

# Progettazione object-oriented di un'interfaccia grafica JavaFX per il simulatore Alchemist

Tesi in Programmazione ad Oggetti

Niccolò Maltoni  
0000719734

Alma Mater Studiorum - Università di Bologna  
Campus di Cesena

## 1 Introduzione

## 2 Alchemist

- Introduzione ad Alchemist
- Il modello di Alchemist

## 3 L'interfaccia classica

Lo scopo di questa tesi è la progettazione e la successiva implementazione di un'interfaccia grafica 2D per il simulatore *Alchemist* [4].

Lo scopo di questa tesi è la progettazione e la successiva implementazione di un'interfaccia grafica 2D per il simulatore *Alchemist* [4].

La nuova interfaccia permette di interagire con la simulazione a tempo di esecuzione e di vedere chiaramente rappresentate informazioni su di essa.

In particolare, è supportata una *struttura modulare di effetti* che rende facilmente osservabili determinate entità del sistema ed eventuali loro proprietà:

In particolare, è supportata una *struttura modulare di effetti* che rende facilmente osservabili determinate entità del sistema ed eventuali loro proprietà:

- l'effetto non fa più riferimento al singolo nodo, bensì costituisce una funzione dall'intero ambiente alla rappresentazione grafica.

In particolare, è supportata una *struttura modulare di effetti* che rende facilmente osservabili determinate entità del sistema ed eventuali loro proprietà:

- l'effetto non fa più riferimento al singolo nodo, bensì costituisce una funzione dall'intero ambiente alla rappresentazione grafica.
- gli *stack* di effetti realizzati possono essere serializzati su file di testo in formato JSON.

Si è scelto di mantenere un'interfaccia il più possibile *user-friendly*, mantenendo un design più simile ai simulatori a scopo videoludico per favorire l'utilizzo da parte di utenti inesperti.



Si è scelto di mantenere un'interfaccia il più possibile *user-friendly*, mantenendo un design più simile ai simulatori a scopo videoludico per favorire l'utilizzo da parte di utenti inesperti.

Lo stile estetico al quale si è deciso di allinearsi è il *Material Design* di Google<sup>1</sup> e la libreria grafica utilizzata per l'implementazione è stata *JavaFX*.

---

<sup>1</sup><https://material.io>

- 1 Introduzione
- 2 Alchemist
  - Introduzione ad Alchemist
  - Il modello di Alchemist
- 3 L'interfaccia classica

# Alchemist

## Introduzione ad Alchemist

Alchemist<sup>2</sup> [4] è un meta-simulatore estendibile completamente open-source che esegue su Java Virtual Machine (JVM), nato all'interno dell'Università di Bologna e reperibile su GitHub<sup>3</sup>.

---

<sup>2</sup><http://alchemistsimulator.github.io>

<sup>3</sup><https://github.com/AlchemistSimulator/Alchemist>

# Alchemist


## Introduzione ad Alchemist

Alchemist<sup>2</sup> [4] è un meta-simulatore estendibile completamente open-source che esegue su Java Virtual Machine (JVM), nato all'interno dell'Università di Bologna e reperibile su GitHub<sup>3</sup>.

L'idea dietro al progetto è quello di riuscire ad avere un framework di simulazione il più possibile generico, in grado di simulare sistemi di tipologia e complessità diverse, mantenendo le prestazioni dei simulatori non generici (come ad esempio quelli impiegati in ambito chimico [2]).

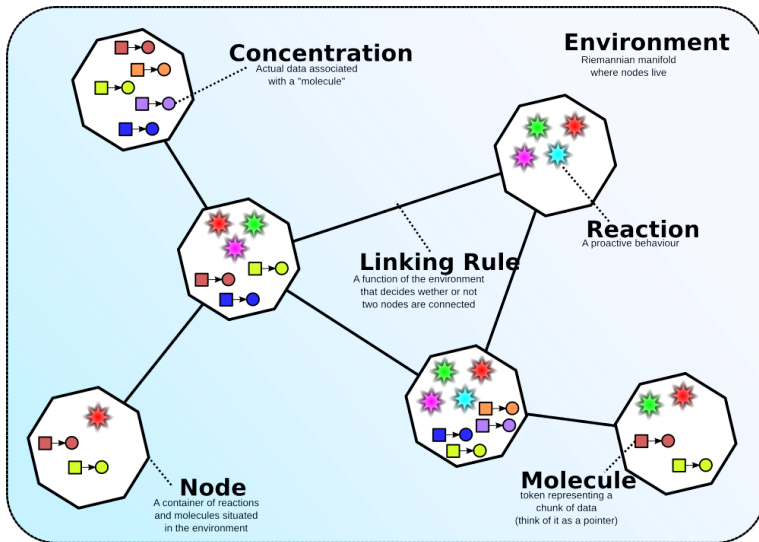
---

<sup>2</sup><http://alchemistsimulator.github.io>

<sup>3</sup><https://github.com/AlchemistSimulator/Alchemist> 

# Alchemist

## Il modello di Alchemist



# Alchemist

## Il modello di Alchemist

**Molecola** Una *Molecola* rappresenta il nome dato ad un particolare dato all'interno di un *Nodo*, del quale ne astrae parte dello stato.

# Alchemist

## Il modello di Alchemist

**Molecola** Una *Molecola* rappresenta il nome dato ad un particolare dato all'interno di un *Nodo*, del quale ne astrae parte dello stato.

**Concentrazione** La *Concentrazione* di una *Molecola* è il valore associato alla proprietà rappresentata dalla *Molecola*.

# Alchemist

## Il modello di Alchemist

**Molecola** Una *Molecola* rappresenta il nome dato ad un particolare dato all'interno di un *Nodo*, del quale ne astrae parte dello stato.

**Concentrazione** La *Concentrazione* di una *Molecola* è il valore associato alla proprietà rappresentata dalla *Molecola*.

**Nodo** Il *Nodo* è un contenitore di *Molecole* e *Reazioni* che risiede all'interno di un *Ambiente* e che astrae una singola entità.



# Alchemist

## Il modello di Alchemist

**Molecola** Una *Molecola* rappresenta il nome dato ad un particolare dato all'interno di un *Nodo*, del quale ne astrae parte dello stato.

**Concentrazione** La *Concentrazione* di una *Molecola* è il valore associato alla proprietà rappresentata dalla *Molecola*.

**Nodo** Il *Nodo* è un contenitore di *Molecole* e *Reazioni* che risiede all'interno di un *Ambiente* e che astrae una singola entità.

**Ambiente** L'*Ambiente* è l'astrazione che rappresenta lo spazio nella simulazione ed è l'entità che contiene i *Nodi*.

# Alchemist

## Il modello di Alchemist

**Molecola** Una *Molecola* rappresenta il nome dato ad un particolare dato all'interno di un *Nodo*, del quale ne astrae parte dello stato.

**Concentrazione** La *Concentrazione* di una *Molecola* è il valore associato alla proprietà rappresentata dalla *Molecola*.

**Nodo** Il *Nodo* è un contenitore di *Molecole* e *Reazioni* che risiede all'interno di un *Ambiente* e che astrae una singola entità.

**Ambiente** L'*Ambiente* è l'astrazione che rappresenta lo spazio nella simulazione ed è l'entità che contiene i *Nodi*.

**Regola di collegamento** La *Regola di collegamento* è una funzione dello stato corrente dell'*Ambiente* che associa ad ogni *Nodo* un *Vicinato*.

# Alchemist

## Il modello di Alchemist

**Vicinato** Un *Vicinato* è un'entità costituita da un *Nodo* detto “centro” e da un insieme di altri *Nodi* (i “vicini”).

# Alchemist

## Il modello di Alchemist

- Vicinato** Un *Vicinato* è un'entità costituita da un *Nodo* detto “centro” e da un insieme di altri *Nodi* (i “vicini”).
- Reazione** Una *Reazione* è un insieme di *Condizioni* sullo stato del sistema che qualora dovessero risultare vere innescherebbero l'esecuzione di un insieme di *Azioni*.  
Ogni *Nodo* è costituito da un insieme (anche vuoto) di *Reazioni*.

# Alchemist

## Il modello di Alchemist

**Vicinato** Un *Vicinato* è un'entità costituita da un *Nodo* detto “centro” e da un insieme di altri *Nodi* (i “vicini”).

**Reazione** Una *Reazione* è un insieme di *Condizioni* sullo stato del sistema che qualora dovessero risultare vere innescerebbero l'esecuzione di un insieme di *Azioni*.  
Ogni *Nodo* è costituito da un insieme (anche vuoto) di *Reazioni*.

**Condizione** Una *Condizione* è una funzione che associa un valore numerico e un valore booleano allo stato corrente di un *Ambiente*.

# Alchemist

## Il modello di Alchemist

**Vicinato** Un *Vicinato* è un'entità costituita da un *Nodo* detto “centro” e da un insieme di altri *Nodi* (i “vicini”).

**Reazione** Una *Reazione* è un insieme di *Condizioni* sullo stato del sistema che qualora dovessero risultare vere innescherebbero l'esecuzione di un insieme di *Azioni*.  
Ogni *Nodo* è costituito da un insieme (anche vuoto) di *Reazioni*.

**Condizione** Una *Condizione* è una funzione che associa un valore numerico e un valore booleano allo stato corrente di un *Ambiente*.

**Azione** Un'*Azione* è una procedura che provoca una modifica allo stato dell'*Ambiente*.

- 1 Introduzione
- 2 Alchemist
  - Introduzione ad Alchemist
  - Il modello di Alchemist
- 3 L'interfaccia classica

# L'interfaccia classica

L'architettura di Alchemist è progettata con paradigma *Model-View-Controller* [3] (MVC), di conseguenza la suddivisione tra componente grafica (*View*) e il blocco “logico” composto da *Model* e *Controller* è netta.



# L'interfaccia classica

L'architettura di Alchemist è progettata con paradigma *Model-View-Controller* [3] (MVC), di conseguenza la suddivisione tra componente grafica (*View*) e il blocco “logico” composto da *Model* e *Controller* è netta.

Questa distinzione è evidente anche per quanto riguarda l'utilizzo pratico del software:

# L'interfaccia classica

L'architettura di Alchemist è progettata con paradigma *Model-View-Controller* [3] (MVC), di conseguenza la suddivisione tra componente grafica (*View*) e il blocco “logico” composto da *Model* e *Controller* è netta.

Questa distinzione è evidente anche per quanto riguarda l'utilizzo pratico del software:

- una simulazione su Alchemist può venire lanciata da terminale, senza che alcuna interfaccia grafica sia necessaria per tutta la durata del periodo di esecuzione ...

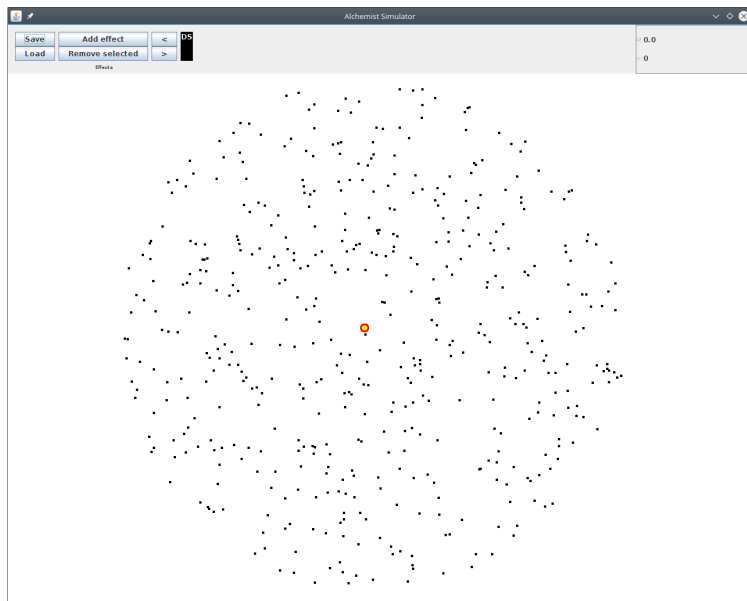
# L'interfaccia classica

L'architettura di Alchemist è progettata con paradigma *Model-View-Controller* [3] (MVC), di conseguenza la suddivisione tra componente grafica (*View*) e il blocco “logico” composto da *Model* e *Controller* è netta.

Questa distinzione è evidente anche per quanto riguarda l'utilizzo pratico del software:

- una simulazione su Alchemist può venire lanciata da terminale, senza che alcuna interfaccia grafica sia necessaria per tutta la durata del periodo di esecuzione ...
- ... oppure essere inizializzata, lanciata e controllata in tempo reale dalla sua interfaccia grafica.

# L'interfaccia classica



# Bibliografia

- [1] J. Bloch. *Effective Java (2nd Edition) (The Java Series)*. 2<sup>a</sup> ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2008. ISBN: 9780321356680.
- [2] D. T. Gillespie. «A general method for numerically simulating the stochastic time evolution of coupled chemical reactions». In: *Journal of Computational Physics* 22.4 (1976), pp. 403–434. ISSN: 0021-9991. DOI: [https://doi.org/10.1016/0021-9991\(76\)90041-3](https://doi.org/10.1016/0021-9991(76)90041-3). URL: <http://www.sciencedirect.com/science/article/pii/0021999176900413>.
- [3] G. E. Krasner, S. T. Pope et al. «A description of the model-view-controller user interface paradigm in the smalltalk-80 system». In: *Journal of object oriented programming* 1.3 (1988), pp. 26–49.
- [4] D. Pianini, S. Montagna e M. Viroli. «Chemical-oriented simulation of computational systems with ALCHEMIST». In: *Journal of Simulation* 7.3 (ago. 2013), pp. 202–215. ISSN: 1747-7786. DOI: [10.1016/j.simulation.2013.08.001](#)