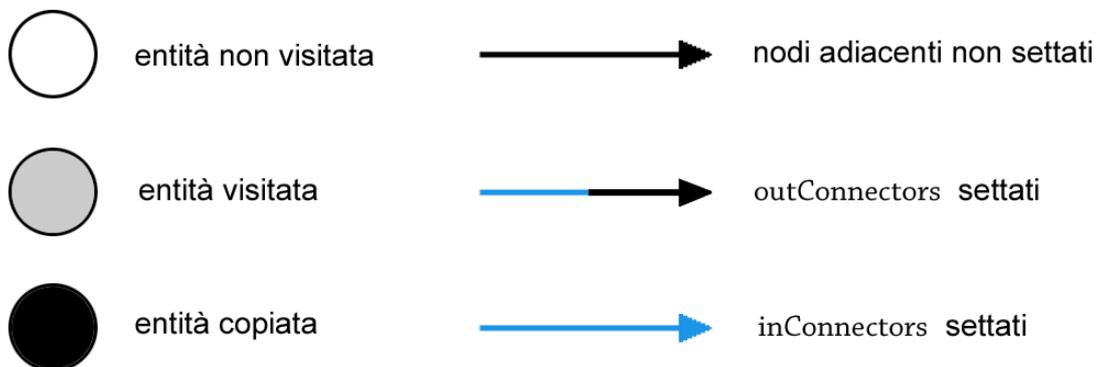


```
- Set<PConnector> inConnectors = new HashSet<>()
- Set<PConnector> outConnectors = new HashSet<>()
```

```
- PNode fromNode
- PNode toNode
```

Soluzione

- ▶ Utilizzo del pattern **Visitor** per separare l'operazione di copia dalla struttura dati
- ▶ Utilizzare una **DFS** (Depth First Search) per attraversare il grafo chiamando ricorsivamente l'operazione di visita sui connettori di uscita/nodi terminali



Riepilogo

- ▶ Analisi e sviluppo di un **simulatore interattivo** per reti di Petri stocastiche
 - ▶ API
 - ▶ GUI
- ▶ Integrato nel **tool ORIS**:
 - ▶ 25 classes
 - ▶ 6 different packages
 - ▶ 2613 code lines
- ▶ Editor View VS Token Game View:
 - ▶ Visitor (Deep copy)
 - ▶ DFS