



WShabtiPict - Guida Utente

Introduzione

WShabtiPict è un semplice tool per:

1. Generare combinazioni di input per i test case dati un insieme di parametri e vincoli
2. Esplicitare gli oracoli per l'insieme delle combinazioni generate
3. Generare classi di test compatibili con JUnit 5

WShabtiPict offre due diverse modalità di utilizzo: interattiva e automatica.

In modalità **interattiva**, il tool guida l'utente nelle fasi di:

- Creazione del modello da testare (parametri, vincoli e grado delle combinazioni)
- Definizione dell'oracolo (umano)
- Creazione della classe di test compatibile con JUnit 5

In modalità **automatica**, il tool offre la possibilità di fornire all'eseguibile dei parametri di ingresso e di accedere singolarmente alle fasi di:

- Creazione di casi di test a partire da un modello pre-esistente fornito tramite file
- Creazione guidata dell'oracolo a partire da casi di test pre-generati forniti tramite file
- Creazione della classe di test a partire dai casi di test completi di oracolo forniti tramite file



Building

Building su Linux, OS/X, *BSD, etc.

In Linux e sistemi Unix-like (come OS/X) la compilazione va eseguita tramite `clang++` o `g++`.

Su Linux è necessario l'utilizzo della libreria `libstdc++` offerta da gcc 5.

Tramite il comando *make* si può effettuare la compilazione automatica.

Building su Windows

Ci sono due possibilità su Windows per compilare ed eseguire il progetto:

- Installare [cygwin](#) con gli add-ons *make* e *build-essential*, ciò permette di ottenere su Windows un sottosistema *Linux*. A questo punto si potrà procedere come su *Linux* avviando la compilazione tramite il comando `make`
- Utilizzare *WSL (Windows Subsystem for Linux)* per eseguire in ambiente *GNU/Linux* senza installare *cygwin*. Per compilare il progetto è necessario soddisfare le seguenti dipendenze che vanno installate manualmente, ad esempio su derivate di *Debian* (come [Ubuntu](#)) i comandi sono:

```
sudo apt install build-essential
sudo apt install make
```



Utilizzo

Esempio (modalità interattiva)

```
./WShabtiPict
SourceCode Path: ClassExample/Calcolatrice.java

-----METHOD: public int somma(int a, int b)-----
#Do you want skip this method? (y/n)
Please specify your choice: n

#Do you want specify input file source? (y/n)
Please specify your choice: n

Please specify, separated by a comma, all possible values of "a"
For a range of numbers you can use the syntax "start:stop:step": 1:5:1,7

Please specify, separated by a comma, all possible values of "b"
For a range of numbers you can use the syntax "start:stop:step": 3,4

#Do you want specify Exclusive Constraints? (y/n)
Please specify your choice: y

To specify a combination to exclude, use the syntax "value1,value2,...".
Use "*" if you don't care about a parameter.

Please specify a combination to exclude or "stop" to terminate: 1,3

Please specify a combination to exclude or "stop" to terminate: *,4

Please specify a combination to exclude or "stop" to terminate: stop

Please specify the order of combinations for "somma": 2

-----ORACLE: public int somma(int a, int b)-----
Please specify the oracle for the following possible combinations
For a range of numbers you can use the syntax "expected_value:delta"
somma(2,3): 5
somma(4,3): 7
somma(3,3): 6
somma(5,3): 8
somma(7,3): 10
```



```
-----METHOD: public float divisione (int a, int b)-----
#Do you want skip this method? (y/n)
Please specify your choice: n

#Do you want specify input file source? (y/n)
Please specify your choice: y
Please specify the parameter file path: divisione_input.txt

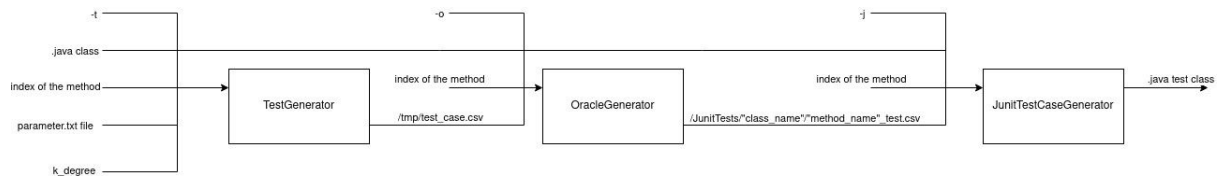
Please specify the order of combinations for "divisione": 2

-----ORACLE: public float divisione (int a, int b)-----
Please specify the oracle for the following possible combinations
For a range of numbers you can use the syntax "expected_value:delta"
divisione(10,1): 10
divisione(20,3): 6:1
divisione(10,3): 3.3:0.2
divisione(5,2): 2.5
divisione(5,3): 1:1
divisione(5,1): 5
divisione(20,1): 20

Your tests have been generated, please check JUnitTests/Calcolatrice/
WARNING: Default constructor is used for class Calcolatrice. If needed,
complete it in the code.
```



Esempio (modalità automatica)



Mode: TestGenerator [-t]

```
./WShabtiPict -t <INPUT_JAVA_CLASS> <INDEX_OF_METHOD_TO_TEST> <INPUT_VALUES_FILE> <K_DEGREE>
./WShabtiPict -t ClassExample/Calcolatrice.java 2 divisione_input.txt 2
```

INPUT_VALUES_FILE

```
parameter_name1:value_p1_1,value_p1_2,value_p1_3,...
parameter_name2:value_p2_start:value_p2_end:step...
...
#CONSTRAINTS
value_p1_x,value_p2_x,value_p3_x,...
value_p1_y,value_p2_y,value_p3_y,...
...
```

Esempio di INPUT_VALUE_FILE

```
a:10,5
b:2:3:1
#CONSTRAINTS
10,2
```



Mode: OracleGenerator [-o]

Questo passo può essere evitato scrivendo manualmente gli oracoli nel file `.csv` generato dal TestGenerator

```
./WShabtiPict -o <INPUT_JAVA_CLASS> <INDEX_OF_METHOD_TESTED> <TEST_INPUT_FILE>  
./WShabtiPict -o ClassExample/Calcolatrice.java 2 ./tmp/test_case.csv
```

TEST_INPUT_FILE

```
value_p1,value_p2,...  
...
```

Esempio di *TEST_INPUT_FILE*

```
10,3  
5,2  
5,3
```



Mode: JUnitTestCaseGenerator [-j]

```
./WShabtiPict -j <INPUT_JAVA_CLASS> <INDEX_OF_METHOD_TESTED> <TEST_WITH_ORACLES_INPUT_FILE>  
./WShabtiPict -j ClassExample/Calcolatrice.java 2 JUnitTests/Calcolatrice/divisione_test.csv
```

TEST_WITH_ORACLES_INPUT_FILE

```
value_p1,value_p2,expected_value  
...
```

Esempio di *TEST_WITH_ORACLES_INPUT_FILE*

```
10,3,3.3  
5,2,2.5  
5,3,1.6
```

TEST_WITH_ORACLES_INPUT_FILE (se almeno un intervallo necessita di un delta)

```
value_p1,value_p2,expected_value,delta  
...
```

Esempio di *TEST_WITH_ORACLES_INPUT_FILE*

```
10,3,3.3,0.1  
5,2,2.5,0 #se c'è almeno un delta, i test esatti dovranno avere delta 0  
5,3,1.6,0.1
```

Sintassi dei file

INPUT_VALUES_FILE

Il file che specifica i possibili valori di un parametro, rispetta la seguente sintassi:

```
<nome_parametro1>:<valore_1>,<valore_2>,<valore_3>,...  
<nome_parametro2>:<valore_1>,<valore_2>,<valore_3>,...  
...  
#CONSTRAINTS  
<valore_p1_x>,<valore_p2_x>,<valore_p3_x>,...  
...
```

Per specificare dei vincoli esclusivi, è sufficiente aggiungere in seguito alla specifica di tutti i parametri, una riga separatrice che inizia con # (*è possibile aggiungere qualunque commento a seguito del cancelletto*).

Dalla riga successiva a quella che inizia con il cancelletto, ciascuna indicherà una combinazione da escludere.

Tramite l'utilizzo di un * si può indicare che il parametro a quella posizione può assumere qualunque valore (è quindi possibile specificare più esclusioni in una sola riga)

Ad esempio:

```
filesystem:ext4,ntfs,fat32  
os:windows,linux  
dimensione_cluster:512,1024,2048,4096  
#CONSTRAINTS  
ext4,windows,*
```

permette di escludere la combinazione ext4,windows ovvero qualunque test che contiene ext4 e windows, a prescindere dal valore del parametro dimensione_cluster.

Nel caso si volesse utilizzare un intervallo di valori continuo, è possibile utilizzare la sintassi:

```
<nome_parametro>:<start>:<stop>:<step>,...
```




Dove start indica il primo valore assunto, stop è l'ultimo valore - incluso - e step specifica l'avanzamento. *E' possibile combinare entrambe le notazioni*

Ad esempio:

$1 : 6 : 5$ viene espanso in $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

$1, 2, 5 : 14 : 2$ viene espanso in $\{1, 2, 5, 7, 9, 11, 13\}$

TEST_INPUT_FILE

Il file contenente i casi di test generati rispetta la seguente sintassi:

```
<valore_p1>,<valore_p2>,...
```

Ogni test generato occupa una riga del file, ciascuna riga è composta dai valori di ogni parametro.

TEST_WITH_ORACLES_INPUT_FILE

Il file contenente i casi di test con oracolo presenta la seguente sintassi

```
<valore_p1>,<valore_p2>,...,valore_atteso,delta
```

Il valore delta risulta necessario solo quando almeno uno dei valori attesi deve essere contenuto all'interno di un intervallo (estremi inclusi). In questo caso sarà necessario aggiungere un delta nullo "0" accanto i rimanenti valori attesi esatti.

Ad esempio:

$10, 3, 3.3, 0.1$

$5, 2, 2.5, 0$ {il valore atteso esatto 2.5 viene rinchiuso in un delta ,0}

$5, 3, 1.6, 0.1$



Limitazioni

WShabtiPict, al momento, supporta solo i tipi standard di Java (incluse le classi Wrapper) e solo metodi che ritornano uno scalare.



Licenza

MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.