Relazione progetto Programmazione Avanzata

Lorenzo Lapucci - MAT: 114763

Responsabilità

DrawingCanvas

Consente di disegnare su un canvas linee o poligoni (con area piena).

DrawingContext

Mantiene le informazioni del disegno, come le dimensioni, la posizione del cursore, i colori e le varie **Shape** contenute nel disegno.

Position2D

Rappresenta un punto in uno spazio 2D, indicato con una coordinata **x** e una **y**.

ResourceReader / ResourceWriter

Consentono di leggere una stringa da una risorsa o scrivere una stringa su una risorsa (es. File).

Token

Rappresenta un token formato da tipo di token (**TokenType**) e valore (**String**), utilizzato per dividere una stringa in pezzi più piccoli (**Token**) riconoscibili successivamente dal **Parser**.

Tokenizer

Si occupa di prendere un pezzo di stringa in input e di restituire il **Token** corrispondente, se nella lista di **TokenType** fornita è presente un **TokenType** che riconosce quel tipo di stringa.

Parser

Presa una stringa in input, la divide in linee, e poi di nuovo in parti più piccole spezzandola su caratteri di tipo *whitespace* (spazi, tabulazioni...) e *word boundaries* (quando prima o dopo di una parola è presente un carattere non alfabetico).

Restituisce al termine una lista di **Statement** (o **SingleParseResult**, contenenti uno **Statement**) corrispondenti alle linee in input.

L'implementazione **LogoParser** utilizza internamente un **Tokenizer** per effettuare questo riconoscimento.

Processor

Esegue degli **Statement** su un determinato **DrawingContext**.

Shape

Rappresenta un qualsiasi tipo di forma che può essere disegnata su un disegno di tipo **DrawingContext**.

Color

Mantiene le informazioni di un colore nello spazio *sRGB*, supporta la trasparenza: red (0 - 255), green (0 - 255), blue (0 - 255), alpha (0 - 255).

Statement

È un qualsiasi comando che può essere eseguito su un disegno tramite un **Processor**.

Interpreter

Generico interpretatore di comandi, può eseguire i comandi uno alla volta utilizzando una coda per i comandi interni (generati da altri comandi) o passando direttamente al successivo (eseguendo tutti i comandi interni).

L'implementazione **LogoInterpreter** utilizza un **Processor** e un **DrawingContext** per eseguire gli **Statement** provenienti dal **Parser**.

(UI) DataController

Rappresenta un controller che ha bisogno di un tipo di dato **T**.

Implementazioni

DrawingCanvas

- (Test) DummyLogoDrawing
- (CLI) LoggingLogoDrawing
- (UI) FXDrawing

DrawingContext

- LogoDrawing
- (Test) DummyLogoDrawing
- (CLI) LoggingLogoDrawing
- (UI) FXDrawing

Position2D

- Point

ResourceReader

- FileResourceReader

ResourceWriter

- FileResourceWriter

Tokenizer

- LogoTokenizer

Parser

- LogoParser

Processor

- LogoProcessor

Shape

- Line
- Polygon

Color

- RGBColor

Statement

- ClearScreenStatement
- PositionStatement
    - HomeStatement
    - MoveStatement
    - RotateAngleStatement
- SetColorStatement
    - SetBackgroundColorStatement
    - SetFillColorStatement
    - SetStrokeColorStatement
- RepeatStatement
- SetDrawingStatement
- SetStrokeSizeStatement

Interpreter

- LogoInterpreter

(UI) DataController

- ChangeDrawingSizeController
- LogoViewerController
- OutputViewerController

Informazioni per la valutazione

Un input di esempio può essere copiato da **README.md**

CLI - Working directory: ./cli/

Non Interattivo

```
$ ./gradlew :cli:run --args="-i input.txt -o output.logo -s 512x5
```

Comando help

```
$ ./gradlew :cli:run --args="-h"
```

Interattivo

```
$ ./gradlew --console plain :cli:run
```

GUI

```
$ ./gradlew :ui:run
```