Progetto Programmazione Avanzata JLogo

Realizzatore

Luca Bianchi, Matricola: 114150

Responsabilità assegnate (e relative interfacce)

Space

Rappresenta lo spazio bidimensionale (contraddistinto da larghezza e altezza) dell'applicazione. Mantiene al suo interno il cursore dell'applicazione, gestendo inoltre la sua posizione (preoccupandosi anche di spostarlo) e la sua direzione a cui punta.

Cursor

Ha la responsabilità di rappresentare un cursore dell'applicazione. Mantiene quindi come informazione quelle relative alla dimensione del tratto della linea disegnata, il colore della linea disegnata, il colore dell'area della figura (quando chiusa) e se sta "stampando" o meno (plot).

Color

Rappresenta un classico colore RGB, caratterizzato quindi dai tre parametri R (red), G (green) e B (blue).

Angle

Mantiene le informazioni di un angolo, rappresentato come un intero. Trattandosi di un angolo si preoccupa anche di far ruotare l'angolo in senso "orario" (right) o "antiorario" (left).

Coordinate

Questa interfaccia ha la responsabilità di rappresentare un punto nello spazio, ed è quindi contraddistinto da una x (l'ascissa) e una y (l'ordinata), entrambe double.

Shape

Rappresenta semplicemente una forma generica che potrà essere memorizzata nel disegno.

Line

(estende l'interfaccia Shape, in quanto è una figura che può essere salvata nel disegno)

Mantiene le informazioni per rappresentare una linea, perciò il punto d'inizio, il punto di fine, il colore e la dimensione del tratto. Tuttavia non mette a disposizione i metodi setter perché dovrebbe essere immutabile una volta creata.

Figure

(estende l'interfaccia Shape, in quanto è una figura che può essere salvata nel disegno)
Mantiene le informazioni per rappresentare una figura chiusa. Quindi ha la responsabilità di gestire le linee che la compongono e il colore di riempimento.

Drawing

Rappresenta il disegno dell'applicazione Logo dove verranno disegnate le varie figure (shape). Ha come responsabilità perciò quella di tener conto del colore dello sfondo, delle figure che contiene e quella di gestire l'aggiunta di una nuova linea.
Infine si preoccupa anche di fornire dei metodi per la stampa delle istruzioni operate sul disegno (aggiunta ed eliminazione di figure) che di default però non fanno nulla.

Command

Ha la responsabilità di rappresentare un generico comando dell'applicazione Logo.
Sarà inoltre ogni comando che implementa tale interfaccia ad avere la responsabilità di effettuare i dovuti controlli sui campi che accetta.

MovementCommand

(estende l'interfaccia Command)
Rappresenta un generico comando di movimento del cursore, di conseguenza sarà caratterizzato da una certa distanza.

RotateCommand

(estende l'interfaccia Command)
Rappresenta un generico comando di rotazione del cursore, perciò sarà contraddistinto da un certo angolo (Angle), quello della rotazione.

CommandExecutor

Ha la responsabilità di eseguire il comando (Command) che gli viene passato. Una volta ricevuto il comando, andrà a modificare lo stato del disegno (Drawing) e dello spazio (Space) associato ad esso su cui lavora.

CommandParser

Questa interfaccia ha la responsabilità di restituire una lista di comandi (Command) a partire da una stringa data.
Più precisamente prende un'unica stringa, perciò nell'implementazione prima andrà ad effettuare uno split della stringa suddividendola in più righe (una per comando), e successivamente basandosi sulla prima parola (se riconosciuta tra i comandi ammessi) restituirà il comando corretto.

FileReader

Ha la responsabilità di leggere il contenuto di un file (preso come file o con il relativo path), che sarà quello dove è presente il programma Logo, e restituire la lista dei comandi letti. Per fare ciò utilizzerà ovviamente il parser (CommandParser) al suo interno.

ShapeWriter

Rappresenta un traduttore di forme (Shape), perciò presa una forma (o presa una lista di forme nel metodo di default), restituisce la rispettiva rappresentazione sotto forma di stringa.

FileWriter

Ha la responsabilità di scrivere su file (passato come file o come path) il contenuto del disegno (Drawing) passato. Al suo interno, per fare ciò, utilizzerà ovviamente uno ShapeWriter per ricevere la rappresentazione a stringa di una forma.

Controller

Rappresenta il controller dell'applicazione Logo, che permette alle altre componenti (la console e l'interfaccia grafica) di interagire con il modello dell'applicazione.
Quindi si preoccuperà di istanziare il tutto e ricevere i comandi da parte delle "viste" e trasformarli sotto forma di istruzioni che faranno cambiare di stato le varie componenti del modello.

Implementazioni

Space

- GridSpace

Cursor

- SimpleCursor

Color

- RGBColor

Angle

- SimpleAngle

Coordinate

- Coordinate2D

Shape

- (interfaccia Line)

    - StraightLine
- (interfaccia Figure)
    - Polygon

Drawing

- SimpleDrawing (nel modulo “api”)
    - A sua volta estesa da:
        - ConsoleDrawing (nel modulo “console”)
        - GuiDrawing (nel modulo “gui”)

Command

- (Interfaccia MovementCommand)
    - BackwardCommand
    - ForwardCommand
- (Interfaccia RotateCommand)
    - LeftRotateCommand
    - RightRotateCommand
- ClearScreenCommand
- HomeCommand
- PlotCommand
- RepeatCommand
- SetFillColorCommand
- SetPenColorCommand
- SetPenSizeCommand
- SetScreenColorCommand

CommandExecutor

- SimpleCommandExecutor

CommandParser

- DefaultCommandParser

FileReader

- DefaultFileReader

ShapeWriter

- DefaultShapeWriter

FileWriter

- DefaultFileWriter

Controller

- LogoController

Istruzioni per la valutazione/esecuzione

GUI

La parte relativa al modulo gui, per lo sviluppo dell'interfaccia grafica, non funziona correttamente in mancanza di tempo per il relativo sviluppo.

Console

Per eseguire l'applicazione da console è necessario invocare il relativo script di gradle (gradlew).
Se ad esempio si posiziona il relativo file (con ad esempio nome: "inputFile.txt") al di fuori della cartella del progetto, allora il relativo comando per l'esecuzione (considerando che si crea un disegno di dimensione 1000x1000 e che il file in output contenente il disegno sarà ad esempio chiamato "outputFile.logo") sarà:

```
$ ./gradlew :console:run --args="../../inputFile.txt ../../outputFile.logo 1000 1000"
```

Questo perché gli argomenti del programma saranno rispettivamente:

- Il file in input
- Il file in output
- La larghezza che avrà il disegno
- L'altezza che avrà il disegno

Un possibile esempio di programma Logo (da poter passare in input) è ad esempio:

```
PENDOWN
RIPETI 30 [FORWARD 100 RIGHT 156]
```

… che serve per generare tale disegno: