JML

Angelo Gargantini
Info 3 2006

Introduzione a JML

- è un insieme di tool e di tecniche per il design by contract in Java ed altro
- disponibile su web:
- http://www.cs.iastate.edu/~leavens/JML/
- www.jmlspecs.org

JML sintassi

- Per rendere JML facile da usare le annotazioni JML vengono aggiunte come commenti nel file .java tra
  /*@ . . . @*/, o dopo //@.
- le proprietà sono scritte come espressioni boolean di java con alcuni operatori in più:
  \result, \forall, \old, ==>, ...
- e alcune parole chiave
  requires, ensures, invariant, ...

Pre- and postconditions

Pre- and post-conditions for methods:

```
/*@ requires amount >= 0;
      ensures balance ==
                              \old(balance) - amount
      && \result == balance;
@*/
public int withdraw(int amount) {
...
}
```

Here \old(balance) refers to the value of balance before execution of the method.

Altri esempi

JML specs can be as strong or as weak as you want.

```
/*@ requires amount >= 0;
      ensures true;
@*/
public int debit(int amount){
...
}
```

This default postcondition “ensures true” can be omitted.

Esempio con <==>

Posso usare l'operatore <==> : se e solo se

```
//@ ensures \result <==> j < n;

public  boolean minore(int j, int n) {
    return j < n;
}
```

non_null

Many invariants, pre- and postconditions are about references not being null. non_null is a convenient short-hand for these.

```
public class Directory {
private /*@ non null @*/ File[] files;
void createSubdir(/*@ non null @*/ String name){…}
Directory /*@ non null @*/ getParent(){…}
```

assert

An assert clause specifies a property that should hold at some point in the code, eg.

```
if (i <= 0 || j < 0) {
...
} else if (j < 5){
//@ assert i > 0 && 0 < j && j < 5;
...
} else {
//@ assert i > 0 && j > 5;
...
}
```

assert

- JML keyword assert now also in Java (since Java 1.4).
- Still, assert in JML is more expressive, for example in

```
...
for (n = 0; n < a.length; n++)
if (a[n]==null) break;
/*@ assert
(\forall int i; 0 <= i && i < n; a[i] != null);
@*/
```

Quantifiers

- JML supports several forms of quantifiers
  - Universal and existential (\forall and \exists)
  - General quantifiers (\sum, \product, \min, \max)
  - Numeric quantifier (\num_of)
- Esempi

```
(\forall Student s; juniors.contains(s);
                                    s.getAdvisor() != null)
(\forall Student s; juniors.contains(s) ==>
   s.getAdvisor() != null)
```

Esempio quantificatori

```
/** Exercise 1: find the minimum element in an
    array. */
/*@   requires a != null && a.length >= 1;
  @   ensures (\forall int i; 0 <= i &&
                        i <  a.length; \result <= a[i])
@   && (\exists int i; 0 <= i && i < a.length;
                                  \result == a[i]);
@*/
    public static int  find_min (int a[])
```

Benefici di JML

- JML specifications provide explicit documentation of contracts and invariants
- Writing JML specs for code, you make explicit assumptions and considerations that have gone into the design of code
- Such JML specifications make it easier to understand code and should help to convince yourself and others that nothing can go wrong.
- Such JML specifications can be used by tools .

Tools per JML

1. JML compiler (jmlc): compila file java per avere class instrumentati
2. JML/Java interpreter (jmlrac):
   performs checks for all JML assertions at runtime: any assertion violation results in a special exception.
3. Extended static checking with escjava2: proves JML assertions at compile time
4. JML/JUnit unit test tool (jmlunit)
5. HTML generator (jmldoc)
6. Altri tool ....

Compilatore JML (jmlc)

- Uso testuale

  $ jmlc Person.java

  produces Person.class

  $ jmlc –Q *.java

  produces *.class, quietly

  $ jmlc –d ../bin Person.java

  produces ../bin/Person.class

Eseguire il codice compilato con jmlc

- Must have JML's runtime classes (jmlruntime.jar) in Java's boot class path
- Automatic if you use script jmlrac, e.g.,

  $ jmlrac PersonMain

Come installare JML

- Scarica jmlXXX.tar.gz dal sito di JML
  - Oppure jml.tar.gz dal sito web del corso sottodirectory codice
  - Oppure da titano
- Unzippalo su c:\ in modo di avere una directory JML che contenga tutto
  - c:\jml -> tutto le sotto directory di jml
  - c:\jml\
  - c:\jml\bin -> programmi veri e propri

Come eseguire JML

- Scrivi la tua classe (esempio file My.java) nella directory c:\JML\bin
- Compilala con jmlc -> prossimo lucido
- Esegui il My.class nella finestra dos, fai cd fino a c:\JML\bin ed esegui

```
jmlrac My
```

Compilare jml

- Testualmente
- Con interfaccia grafica:
  - Doppio click su c:\JML\bin\jml-release.jar
    - Seleziona la directory c:\JML\specs che contiene le specifiche jml delle librerie
    - Seleziona jml checker se vuoi controllare la sintassi
    - Seleziona jmlc se vuoi utilizzare il jml compiler per produrre il file .class

Esercizio

- Fai tu il metodo findMin

Esercizio JML/JUnit

- Scrivi l'intestazione di un metodo che calcola il resto della divisione intera tra due numeri interi (non scrivere per ora l'implementazione)
- Junit: Scrivi dei casi di test con Junit
  - Esegui i test e nota che alcuni (se non tutti) i test falliranno
- JML: Scrivi le precondizioni e le post condizioni
- Prova a richiamare il metodo (nel main)
  - Con le precondizioni non valide
  - Compila ed esegui in JML (con jmlc e jmlrac)

- Modifica ora la classe
  - Con le precondizioni valide (ma le post condizioni non valide, visto che non hai ancora implementato il metodo
  - Esegui il codice in JML: cosa osservi?
- Implementa il metodo in modo corretto
  - esegui i casi di test con Junit
  - Esegui i casi di test con JML

Altri esempi

- Divisione intera tra due numeri
- Algoritmo che dato un array V di numeri restituisce un numero in V che non sia il massimo
- Ordinamento di un array di stringhe
  - Nota che la post condizione non è semplice da scrivere,
- Ordinamento di un array di interi
- Dato una stringa, restitusce la stessa stringa ma con la prima lettera maiuscola.