Programmazione Avanzata
Specifica Progetti Settembre 2021

Logo è un linguaggio di programmazione didattico sviluppato alla fine degli anni 60. L'idea degli sviluppatori era quello di fornire un semplice strumento per "programmare" dei disegni. L'obiettivo del progetto è quello di sviluppare un ambiente di esecuzione per il Logo in Java.
Una descrizione dettagliata del linguaggio e della sua storia è disponibile nella pagina Wikipedia di Logo. Di seguito viene riportata una breve descrizione.

Il linguaggio Logo

Un programma Logo ha come obiettivo quello di disegnare un insieme di segmenti (rettilinei o curvi) all'interno di un'area di disegno. L'esecuzione di un programma fa muovere un cursore (storicamente rappresentato come una tartaruga o un triangolo) generando un disegno sottostante.

L'area dove viene realizzato il disegno prodotto dal programma consiste in una porzione di piano aventi dimensioni finite (base,altezza). La coordinata (0,0) indica il punto in basso a sinistra del piano. La posizione centrale dell'area di disegno, ossia quelle di coordinate (base/2,altezza/2), viene detta *home*. L'area è anche caratterizzata da un colore di sfondo. Il colore iniziale è *bianco*.

Il cursore è caratterizzato dai seguenti attributi:

- *Posizione*: posizione del cursore
- *Direzione:* indica l'angolo verso cui è puntato il cursore. L'angolo viene rappresentato con un intero nell'intervallo [0,360] dove lo 0 indica la direzione orizzontale verso destra (*ore 3*, se consideriamo un orologio).
- *Colore Linea*: indica il colore della linea prodotta dal cursore come conseguenza di uno spostamento.
- *Colore Area*: indica il colore dell'area che viene colorata quando una serie di segmenti producono un'area chiusa.
- *Plot:* è un parametro booleano che sta ad indicare se durante uno spostamento il cursore genera o meno un tracciato.

Per default, il cursore è posizionato nella *home*, ha una direzione di gradi 0, il colore della linea è il nero, mentre il colore dell'area è il bianco (come il colore di sfondo di default).

Il movimento del cursore viene definito per mezzo di un programma che è costituito da una lista di istruzioni della seguente forma:

- `FORWARD <dist>`: sposta il cursore in avanti verso la sua direzione (se si raggiungono i limiti dell'area il cursore si ferma al bordo);
- `BACKWARD <dist>`: sposta il cursore indietro rispetto la sua direzione (se si raggiungono i limiti dell'area il cursore si ferma al bordo);

- `LEFT <angle>`: ruota il cursore in senso *antiorario* dei gradi descritti dal parametro (gli angoli sono indicati come interi nel range [0, 360]);
- `RIGHT <angle>`: ruota il cursore in senso *orario* dei gradi descritti dal parametro (gli angoli sono indicati come interi nel range [0, 360]);
- `CLEARSCREEN`: cancella quanto disegnato;
- `HOME`: muove il cursore nella posizione di *home*;
- `PENUP`: stacca la penna dal foglio;
- `PENDOWN`: attacca la penna al foglio;
- `SETPENCOLOR <byte> <byte> <byte>`: imposta il colore della penna al colore rappresentato dal colore RGB rappresentato dai tre byte dati (<byte> indica un valore intero nel range [0,255]);
- `SETFILLCOLOR <byte> <byte> <byte>`: imposta il colore del riempimento di *un'area chiusa*;
- `SETSCREENCOLOR <byte> <byte> <byte>`: imposta il colore di background dell'area di disegno;
- `SETPENSIZE <size>`: indica la grandezza del tratto della penna, <size> è un intero di grandezza maggiore o uguale a 1;
- `RIPETI <num> [ <cmds> ]`: ripete la sequenza di comandi presenti nella lista dei comandi <cmds>.

Durante il movimento del cursore, come indicato dal programma, il cursore genera una sequenza di linee. Quando le linee *si chiudono* viene individuata *un'area chiusa*, questa consiste in una sequenze di linee che, partono e terminano nello stesso punto. Una volta individuata un'area chiusa, le linee non possono più far parte di una seconda area chiusa.

Il Progetto

Sviluppo Base (Valutazione massima 22)

Il progetto consegnato a questo livello di sviluppo dovrà:

- Definire una gerarchia di classi per rappresentare il programma Logo e le istruzioni;
- Definire una gerarchia di classi per rappresentare il risultato dell'esecuzione di un programma logo;
- Definire una gerarchia di classi per *controllare* l'esecuzione di un programma Logo;
- Definire le classi per leggere un programma da file;
- Definire la classi per scrivere (secondo un formato da voi definito e descritto nella relazione che accompagna il progetto) il risultato dell'esecuzione del programma.

Le classi prodotte devono supportare possibili estensioni come, ad esempio, la possibilità di introdurre nuovi comandi e di generare tipologie di segmenti diversi (curve, archi,...).
Per poter consentire di valutare l'efficacia e la correttezza di quanto sviluppato, il codice dovrà essere accompagnato da opportuni test ed esempi (anche questi da inserire nel folder "test" di gradle).

N.B. Per leggere un programma Logo da un file è sufficiente caricare il contenuto del file come array di String testuale e *segmentarlo* con gli spazi, fine linea e tablature. A quel punto si ottiene un'array di String (o una List<String>) che consentirà di ricostruire semplicemente la lista di comandi.

Implementazione Media (Valutazione massima 25)

In questo livello di implementazione, oltre alle funzionalità dello *sviluppo base*, dovrà essere sviluppata una applicazione a console che ricevuti in input un programma Logo produce descrive a console l'esecuzione del programma e produce in output il file generato.

Implementazione Avanzata (Valutazione massima 30 e Lode)

In questo livello di implementazione, oltre alle funzionalità dello *sviluppo base*, dovrà essere implementata una semplice interfaccia grafica che consenta di visualizzare l'esecuzione del programma Logo e, in modo opzionale, consenta di eseguire il programma stesso sia *passo-passo* che in modo automatico.

.

Griglia di valutazione

Il progetto verrà valutato secondo la seguente griglia di valutazione:

1. Definizione delle responsabilità, le responsabilità dovranno essere descritte in una breve relazione, in pdf, allegata al progetto (**NB: descrivere le responsabilità non vuol dire elencare le interfacce e le classi sviluppate**);
2. Coerenza dell'implementazione delle responsabilità. Per ogni responsabilità individuata al punto precedente indicare le interfacce e le classi che la implementano;
3. Rispetto dei principi SOLID;
4. Pulizia del codice ed assenza di Code Smell;
5. Estendibilità del codice prodotto;
6. Organizzazione del progetto (Gradle): il progetto gradle dovrà funzionare senza errori eseguendo i comandi "gradle build" e successivamente "gradle run" (quest'ultimo con eventuali parametri aggiuntivi).
7. Documentazione del codice (formado javadoc);
8. Presenza dei test (in proporzione al numero di classi e metodi testati);
9. Uso dei principi e metodologie viste a lezione;
10. Rispetto delle specifiche di consegna.

Modalità di Consegna

Il progetto dovrà essere consegnato in una cartella denominata CognomeNomeMatricola in un archivio *zip, tgz* o *tar.gz* con lo stesso nome. **L'uso di altri formati di consegna non è permesso! Nel caso il progetto non verrà valutato e considerato come non consegnato (non potendo quindi partecipare alla prova scritta)**.

All'interno della cartella CognomeNomeMatricola si dovrà trovare:

- I sorgenti del progetto *gradle*
- Un pdf con la descrizione delle responsabilità assegnate, delle loro implementazioni, e delle istruzioni necessarie a valutare il progetto.