PROGETTO PROGRAMMAZIONE AVANZATA JLOGO

Realizzatore : Davide Nappo 110157

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. Per l'implementazione è stato utilizzato il pattern di programmazione MVC creando 3 packages :

- Model (qui si trovano i componenti che presentano i dati in oggetto)

- View (il terminale che presenta i dati all'utente)

-Controllers(qui si trovano le componenti che si occupano dell'interazione con l'utente)

E' stato creato inoltre un package utility con alcune classi utili alla realizzazione del progetto

Descrizione interfacce e responsabilità associate

Le interfacce e le responsabilità assegnate e trovate sono :

Field (model) : Rappresenta il foglio di carta dove avverà il disegno.Il Field ha una base,altezza,un colore di sfondo e un centro. Il Field contiene anche un cursore e si occupa di aggiornare la posizione del cursore al suo interno. Le proprietà del Field(colore,base,altezza) verranno restituite attraverso un metodo in modo da poter essere scritte su un file con il formato corretto.

Cursor(model) : Interfaccia che rappresenta un cursore. Il cursore si trova all'interno di un Field e ha una posizione, l' angolo verso cui e' puntato,un colore della linea che produce spostandosi, un plot(un parametro che indica se il cursore sta toccando il foglio oppure no) e lo spessore del tratto.(N.B nella specifica del progetto si parlava anche del colore di riempimento dell'Area,il fillColor.Ho ritenuto assegnare tale responsabilità al Poligono stesso).

Segment (model): Interfaccia che rappresenta un segmento generico. Un segmento una volta tracciato, a rigore di logica non puo' essere modificato.Ha un punto di inizio e un punto di fine, un colore,e una dimensione(size)

Polygon(model):Interfaccia che rappresenta un poligono(una figura geometrica piana delimitata da una linea spezzata chiusa). Un poligono è quindi formato da una lista di segmenti.Presenta un metodo che ritorna la lista di segmenti di cui è formato e un metodo che ritorna il colore di riempimento(nella specifica era una caratteristica del cursore,ho preferito spostarla qui).

IMPLEMENTAZIONI

DefaultCursor(model): Implementazione di default dell'interfaccia Cursor.

DefaultField(model): Implementazione di default dell'interfaccia Field.

DefaultPolygon(model):Implementazione dell'interfaccia Polygon con una lista di segmenti. Un poligono si crea quando la coordinata start del primo segmento e' uguale alla coordinata end dell'ultimo segmento.Un poligono esiste solamente se è composto da almeno 3 segmenti.

Line(model):Implementazione dell'interfaccia Segment. Classe che rappresenta una Linea, ovvero un segmento di retta.

ALTRE CLASSI

CartesianCoordinate(model): Record che rappresenta una generica coordinata in un piano a due dimensioni. Prende una x e una y come parametri,entrambi double.

Angle(utility): Enum che rappresenta un angolo di rotazione,può essere positivo o negativo.

ColorRGB(utility): è un enum che rappresenta un insieme finito di colori : WHITE,BLACK,GREEN,RED ,YELLOW, BLUE, ORANGE;

Direction(utility): e' un enum che rappresenta un insieme finito di direzioni : FORWARD,BACKWARD; è positiva se vado avanti,negativa se vado indietro.

FileManager(utility):classe che si occupa della gestione dei file.Sono presenti solo operazioni che mi permettono di salvare su file,non di caricare da file.

Utilities(utility):Un'utility class creata per lanciare un'eccezione.

ControllerCursor(controller):Controller del cursore.Permette l'utilizzo dei metodi getter e setter del Cursor affinchè si ottenga l'effettivo spostamento del cursore in una determinata posizione e la sua rotazione.

ControllerField(controller):Controller del field.Presenta vari metodi come il metodo init che inizializza Field,controller del Cursore e il File manager, checkIfOutOfField che controlla se il cursore esce fuori dal campo, checkPolygonPresence che controlla la presenza di un poligono.

ESECUZIONE

Per l'esecuzione del programma Logo ho una classe TerminalView che mi fa da vista e al cui interno contiene un ControllerField controllerField il quale in base alle interazioni dell'utente va a manipolare i modelli.

L'app parte con TerminalView.init() .

All'interno di init() abbiamo 2 metodi fondamentali : printMenu() and readCommand().

printMenu() stampa il menù con le varie opzioni : Nuovo disegno,Carica disegno,Esci.Carica disegno non è stato implementato.In base all'opzione scelta viene chiamato uno dei 3 metodi : newDraw(),loadDraw(),System.exit

Se scegliamo di creare un nuovo disegno ci verrà chiesto di inserire l'altezza del foglio,la base,e se desideriamo usare il colore di sfondo di default bianco (e il colore del cursore sarà nero).Altrimenti sceglieremo il colore di sfondo e il colore del cursore (tutto ciò avviene con i metodi askColorField() e askCursorColor()).

Una volta che il programma è inizializzato con i parametri giusti possiamo o uscire scrivendo "exit" (il file con le informazioni verrà salvato nel file output.txt all'interno della cartella resources) o leggere la lista di tutti i comandi scrivendo "help".

L'accesso ai comandi,tra cui exit e help,è reso possibile dal metodo readCommand().Il comando repeat non è stato implementato.

ESEMPIO DI COMANDI DA INSERIRE

Posizionarsi da linea di comando nella cartella del progetto e scrivere gradle run.

Dopo aver scelto di creare un nuovo disegno,impostato base e altezza del field a 500, colore di sfondo bianco.

forward

50

left

90

forward

50

left

90

forward

50

left

90

forward

50

exit

Output generato nel file output.txt della cartella resources:

SIZE <500.0> <500.0> <255> <255> <255>

<SHAPE>

<POLYGON> <4> <0> <0> <0>

LINE <250.0> <250.0> <300.0> <250.0> <0> <0> <0> <1>

LINE <300.0> <250.0> <300.0> <300.0> <0> <0> <0> <1>

LINE <300.0> <300.0> <250.0> <300.0> <0> <0> <0> <1>

LINE <250.0> <300.0> <250.0> <250.0> <0> <0> <0> <1>

</POLYGON>

</SHAPE>