FASTuml - Documentazione Utente

Arici Andrea, Marchesi Gabriele, Tironi Cristian  
2023/2024

# Installazione Programma

Requisiti:

1. Java SDK 18
2. Java deve essere presente nel path delle variabili d’ambiente (per poter eseguire i comandi sul terminale)

# Esecuzione Programma

Per eseguire il programma:

1. Scaricare l’eseguibile codeGeneration.jar
2. Aprire il terminale e posizionarsi dentro nella stessa cartella in cui è stato scaricato codeGeneration.jar
3. Eseguire il comando:

java -jar codeGeneration.jar --input-file path/to/input.file

Di default verrà generato, nel path in cui ci si trova al momento dell’esecuzione del comando, solo l’immagine .png relativa al class diagram.

Sono inoltre possibili le seguenti opzioni:

**Genera solo il file Java**:

java -jar codeGeneration.jar --input-file path/to/input.file --language java

**Genera solo il file Python**:

java -jar codeGeneration.jar --input-file path/to/input.file --language python

**Genera sia file Java che python**:

java -jar codeGeneration.jar --input-file path/to/input.file --language both

**Specifica la cartella di output (solo per i file Java e Python)**:

java -jar codeGeneration.jar --input-file path/to/input.file --output-folder path/to/output

**Nomi file personalizzati (solo per i file Java e Python)**:

java -jar codeGeneration.jar --input-file path/to/input.file --java-file MyJavaFile.java --python-file MyPythonFile.py

# Documentazione della Sintassi

## Creazione delle Classi

* **ABSTRACT (facoltativo)**: Se presente, indica che la classe è astratta e non può essere istanziata.
* **CLASS**: Parola chiave che definisce una classe.
* **ID**: Nome della classe. Il nome è un identificatore che segue la convenzione di iniziare con una lettera o un carattere di sottolineatura, seguito da lettere, numeri o sottolineature.

**Esempio:**

abstract class Car {

}

## Creazione degli Attributi

Gli attributi di una classe sono dichiarati usando la regola attributeDeclarationRule, che include visibilità, tipo e nome dell'attributo, nonché la possibilità di dichiarare un attributo come readOnly.

* **Visibilità**: La visibilità dell'attributo può essere una delle seguenti:
  + public
  + private
  + protected
  + package
* **Tipo**: Il tipo dell'attributo può essere un tipo primitivo (es. int, float, boolean) o un tipo complesso definito precedentemente.
* **Nome**: Il nome dell'attributo (un identificatore).
* **Valore iniziale**: Facoltativo, l'attributo può essere inizializzato con un valore di tipo STRING, INT o FLOAT.
* **READONLY**: Facoltativo, indica che l'attributo è solo in lettura (non modificabile).

**Esempio:**

class Car {

private String model = "Tesla";

public int speed;

readOnly int year = 2020;

}

## Creazione dei Metodi

I metodi sono definiti nella regola operationDeclarationRule, che permette di specificare la visibilità, il tipo di ritorno, il nome del metodo e i parametri.

* **Visibilità**: La visibilità del metodo (public, private, protected, package).
* **Tipo di ritorno**: Tipo di ritorno del metodo (può essere un tipo di dato o void per metodi che non restituiscono nulla).
* **Nome**: Nome del metodo (identificatore).
* **Parametri**: Parametri del metodo, ciascuno con un tipo e un nome. È possibile dichiarare più parametri separati da virgola.

**Esempio:**

class Car {

public void startEngine() {

// codifica per avviare il motore

}

private int calculateFuelEfficiency(int distance, int fuel) {

return distance / fuel;

}

}

## Creazione delle Enumerazioni

Le enumerazioni sono definite tramite la regola enumDefinitionRule e consentono di dichiarare un set di valori predefiniti per una variabile.

**Sintassi:**

* **ENUM**: Parola chiave per definire un'enumerazione.
* **n=ID**: Il nome dell'enumerazione (un identificatore).
* **enumCodeRule**: Definisce i valori che appartengono all'enumerazione.

**Esempio:**

enum Days {

Monday;

Tuesday;

Wednesday;

Thursday;

Friday;

}

## Creazione delle Relazioni

Le relazioni tra classi sono dichiarate tramite la regola relationsDefinitionRule. Le relazioni possono essere di vari tipi, come Inherits, Shared, Composed e altre.

**Sintassi:**

* **nameClass1, nameClass2**: I nomi delle classi coinvolte nella relazione.
* **relationTypeRule**: Il tipo di relazione (es. Inherits, Shared, Composed).
* **multiplicityRule**: La molteplicità della relazione, che specifica il numero di istanze della classe coinvolta nella relazione.

**Tipi di Relazione (relationTypeRule):**

* **UNDREL**: Relazione non definita.
* **SXREL**: Relazione sinistra.
* **DXREL**: Relazione destra.
* **INHERITS**: Ereditarietà.
* **SHARED**: Condivisione.
* **COMPOSED**: Composizione.

**Esempio:**

relations {

ClasseB 1,1 inherits ClasseA 1,3;

}