**Assignment-02**

**Approccio asincrono a Task – Framework Executors**

**Architettura e strategia risolutiva**

A livello architetturale in questo approccio si è pensato di realizzare la struttura di base implementando il pattern **MVC** per facilitare l’interazione del sistema con l’interfaccia grafica a linea di comando e con la GUI.

Raffinando il livello di dettaglio dell’architettura, si elencano gli elementi principali delle varie sezioni:

**Model:**

* DirectoryScanner
* ScanFolderTask
* CountLinesTask

Il Model contiene al suo interno il servizio offerto dal sistema, ovvero il **DirectoryScanner**, che funge da starting-point per l’esecuzione dei Task.

Per quanto riguarda la progettazione dei Task ne sono stati individuati due, lo **ScanFolderTask** si occupa dello scanning di una Folder, mentre il **CountLinesTask** del conteggio delle linee a partire da un file .java.

**Controller:**

* SourceAnalyzer

Il controller implementa l’interfaccia SourceAnalyzer che al suo interno contiene le definizioni di getReport() e analyzeSources().

**View:**

* ConsoleView
* GUIView

**Comportamento del sistema**

**Esecuzione dei Task**

L’esecuzione dei task sopra descritti è stata organizzata utilizzando una strategia ricorsiva, implementando un meccanismo di fork-join. Come executor è stato utilizzato un **ForkJoinPool** a cui è stato registrato uno ScanFolderTask iniziale.

Al suo interno ScanFolderTask esegue una **fork** per ogni subfolder al suo interno creando ulteriori ScanFolderTask, mentre per ogni documento .java effettua una fork facendo eseguire un CountLinesTask.

Al termine di tutte le fork, esegue una **join** per ogni Task figlio attendendo i risultati e restituendoli come risultato della propria terminazione.

Il CountLinesTask si occupa semplicemente del conteggio delle linee di codice.

**Metodo getReport()**

Il metodo getReport riceve in input tutte le informazioni di setup per poter inizializzare l’esecuzione e restituisce in output un **ForkJoinTask** (Che implementa l’interfaccia Future). In questo modo quando la CLI richiama getReport riceve istantaneamente una Future vuota e in modo asincrono, quando l’esecuzione dei Task termina, essa viene riempita con il risultato.

**Metodo analizeSources()**

Il metodo analizeSources() riceve anch’esso le informazioni per il setup dell’esecuzione, ma a differenza di getReport(), restituisce il riferimento ad un oggetto di tipo Result che conterrà il risultato. Successivamente tramite il pattern **Observer**, verrà notificato di nuovi risultati disponibili e in questo modo andrà a consultare la variabile corrispondente.

**Approccio asincrono ad eventi – Framework Vert.x**

**Architettura e strategia risolutiva**

Analogamente agli approcci precedenti.