

FASTuml - Documentazione Utente

Arici Andrea, Marchesi Gabriele, Tironi Cristian
2023/2024

Sommario

Installazione Programma..... 4

Esecuzione Programma..... 6

Documentazione della Sintassi 7

 Creazione delle Classi 7

 Creazione degli Attributi 8

 Creazione dei Metodi..... 9

 Creazione delle Enumerazioni..... 10

 Creazione delle Relazioni..... 11

Introduzione

FASTuml è un programma in grado di generare un diagramma delle classi ed una relativa struttura di partenza del codice implementativo utilizzando una sola sintassi. Questo evita la duplicazione nella scrittura del codice, che può portare a perdite di tempo, errori o mancata creazione di documentazione. FASTuml è in grado di generare del codice di partenza nei linguaggi java e python. Queste strutture iniziali saranno perfettamente identiche al diagramma delle classi, anch'esso generato da FASTuml sotto forma di .png.

Il programma necessita una singola riga di comando nel prompt per il funzionamento, attraverso cui si possono gestire diverse impostazioni del funzionamento

FASTuml non richiede l'uso di internet.

Installazione Programma e Requisiti

Sistema Operativo:

- Windows Server 2008 R2 SP1 (64 bit) e versioni successive
- Windows Vista SP2 e versioni successive
- Mac OS X 10.8.3+
- Ubuntu Linux 5.5+
- Oracle Solaris 10 Update 9+ (64 bit)

In tutti i casi sono richiesti i privilegi di amministratore

RAM:

- 128 MB

Spazio di archiviazione richiesto:

- 22 MB

Requisiti Java:

1. Java JDK (utilizzare la versione più recente possibile per evitare conflitti)
2. Java deve essere presente nel path delle variabili d'ambiente (per poter eseguire i comandi sul terminale). Vedi sotto in caso non sia già inserito nelle variabili d'ambiente.

Scaricare l'eseguibile codeGeneration.jar e posizionarlo dentro la cartella in cui lo si vuole utilizzare.

Inserimento Java nelle variabili d'ambiente

Per impostare JAVA_HOME su un sistema Windows, eseguire queste azioni.

1. Sul pannello di controllo:
 - Fare clic su **Sistema**.
 - Fare clic su **Impostazioni di sistema avanzate**.
Viene visualizzata la finestra Proprietà di sistema.
2. Fare clic sul pulsante **Variabili di ambiente**.
3. Fare clic sul pulsante *Nuovo* nella sezione delle variabili di sistema.
4. Aggiungere il nome della variabile JAVA_HOME e specificare un percorso alla directory jre . Ad esempio: C:\wlp_16002\IBM\WebSphere\Liberty\java\java_1.8_64\jre
Alcuni comandi di Collective Controller richiedono che il percorso della directory jre\bin di installazione Java sia disponibile nel percorso di sistema, quindi aggiungere anche un percorso alla directory jre\bin .

5. Salvare le modifiche. Potrebbe essere necessario riavviare il computer per rendere effettive le modifiche.
6. Per verificare le modifiche, da una riga comandi immettere `set JAVA_HOME`. Il comando visualizza le impostazioni `JAVA_HOME` :
`JAVA_HOME=C:\wlp_16002\IBM\WebSphere\Liberty\java\java_1.8_64\jre.`

Esecuzione Programma

Per eseguire il programma:

1. Aprire il terminale e posizionarsi dentro nella stessa cartella in cui è stato scaricato codeGeneration.jar (su terminale Windows scrivere da linea di comando "cd <nome_cartella>")
2. 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. (Vedi subito sotto per tutte le opzioni)

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

In questa sezione verrà descritto il linguaggio di FastUml e come utilizzarlo. All'inizio di ogni paragrafo è presente la sintassi per creare la relativa istanza. All'interno della regola sintattica vengono utilizzate:

- le parentesi quadre “[]” per indicare che il contenuto al loro interno può anche non essere presente all'interno della regola.
- Il simbolo di asterisco “*” dopo del parentesi quadre per indicare che è possibile ripetere più di una volta ciò che è al suo interno

Creazione delle Classi

Sintassi:

```
[abstract] class nomeClasse {  
}
```

- **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

Esempio:

```
abstract class Car {  
}
```

```
Class Studente{  
}
```

Creazione degli Attributi

- **Visibilità:** La visibilità dell'attributo può essere una delle seguenti

Visibilità	Simbolo / Parola chiave	Disegno
public	public	+
protected	protected	#
private	private	-
package private	package	None

- **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.
- **Valore iniziale:** Facoltativo, l'attributo può essere inizializzato con un valore che deve essere coerente al suo Tipo.
- **READONLY:** Facoltativo, indica che l'attributo è solo in lettura.

Esempio:

```
class Car {  
    private String model = "Tesla";  
    public int speed;  
    readOnly int year = 2020;  
}
```


Creazione dei Metodi

Sintassi:

```
operation {  
    visibilità TipoDiRitorno Nome( [TipoParametro nomeParametro]* );  
}
```

- **Visibilità:** La visibilità del metodo (public, private, protected, package). Vedi “Creazione degli Attributi” per i possibili tipi.
- **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
- **Parametri:** Parametri del metodo, ciascuno con un tipo e un nome. È possibile dichiarare più parametri separati da virgola.

Esempio:

```
class Car {  
    operation {  
        public void methodA();  
        protected int methodB(int x, String y);  
    }  
}
```

Creazione delle Enumerazioni

Sintassi:

```
enum Nome {  
    [Valore Enum];  
}
```

- **enum:** Parola chiave per definire un'enumerazione.
- **Nome:** Il nome dell'enumerazione.
- **Valore Enum:** Definisce i valori che appartengono all'enumerazione.

Esempio:







```
enum Days {  
    Monday;  
    Tuesday;  
    Wednesday;  
    Thursday;  
    Friday;  
}
```

Creazione delle Relazioni

Sintassi:

```
relations {  
    nomeClasse1 molteplicita1 {tipoRelazione} nameClass2 molteplicita2;  
}
```

- **nameClass1, nameClass2:** I nomi delle classi coinvolte nella relazione.
- **tipoRelazione:** Il tipo di relazione

Tipo	Simbolo	Disegno
Relazione non definita	-	
Relazione sinistra	<	
Relazione destra	>	
Ereditarietà	inherits	
Composizione	composed	
Condivisione	shared	

- **molteplicità:** La molteplicità della relazione, che specifica il numero di istanze della classe coinvolta nella relazione.

Esempio:

```
relations {  
    ClasseA 1,1 inherits ClasseB 1,3;  
    ClasseC 1,* - ClasseD 1,*;  
    ClasseE 0,1 < ClasseF 1,3;  
    ClasseG 1,1 > ClasseH 1,3;  
    ClasseE 0,* > ClasseF 1,*;  
}
```

Errori e Warning

In caso di errori di sintassi o di semantica verranno evidenziati tramite un messaggio sul prompt al momento dell'esecuzione del comando iniziale. Il messaggio contiene una breve spiegazione dell'errore e anche la posizione di riga e colonna del file di input in cui si è verificato.

In caso di errore di sintassi il programma indicherà quale parola non è stata riconosciuta e cosa invece si sarebbe aspettato di trovare.

In caso di errore, non verrà generata nessuna visualizzazione e nessuna struttura codice.