Java Modeling Language

Viliam Holub
PEL & SRG
http://www.ucd.ie

UNIVERSITY COLLEGE DUBLIN
School of Computer Science
Java Modeling Language

http://www.cs.ucf.edu/~leavens/JML/

Behavioral interface specification language for Java
Follows the design by contract paradigm
Using pre- and postconditions and invariants
Annotations in the Java program as comments
- Can be compiled with any Java compiler

jmlc
assertion generator
ESC/Java
static checker
Annotations in comments

Begins with @, ends with ;

//@ specification
/*@ specification */

Follows Java syntax with extensions
Basic keywords

requires
ensures
invariant

signals
assignable
pure
also
assert

\result
\old(name)
\forall
\exists
a \implies b
a \iff b
a \iff b
+ java logical operators
```java
String getFullName( String first, String last) {
    return first +" " +last;
}
```

What does the method require?
Example 1/2

```java
//@ requires first != null;
//@ requires last != null;
//@ requires first.length() > 0;
//@ requires last.length() > 0;
String getFullName( String first, String last)
{
    return first +" " +last;
}
```

What does the method ensure?
//@ requires first != null;
//@ requires last != null;
//@ requires first.length > 0;
//@ requires last.length > 0;
//@ ensures \result != null;
String getFullName( String first, String last) {
    return first +" " +last;
}

Is there a better explanation of the result?
Example 1/4

//@ requires first != null;
//@ requires last != null;
//@ requires first.length > 0;
//@ requires last.length > 0;
//@ ensures \result != null && (* \result is a full name of the person *);

String getFullName( String first, String last)
{
    return first +" " +last;
}
public double sqrt( double x) 
{
    return Math.sqrt( x);
}

What does the method require and ensure?
//@ public ghost static final double tolerance = 0.001;
//@ requires x >= 0.0;
//@ ensures (\result*\result-x < tolerance) && (\result*\result-x > -tolerance);
public static double sqrt( double x)
{
    return Math.sqrt( x);
}
class Date
{
    int day;
    int hour;

    public Date( int d, int h) {...}
    public void setDay( int d) {...}
    public void setHour( int h) {...}
    public int getDay() {return day;}
}

What do the methods require and ensure?
Example 3/2

class Date
{
    int day;
    int hour;

    //@requires 1<=d && d <=31;
    //@requires 0<=h && h < 24;
    public Date( int d, int h) {...}
    //@requires 1<=d && d <=31;
    //@ensures day == d;
    public void setDay( int d) {...}
    //@requires 0<=h && h < 24;
    //@ensures hour == h;
    public void setHour( int h) {...}
    //@ensures \result == day && 1<=\result && \result<=31; (!)
    public int getDay() {return day;}
}
class Date
{
    int day;
    int hour;

    //@invariant 1<=day && day<=31;
    //@invariant 0<=hour && hour<24;

    //@requires 1<=d && d <=31;
    //@requires 0<=h && h < 24;
    public Date( int d, int h) {...}
    //@requires 1<=d && d <=31;
    //@ensures day == d;
    public void setDay( int d) {...}
    //@requires 0<=h && h < 24;
    //@ensures hour == h;
    public void setHour( int h) {...}
    //@ensures \result == day && 1<=\result && \result<=31;
    public int getDay() {return day;}
}
boolean contains( int[] array, int element);

What does the method require?
Example 4/2

//@requires array != null;
boolean contains(int[] array, int element);

What does the method ensure?
//@requires array != null;
//@ensures \result <--> \exists int i; 0<=i && i<array.length; array[i] == element;
boolean /*@pure*/ contains( int[] array, int element);
int binarySearch(int[] array, int element);

Returns index of the element.

What does the method require?
Example 5/2

//@requires array != null;
//@requires \forall int i; 0<i && i<array.length;
array[i-1] <= array[i];

int binarySearch( int array[], int element);

What does the method ensure?
//@requires array != null;
//@requires \forall int i; 0<i && i<array.length;
array[i-1] <= array[i];
//@ensures (\exists int i; 0<i && i<array.length;
array[i] == element) ==> array[\result] == element;
//@ensures !(\exists int i; 0<i && i<array.length;
array[i] == element) ==> \result == -1;
int /*@pure*/ binarySearch( int[] array, int element);