Pepe Riccardo

# PCD - Assignment 02

Maggio 2025

## Analisi del problema

I requisiti dell'assignment 02 di programmazione concorrente e distribuita consistono nel realizzare una libreria che sfrutti la programmazione asincrona e un programma dotato di GUI che utilizzi la programmazione reattiva, entrambi con il fine di analizzare le dipendenze tra dei sorgenti Java, in particolare:

- La libreria che sfrutta la programmazione asincrona deve mettere a disposizione tre metodi asincroni che permettano di analizzare le dipendenze di un singolo file o di un singolo package o di un intero progetto. Inoltre ogni metodo deve essere associato ad un test.
- Il programma dotato di GUI deve possedere un componente per scegliere la root directory da analizzare, un pulsante per far partire l'analisi, un contatore per le dipendenze e uno per le classi/interfacce. Il risultato deve essere mostrato dinamicamente rappresentando mano a mano anche i risultati parziali.

Indipendentemente dalla versione della soluzione, per risolvere il problema delle dipendenze bisognerà passare attraverso i seguenti step:

- Trovare i file da analizzare
- Leggere i file dal file system
- Visitare il sorgente con Java Parser
- Ottenere e aggregare il risultato dell'analisi

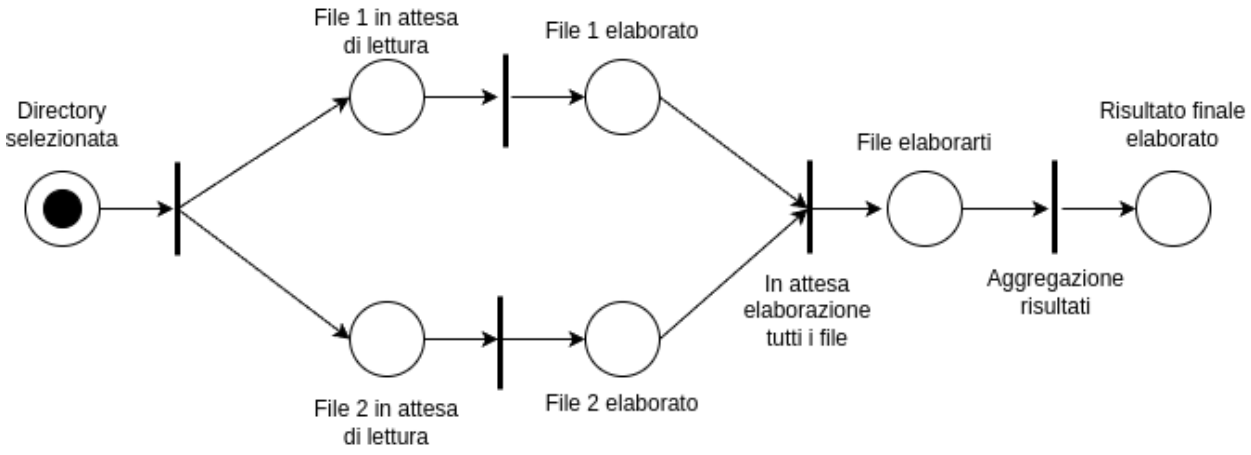
## Design e architettura della soluzione asincrona

Per realizzare la libreria asincrona in grado di identificare le dipendenze, si è scelto di sfruttare la libreria vertx di java in combinazione con JavaParser. Si è deciso di restituire per ogni metodo messo a disposizione dalla libreria una Promise, ovvero un risultato calcolato in modo asincrono. Il metodo per analizzare la singola classe è stato poi riutilizzato dagli altri metodi. Il risultato della Promise è un grafo rappresentato attraverso la libreria JGrapht.

## Design e architettura della soluzione reattiva

Per quanto riguarda la soluzione con GUI che sfrutta la programmazione reattiva, si è scelto di utilizzare JavaRX in combinazione con JavaParser. Il *model* di questo programma consiste in un metodo che restituisce un *Flowable*, osservato poi dalla GUI di Swing. Il Flowable passa attraverso diversi processi: viene generato un *Observable* che genera un'emissione per ogni file trovato all'interno di una directory di partenza, successivamente il file viene letto, effettuato il parsing ed infine viene restituito il risultato sotto forma di grafo che l'interfaccia grafica poi elabora e rappresenta. La GUI utilizza la libreria GraphStream, particolarmente adatta a rappresentare i grafi in modo dinamico.  E' fondamentale specificare che l'operazione di ricerca delle dipendenze avvenga in background rispetto all'EDT, in quanto rischierebbe di bloccare l'interfaccia grafica, a discapito della dinamicità del programma.

## Diagramma di Petri soluzione asincrona




## Diagramma di Petri soluzione reattiva