Tecnología de la Programación

ESC/Java2

David Cabrero Souto

Facultad de Informática
Universidade da Coruña

Curso 2007/2008

ESC/Java2

- Extended Static Checking for Java.
- `http://secure.ucd.ie/products/opensource/ESCJava2/`
- Encuentra errores de programación.
- Usa técnicas de razonamiento automatizadas basadas en una variante simplificada de Hoare/weakest precondition calculus.
- Usa las anotaciones de JML.
- Trata de probar la corrección de las especificaciones.
- Totalmente automático.
- En tiempo de compilación.

Versus JML

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

- JML *puede encontrar* el error.
- Esc/Java2 *encontrará* el error.
- Validación vs. verificación

Limitaciones

- No es fiable (*sound*).
  - Produce falsos negativos.
- No es completo.
  - Produce falsos positivos.

Limitaciones (II)

- Examina cada método individualmente.
- En el ejemplo se queja de que `b[0]` puede ser una dereferencia de null.

```
class A {
    byte[] b;
    public void n() { b = new byte[20]; }
    public void m() { n();
                      b[0] = 2; }
    ...
```

- Solución añadir postcondición a `n()`

```
//@ ensures b != null && b.length = 20;
```

Limitaciones (II)

- Examina cada método individualmente.
- En el ejemplo se queja de que `b[0]` puede ser una dereferencia de null.

```
class A {
    byte[] b;
    public void n() { b = new byte[20]; }
    public void m() { n();
                      b[0] = 2; }
    ...
```

- Solución añadir postcondición a `n()`

```
//@ ensures b != null && b.length = 20;
```

Limitaciones (II)

- Examina cada método individualmente.
- En el ejemplo se queja de que `b[0]` puede ser una dereferencia de null.

```
class A {
    byte[] b;
    public void n() { b = new byte[20]; }
    public void m() { n();
                      b[0] = 2; }
    ...
```

- Solución añadir postcondición a `n()`

```
//@ ensures b != null && b.length = 20;
```

Beneficios

- Encuentra errores.
- Prueba bien:
  - Ausencia de excepciones en tiempo de ejecución.
    (ej. Null-, ArrayIndexOutOfBounds-, ClassCast-)
  - Propiedades simples
    (ej. invariant n > 0)
- Obliga a escribir contratos.
- Complementa al type-checker.

Ejecución

- CLI.

  ```
  escjava2 [options] file.java
  ```

- GUI.

  ```
  java -jar esctools2.jar
  ```

Ejemplo

```
class Bag {
   int[] a;
   int n;
   Bag(int[] input) {
     n = input.length; a = new int[n];
     System.arraycopy(input, 0, a, 0, n);
   }

   int extractMin() {
     int m = Integer.MAX_VALUE;
     int mindex = 0;
     for (int i = 1; i <= n; i++) {
          if (a[i] < m) { mindex =i; m = a[i]; } }
     n--;
     a[mindex] = a[n];
     return m;
   }
}
```

Ejemplo

```
class Bag {
   int[] a;
   int n;
   Bag(int[] input) {
     n = input.length; a = new int[n];
     System.arraycopy(input, 0, a, 0, n);
   }

   int extractMin() {
     int m = Integer.MAX_VALUE;
     int mindex = 0;
     for (int i = 1; i <= n; i++) {
          if (a[i] < m) { mindex =i; m = a[i]; } }
     n--;
     a[mindex] = a[n];
     return m;
   }
}
```

Warning: Posible dereferencia de nulo

Ejemplo

```
class Bag {
   /@ non_null @/ int[] a;
   int n;
   Bag(int[] input) {
     n = input.length; a = new int[n];
     System.arraycopy(input, 0, a, 0, n);
   }

   int extractMin() {
     int m = Integer.MAX_VALUE;
     int mindex = 0;
     for (int i = 1; i <= n; i++) {
          if (a[i] < m) { mindex =i; m = a[i]; } }
     n--;
     a[mindex] = a[n];
     return m;
   }
}
```

Ejemplo

```
class Bag {
   /@ non_null @/ int[] a;
   int n;
   Bag(int[] input) {
     n = input.length; a = new int[n];
     System.arraycopy(input, 0, a, 0, n);
   }

   int extractMin() {
     int m = Integer.MAX_VALUE;
     int mindex = 0;
     for (int i = 1; i <= n; i++) {
          if (a[i] < m) { mindex =i; m = a[i]; } }
     n--;
     a[mindex] = a[n];
     return m;
   }
}
```

Warning: Posible fuera de rango

Ejemplo

```
class Bag {
   /@ non_null @/ int[] a;
   int n; //@ invariant 0 <= n && n <= a.length;
   Bag(int[] input) {
     n = input.length; a = new int[n];
     System.arraycopy(input, 0, a, 0, n);
   }

   int extractMin() {
     int m = Integer.MAX_VALUE;
     int mindex = 0;
     for (int i = 1; i <= n; i++) {
          if (a[i] < m) { mindex =i; m = a[i]; } }
     n--;
     a[mindex] = a[n];
     return m;
   }
}
```

Ejemplo

```
class Bag {
   /@ non_null @/ int[] a;
   int n; //@ invariant 0 <= n && n <= a.length;
   Bag(int[] input) {
     n = input.length; a = new int[n];
     System.arraycopy(input, 0, a, 0, n);
   }

   int extractMin() {
     int m = Integer.MAX_VALUE;
     int mindex = 0;
     for (int i = 1; i <= n; i++) {
          if (a[i] < m) { mindex =i; m = a[i]; } }
     n--;
     a[mindex] = a[n];
     return m;
   }
}
```

Warning: Posible fuera de rango

Ejemplo

```
class Bag {
   /@ non_null @/ int[] a;
   int n; //@ invariant 0 <= n && n <= a.length;
   Bag(int[] input) {
     n = input.length; a = new int[n];
     System.arraycopy(input, 0, a, 0, n);
   }

   int extractMin() {
     int m = Integer.MAX_VALUE;
     int mindex = 0;
     for (int i = 0; i < n; i++) {
          if (a[i] < m) { mindex =i; m = a[i]; } }
     n--;
     a[mindex] = a[n];
     return m;
   }
}
```

Ejemplo

```
class Bag {
   /@ non_null @/ int[] a;
   int n; //@ invariant 0 <= n && n <= a.length;
   Bag(int[] input) {
     n = input.length; a = new int[n];
     System.arraycopy(input, 0, a, 0, n);
   }

   int extractMin() {
     int m = Integer.MAX_VALUE;
     int mindex = 0;
     for (int i = 0; i < n; i++) {
          if (a[i] < m) { mindex =i; m = a[i]; } }
     n--;
     a[mindex] = a[n];
     return m;
   }
}
```

Warning: Posible índice negativo

Ejemplo

```
class Bag {
   /@ non_null @/ int[] a;
   int n; //@ invariant 0 <= n && n <= a.length;
   Bag(int[] input) {
     n = input.length; a = new int[n];
     System.arraycopy(input, 0, a, 0, n);
   }
   //@ requires n > 0;
   int extractMin() {
     int m = Integer.MAX_VALUE;
     int mindex = 0;
     for (int i = 0; i < n; i++) {
          if (a[i] < m) { mindex =i; m = a[i]; } }
     n--;
     a[mindex] = a[n];
     return m;
   }
}
```