

# Uso dei tool JML

**ANGELO GARGANTINI**

## Indice

1. COME INSTALLARE JML .....	3
1.1 Scaricare i file necessari .....	3
1.2 Scompattare il file .....	3
2. COME USARE I TOOL DI JML .....	4
2.1 Controllo della sintassi .....	5
2.2 Compilazione dei file JML .....	6
2.3 Esecuzione dei file compilati con jmlc .....	6
2.3.1 Eseguire file JML che non rispettino il contratto.....	7

## 1. Come Installare JML

Vediamo come installare i tool di JML.

### 1.1 Scaricare i file necessari

Per prima cosa devi scaricare il seguente file:

JML.5.4.tar.gz

Dal sito: <http://sourceforge.net/projects/jmlspecs/>

O più direttamente:

[http://sourceforge.net/project/showfiles.php?group\\_id=65346](http://sourceforge.net/project/showfiles.php?group_id=65346)

Nota che le versioni precedenti di JML non funzionano con Java 5, mentre questa (e le successive) funzionano anche con Java 5 (anche se non leggono sorgente per Java 5 – non potete usare i generics o gli enumeration, per esempio).

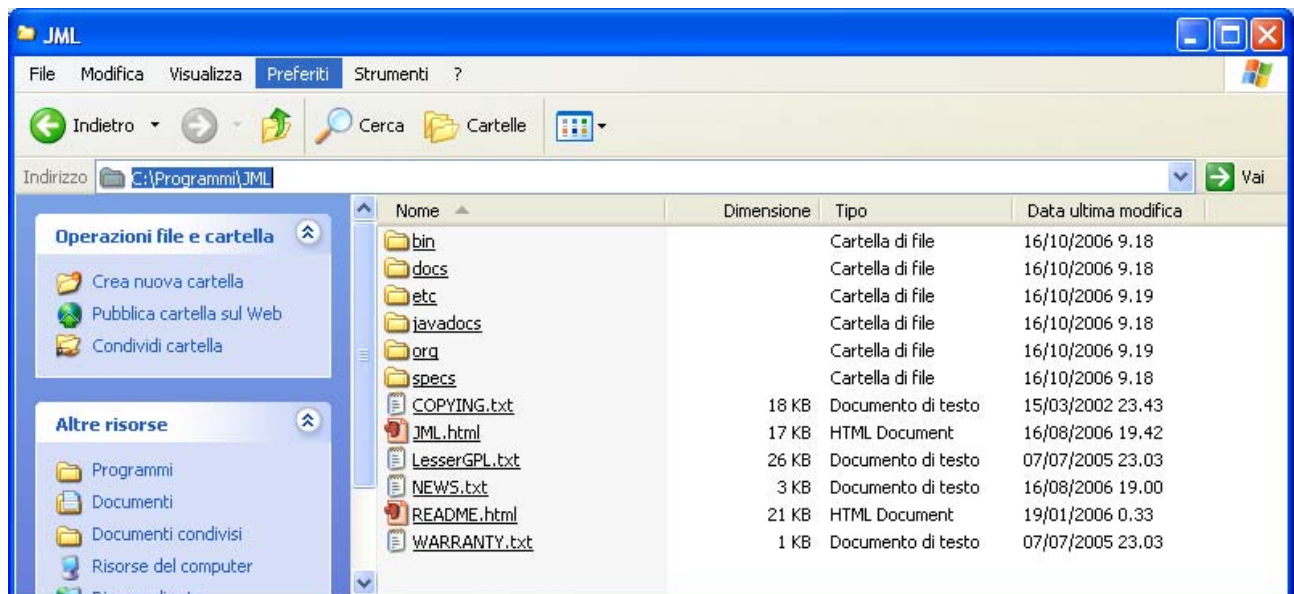
Nota anche che esiste un plugin di JML per eclipse, ma l'installazione richiede diverse modifiche e trucchi vari: se volete provare ad installarlo domandate al docente le istruzioni.

### 1.2 Scompattare il file

Scompatta il file (usando winzip, winrar o semplicemente un gunzip), in un directory di tua scelta (possibilmente con nome senza spazi). Ad esempio in

C:\Programmi

La cartella C:\Programmi\JML a questo punto dovrebbe apparire così:



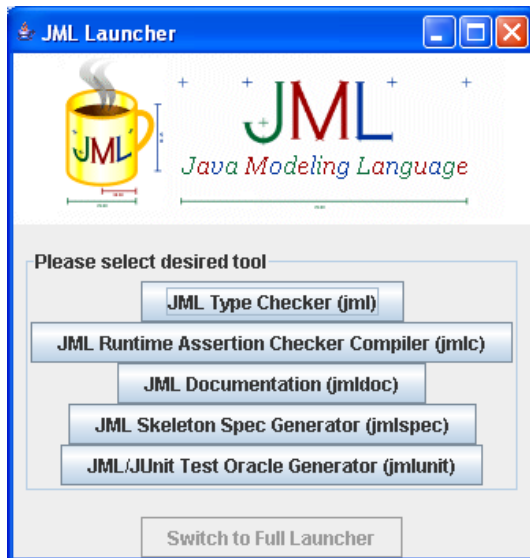
## 2. Come usare i tool di JML

Vediamo ora come eseguire i tools di JML in formato grafico.

Apri la directory C:\Programmi\JML\bin.

Fai doppio click su jml-release.jar (oppure in riga di comando: `java -jar jml-release.jar`). Nota che devi avere jdk installato correttamente.

Apparirà la seguente finestra che è il dialogo principale di JML:



Scriviamo un piccolo programma JML composto da due classi.

La prima è Persona.java, che rappresenta una persona. Ha un metodo setAltezza che può prendere solo un numero  $> 0$  e questo è quanto previsto dal contratto.

```
public class Persona {
    public float altezza, peso;

    public float getAltezza() {
        return altezza;
    }
    //@ requires altezza > 0;
    public void setAltezza(float altezza) {
        this.altezza = altezza;
    }
}
```

```
public float getPeso() {
    return peso;
}
public void setPeso(float peso) {
    this.peso = peso;
}

public float getIndiceDiMassa(){
    float indice = peso /((altezza/100) * (altezza/100) );
    return indice;
}
}
```

Aggiungiamo una classe Main.java che contiene il programma vero e proprio:

```
public class Main {
    public static void main(String[] args) {
        Persona p = new Persona();
        p.setAltezza(170);
        p.setPeso(70);
        System.out.println(p.getIndiceDiMassa());
    }
}
```

Salviamo le due classi in due file in una directory, ad esempio sul desktop nella directory JML\_ex.

## 2.1 Controllo della sintassi

Dalla finestra principale di JML, clicca su JML type checker (jml). Appare un dialogo in cui puoi aggiungere i due file tra i file da controllare e lancia jml con l'icona a freccia.

La prima volta che si avvia il JML type checker appare un messaggio di errore che dice che non è stata trovata la directory contenete le specs (che è C:\Programmi\JML\specs). Clicca Browse fino a fargli prendere quella directory.

Appare una finestra di output in cui fa vedere se ci sono errori. Se tutto va bene, finisce con Done.

## 2.2 Compilazione dei file JML

Dalla finestra principale di JML, clicca su JML runtime assertion checker compiler (jmlc). Appare un dialogo in cui ancora devi selezionare i due file e lanciare il compilatore jmlc. Appare una finestra di output in cui JML mostra i file che sta controllando e alla fine se dice Done ha prodotto i due .class, che sono il bytecode desiderato

## 2.3 Esecuzione dei file compilati con jmlc

Per eseguire i class compilati con JML devi avere nel classpath (dove java cerca le librerie) JML/bin/jmlruntime.jar (nel class path di boot) e JML/bin/models.jar o JML/bin/jmlmodelsnonrac.jar.

C'è uno script che imposta in modo corretto il classpath e si chiama jmlrac che si trova sotto C:\Programmi\JML\bin. Usa in generale jmlrac al posto di java come segue:

```
jmlrac my.package.name.ClassnameOfMain
```

dove ClassnameOfMain è il programma principale.

Per far vedere a Windows lo script jmlrac ci sono diverse soluzioni. Noi scegliamo quella di includere la cartella C:\Programmi\JML\bin nel nostro PATH. Per far questo dobbiamo fare aggiornare il PATH di windows. Dal Pannello di controllo-> Sistema -> (tab) Avanzate -> (tasto) variabili d'ambiente. In variabili di sistema, dobbiamo modificare la variabile PATH e aggiungere ";c:\Programmi\JML\bin" (nota che ogni cartella del PATH è separata da ";").

Dobbiamo poi modificare il file jmlenv.bat che si trova sotto C:\Programmi\JML\bin in modo che la variabile JMLDIR punti alla directory dove abbiamo messo JML. Nel nostro caso:

```
set JMLDIR=c:\Programmi\JML
```

A questo punto apriamo una finestra di comando dos (start -> accessori -> prompt dei comandi) e se scriviamo:

```
jmlc -version
```

deve apparire un messaggio che dice la versione di JML che stiamo usando:

```
C:\Documents and Settings\garganti> jmlc --version
usage: org.jmlspecs.jmlrac.Main [option]* [--help] <jml-files>
Version: JML tools release 5.4_rc1 (August 16, 2006)
error: No input files given
```

A questo punto possiamo eseguire il nostro Main. Facciamo cd fino alla directory che contiene il main e poi lanciamo:

```
jmlrac Main
```

Dovremmo avere un output simile.

```
C:\Documents and Settings\garganti\Desktop\JML_ex>jmlrac Main
24.221453
```

Nota che a questo punto possiamo usare anche il compilatore in riga di comando, chiamando ad esempio:

```
jmlc Main.java
```

Attenzione: se jml-rac non trova il .class che vuoi eseguire, può essere che sia perché java non trova il .class. In questo caso puoi usare l'opzione `-cp` (`- class path`) per dirgli esattamente dove si trova il tuo .class.

Esempio: `jml-rac -cp . Persona`

### 2.3.1 Eseguire file JML che non rispettino il contratto.

Se modifichiamo il main in modo di settare l'altezza ad un numero negativo:

```
p.setAltezza(-10);
```

Ricompiliamo il Main (usando la finestra di dialogo di JML o con il comando `jmlc Main` nella finestra dos) e eseguiamo nuovamente il Main otteniamo:

```
C:\Documents and Settings\garganti\Desktop\JML_ex>jmlrac Main
Exception in thread "main"
org.jmlspecs.jmlrac.runtime.JMLInternalPreconditionError:
by method Persona.setAltezza regarding specifications at
File "Persona.java", line 8, character 31
when 'altezza' is -10.0F at Main.main(Main.java:320)
```

## 2.4 Esercizio

Per ogni esercizio compila ed esegui in JML (con `jmlc` e `jmlrac` se è richiesta l'esecuzione)

### 2.4.1 Programma base

Scrivi un programmino Java con un metodo (ad esempio statico) che calcola il resto della divisione intera tra due numeri interi  $> 0$  (non scrivere per ora l'implementazione, scrivi solo l'intestazione, restituisci zero)

Controlla la sintassi e compila.

### 2.4.2 Contratto

Aggiungi le precondizioni e le post condizioni (scegli tu precondizioni e postcondizioni che ti sembrano ragionevoli). Se riesci non usare nelle postcondizioni l'operatore di modulo.

Controlla la sintassi e compila.

### **2.4.3 Main**

Aggiungi un main e prova a richiamare il metodo (nel main) con dati validi (es 8,4).

Controlla la sintassi, compila ed esegui.

### **2.4.4 Violazioni**

Prova con altri dati (validi) ma che non hanno resto 0. Controlla la sintassi, compila ed esegui.

Vedi che le postcondizioni non sono rispettate? (il metodo non è ancora implementato)

### **2.4.5 Implementazioni**

Aggiungi l'implementazione del metodo riesegui e vedi che stavolta funziona.

### **2.4.6 Violazioni precondizioni**

Richiamalo dal main stavolta con le precondizioni non valide. Cosa osservi?



## Soluzione

### 3. Programma “resto” in JML

Una possibile soluzione segue:

1. Crea il file Prova.java così:

```
public class Prova {  
  
    static public int resto(int a, int b){  
        return 0;  
    }  
}
```

Compila con jmlc: non ci sono errori.

2. Possibile preconditioni:  $a > 0$  e  $b > 0$

postcondizione:  $\text{\result} + a/b * b = a$  (il resto + il quoziente intero \* il divisore = dividendo)

Il file diventa con la sintassi JML:

```
public class Prova {  
  
    //@ requires a > 0 && b > 0;  
    //@ ensures \result + a/b * b == a ;  
    static public int resto(int a, int b){  
        return 0;  
    }  
}
```

che passa JMLC.

### 3 .Aggiungo il main:

```
static public void main(String[] args){  
  
    System.out.println("resto di 8 div 4 = " + resto(8,4));  
  
}
```

ed eseguo con jmlrac e jmlc osservando l'output e nessun errore.

### 4. Modifico ora il main:

```
static public void main(String[] args){  
  
    System.out.println("resto di 8 div 4 = " + resto(8,4));  
    System.out.println("resto di 9 div 4 = " + resto(9,4));  
  
}
```

Stavolta eseguo ed osservo la violazione della postcondizione:

```
C:\Documents ...>jmlrac Prova
```

```
resto di 8 div 4 = 0
```

```
Exception in thread "main" org.jmlspecs.jmlrac.runtime.JMLInternalNormalPostcond  
itionError: by method Prova.resto regarding specifications at  
File "Prova.java", line 5, character 31 when
```

```
'b' is 4
```

```
'a' is 9
```

```
\result' is 0
```

```
at Prova.resto(Prova.java:554)
```

```
at Prova.internal$main(Prova.java:15)
```

at Prova.main(Prova.java:722)

5. Modifico Prova, implementando il metodo – uso questa volta %:

```
//@ requires a > 0 && b > 0;  
  
//@ ensures \result + a/b * b == a ;  
static public int resto(int a, int b){  
    return a%b;  
}
```

Eseguo e tutto va bene:

```
C:\...\JML_ex>jmlrac Prova
```

resto di 8 div 4 = 0

resto di 9 div 4 = 1

6. Se provo con le precodizioni non valide nel main:

```
static public void main(String[] args){  
  
    System.out.println("resto di 8 div 4 = " + resto(8,4));  
    System.out.println("resto di 9 div 4 = " + resto(9,4));  
    System.out.println("resto di -19 div 4 = " + resto(-19,4));  
  
}
```

Compilo ed eseguo, ottengo:

```
C:\Documents and Settings\gargantini.RICCOBEN-CF6958\Desktop\JML_ex>jmlrac  
Prova
```

resto di 8 div 4 = 0

resto di 9 div 4 = 1

Exception in thread "main" org.jmlspecs.jmlrac.runtime.JMLInternalPreconditionEr

ror: by method Prova.resto regarding specifications at

File "Prova.java", line 4, character 25 when

'b' is 4

'a' is -19

at Prova.main(Prova.java:733)