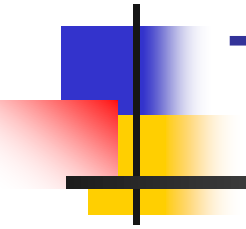
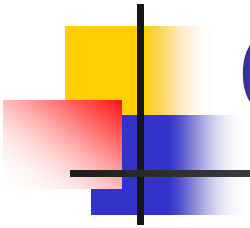


Taller de Programación

Parte III

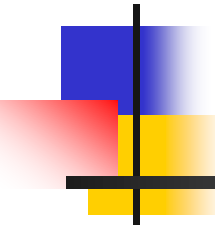
Codificación con miras a la Prueba

Jhonny Felípez Andrade
jrfelizamigo@yahoo.es



Contenido

- Algunos errores de programación.
- Especificación de programas.
  - Precondición.
  - Post condición.
  - Invariantes.
- Diseño por Contratos.
  - JML



Algunos errores de programación.

Algunos errores de Programación

```
void copia(char[]  b)
{  char[] datos = new char(100);
   for (int i = 1; i < b.length; i++)
     datos[i] = b[i]
     ..
     ..
     ..
}
```

¿Que sucedería si ***b*** tuviera mayor cantidad de elementos en comparación a datos?

Algunos errores de Programación

```
#include  <stdio.h>
#include  <stdlib.h>
#include  <string.h>
int  main(int  argc,  char  *argv[])
{  char  copia[2];
   strcpy(copia,argv[1]);
   printf("%s\n",  copia);
   return  0;
}
```

¿Que sucedería si ***argv[1]*** tuviera mayor cantidad que 2 elementos?

Algunos errores de Programación

```
>desp  12
12

>desp  123456  (Obtenemos una lista interminable)
123456
123456
123456
....
```

Algunos errores de Programación

```
class Nombre {
   String  nombre;
   Nombre(String n) { nombre = n;}
   String devuelveNombre() { return  (nombre); }
}

Nombre persona = new Nombre(nombrePersona);
int largo = persona.devuelveNombre().trim().length();
```

¿Qué sucedería cuando nombrePersona sea un valor nulo?

Algunos errores de Programación.
Solución tentativa

```
Nombre persona = new Nombre(nombrePersona);
int largo = 0;
String dato = persona.devuelveNombre();

if (dato == null)
 System.err.println("Error grave: Falta la propiedad
   'nombre'");
else
 largo = dato.trim().length();
```

En un sistema distribuído.

¿Qué ocurriría si no se encuentra la clase?

El sistema falla!

Algunos errores de Programación.
Solución tentativa

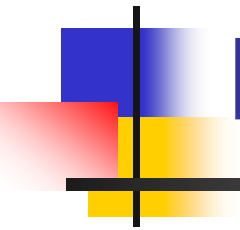
```
Nombre persona = new Nombre(nombrePersona);

if (persona == null)
 System.err.println("Error grave: clase Persona no
   disponible");

int  largo = 0;
String dato = persona.devuelveNombre();

if (persona == null)
 System.err.println("Error grave: Falta la propiedad
   'nombre'");
else
 largo = dato.trim().length();
```

Algunos errores de Programación Consideraciones.

- Cuando un programa ha sido probado y se supone que funciona correctamente aún puede presentar algunas fallas.
- Por otro lado, cuanto más controles se tengan se demorará más tiempo de proceso.

Algunos errores de Programación Consideraciones.

- Entonces será necesario especificar el código de una forma que se pueda determinar una serie de problemas con mucha anticipación.



Especificación de programas.

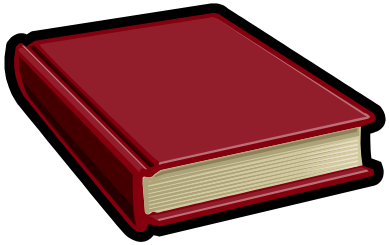
Especificación de programas.

- La especificación de programas tienen su origen en Hoare (años 70) quien desarrolló las bases para la especificación formal de programas.

Destinatarios de una especificación

Usuarios del algoritmo, debe recoger todo lo necesario para un uso correcto del mismo.

Especificación



Algoritmo

Implementador del algoritmo, define los requisitos que cualquier implementación **válida** debe satisfacer.

Especificación de programas.

- Cuando se especifica un programa hay que considerar que no solo puede tener errores, también puede ser que esté equivocada la especificación del mismo. Por esta razón también deben verificarse las especificaciones de los mismos.

Especificación de programas.
Notación.

$$\{P\}C\{Q\}$$

P = Precondición

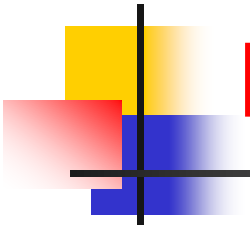
Q = Post condición

C = Código

Ejemplo:

$$\{X=1\}X=X+1\{X=2\}$$

Especificación de programas. Precondición.



- Son lo valores iniciales que toman las variables del programa, o también, los prerrequisitos necesarios para su correcto funcionamiento.
- Por ejemplo, podría ser que una variable no sea nula, que los datos estén ordenados, la existencia de una clase, etc.

Especificación de programas.
Post condición.

- Son los valores finales que toman las variables del programa

Especificación de programas. Correctitud parcial y totall.

Correctitud parcial

$$\{P\}C\{Q\}$$

Correctitud total

$$[P]C[Q]$$

Correctitud total = Correctitud parcial + terminación

Especificación de programas.
Especificación de correctitud total.

$$[P]\ C\ [Q]$$

- Llamemos a [P] precondición, a [Q] post condición y C un programa.
- Para verificar que es correcta, es necesario que el programa C termine.

Especificación de programas.
Ejemplo. Intercambio de variables.

[X = x and Y = y]

R := X ; X := Y ; Y := R

[X = y and Y = x]

Especifica la correctitud total de que los valores de X, Y se intercambian.

Especificación de programas.
Ejemplo. Máximo número.

$\{X > Y\}$

$Z := X ;$

$\{Z > Y \wedge X \wedge Y\}$

$\{verdad\}$

if $X > Y$ then $Z := X$ else $Z := Y$ endif

$\{Z \geq X \wedge Z \geq Y \wedge (Z=X \vee Z=Y)\}$

Especificación de programas.
Otros ejemplos.

{true}
 if X % 2 = 1 then PAR := X + 1; endif
{PAR $\wedge$ X}

{K % 2 = 0 $\wedge$ Y * Z^K=C}
R := K / 2
{Y * Z^{2*R} = C $\wedge$ K}

Especificación de programas.

Ejemplo. División por restas sucesivas.

{X = x and Y = y and X>=0 and Y > 0}
R := X ;
Q := 0;
while Y ≤ R do
  R := R – Y ; Q := Q + 1
{R < Y and X = R + (Y * Q)}

No todas las variables deben especificarse en una precondición, tal el caso de R y Q.

Especificación de programas.
Ejemplo. ¿Esta X en el array A?

$\{A(1..N) \wedge 1 \leq N \wedge X=x\}$

I := 1; ESTA := FALSE;

while not ESTA and I<= N do

    if A(I) = X then ESTA := TRUE; endif;

    I := I + 1;

$\{ESTA = \exists X \in \{A(1),..,A(N)\} \wedge 1 \leq I \leq N \wedge A(1..N) \wedge 1 \leq N\}$

Especificación de programas.

Ejemplo. Cuenta el número que X aparece en el array A.

$\{A(1..N) \wedge 1 \leq N \wedge X=x\}$

I := 1; V := 0;

while I ≤ N do

    if A(I) = X then V := V + 1; endif;

    I = I + 1;

$\{V = \mathit{Nj}\ (1 \leq j \leq N \wedge A(j) = X) \wedge A(1..N) \wedge 1 \leq N\}$

Especificación de programas.

Ejemplos. Cuenta el número de múltiplos de 5.

{A(1..N) $\wedge$ 1 $\leq$ N }

I := 1; CUANTOS := 0;

while I $\leq$ N do

    if A(I) MOD 5 = 0 then CUANTOS := CUANTOS + 1;

    endif;

    I := I + 1;

{CUANTOS = N_j (1 $\leq$ j $\leq$ N $\wedge$ A(j) % 5 = 0) $\wedge$ A(1..N) $\wedge$ 1 $\leq$ N}

Especificación de programas.

Ejemplo. Obtiene el máximo elemento del array A.

$\{A(1..N) \wedge 1 \leq N\}$

I := N; MAX := A(N);

while I > 1 do

    I = I - 1;

    if A(I) > MAX then MAX := A(I); endif;

$\{MAX = A(J) > A(K) \wedge 1 \leq J,K \leq N \wedge$
$A(1..N) \wedge 1 \leq N\}$

Especificación de programas.

Ejemplo. Busqueda Binaria en un array A.

{A(1..N) está ordenado $\wedge$ 1 $\leq$ N $\wedge$ X}
BINSEARCH (A, N, X)
F = 1
L = N
while F <> L do
    M = (F+L)/2
    if A(M)>= X then
        L = M
    else
        F = M + 1
    end if
end while
if A(F) = X then return F end if
return 0
{ F / 1$\leq$ F $\leq$ N $\wedge$ A[F] = X $\vee$ 0 / X no existe}

Especificación de programas.

Ejemplo. Obtiene el valor n-ésimo de la serie de fibonacci.

$\{N \geq 0\}$

A := 0; B := 1; K := 1;

while K ≤ N do

    B := B + A; A := B - A;

    K := K + 1;

$\{A=S_N \text{ and } N \geq 0 \wedge S_0 = 0 \wedge S_1 = 1 \wedge S_K = S_{K-1} + S_{K-2} \text{ and } K \geq 2\}$

Especificación de programas.

Ejemplo. Obtiene el factorial de un número natural N.

$\{N \geq 0\}$

F := 1; I := N;

while I ≥ 1 do

    F := F * I; I := I - 1;

$\{(F=N!=N*(N-1)*(N-2)*...*3*2*1) \vee (F = 1 \wedge N =0)\}$

Especificación de Programas. Invariantes

- Una invariante es una propiedad que es verdadera y no cambia durante la ejecución del programa.
- Esta propiedad engloba a la mayor parte de los elementos o variables del programa.
- Cada ciclo tiene una propiedad invariante.
- Las invariantes explican las estructuras de datos y algoritmos.

Especificación de Programas
Invariantes - Ejemplo

Invariante: $R = 2^I$

$\{R = 2^I\}$

$I := I + 1;$

$R := R * 2$

$\{R = 2^I\}$

Especificación de Programas
Invariantes - Ejemplo

- **Condición inicial y final en un bucle.**

```
I= 0; Q= 0; P= 1;
while (I<N)
{
  I= I+1; Q= Q+P; P= P+2;
}
```

- **Antes del bucle podemos establecer la siguiente condición:**
  - **$\{P_0 = I=0 \wedge Q=0 \wedge P=1\}$**
- **¿Que podemos establecer después del bucle?**
  - **$\{P_n = I=N \wedge Q=? \wedge P=?\}$**

Especificación de Programas
Invariantes – Ejemplo Cont.

- Las variables van cambiando de valor; pero se mantienen invariables ciertas relaciones entre ellas.
- Para ver estas relaciones, veamos los valores para los estados P_i observados justo antes de evaluar la condición del *while*.
- Estudiando los valores podemos comprobar que la relación entre las tres variables es:
- $Q = I^2 \wedge P = 2I+1 \wedge I \in \{0..N\}$ ← Invariante

Especificación de Programas

Invariantes – Verificación matemática con ejemplo de ejecución.

Iteración	N	I	Q	P	$Q=I^2$	$P=2I+1$
0	3	0	0	1	0	1
1	3	1	1	3	1	3
2	3	2	4	5	4	5
3	3	3	9	7	9	7

Especificación de Programas

Invariantes – Factorial de un número natural N.

Invariante: F = I!

{N ≥ 0}

F := 1; I := 0;

while N > I do

    I := I + 1; F := F * I;

{(F=N!=N*(N-1)*(N-2)*...*3*2*1) ∨ (F = 1 ∧ N =0)}

Especificación de Programas

Invariantes – Verificación matemática con ejemplo de ejecución.

Iteración	N	F	I	F=I!
0	3	1	0	1
1	3	1	1	1
2	3	2	2	2
3	3	6	3	6

Especificación de Programas

Invariantes – Producto en base a sumas.

Invariante: $PROD = \sum_{P=1}^{J-1} A$

{A,B $\in$ naturales $\wedge$ B > 0}

PROD := 0; J :=1;

while J<=B do

  J := J + 1;

  PROD := PROD + A;

{PROD = N * A}

Especificación de Programas

Invariantes – Verificación matemática con ejemplo de ejecución.

Iteración	A	B	PROD	J	$PROD = \sum_{P=1}^{J-1} A$
0	3	2	0	1	0
1	3	2	3	2	3
2	3	2	6	3	6

Especificación de Programas

Invariantes – Potencia.

Invariante: $R=A^X \wedge 0 \leq X < B \wedge B = b \wedge A = a$

$\{A = a \wedge B = b \wedge a > 0 \wedge b >= 0\}$

X = 0; P = 1;

while (X < B) do

    P = P * A;

    X = X + 1;

$\{P = A^B\}$

Especificación de Programas

Invariantes – Verificación matemática con ejemplo de ejecución.

Iteración	A	B	X	P	$R=A^X$
0	2	3	0	1	1
1	2	3	1	2	2
2	2	3	2	4	4
3	2	3	3	8	8

Especificación de Programas

Invariantes – Suma de los elementos de un vector.

Invariante: $S = \sum_{J=0}^{I-1} V(J) \wedge 0 \leq I \leq N$

Pre {V (0..N) $\wedge$ N ≥ 0}

S = 0; I = 0;

while (I < N) do

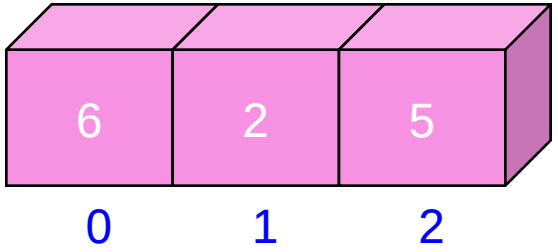
    S = S + V(I);

    I = I + 1;

Post { $S = \sum_{I=0}^{N-1} V(I)$ }

Especificación de Programas

Invariantes – Verificación matemática con ejemplo de ejecución.



Iteración	N	S	I	$S = \sum_{J=0}^{I-1} V(J)$
0	3	0	0	0
1	3	6	1	6
2	3	8	2	8
3	3	13	3	13

Especificación de Programas

Invariantes – Ejercicio.

Determinar la invariante del siguiente ciclo:

```
X:=0;
while (X < 20) do
   X:=X+5;
```

Especificación de Programas

Invariantes – Ejercicio.

Determinar la invariante del siguiente ciclo:

{N ≥ 1}

PROD := 0; I :=1;

while I<>N+1 do

  PROD := PROD + I * I;

  I := I + 1;

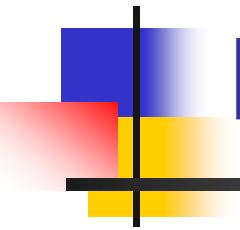
{ $PROD = \sum_{I=1}^{N} I^2$ }

Especificación de Programas

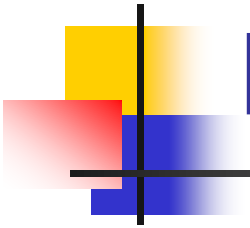
Invariantes – Ejercicio.

¿Cual es la invariante?

```
int buscar(int [] v, int t){
// precondición el vector v no esta ordenado
// precondición t es el valor a buscar
int i=0;
while ((i<v.length()&&(v[i]!=t) { i++; }
if (i<n)
   return (i)
else
   return (-1)
}
// post condición devuelve -1 cuando no existe y
//                devuelve la posición i cuando existe
```

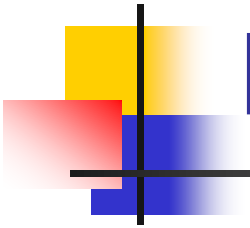


Diseño por Contratos



Metodologías de Desarrollo

- Programación defensiva
- Programación Extrema
- Diseño por contratos



Diseño por Contratos

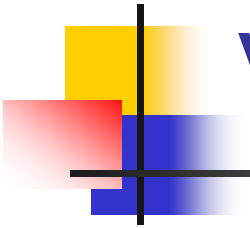
- Orígenes en 1990 con los trabajos de Bertrand Meyer, y el lenguaje de programación Eifel.
- Establece una serie de relaciones con fundamentos en la lógica formal entre diferentes clases del programa.
- Se orienta a la detección de errores en tiempo de ejecución y facilidad de mantenimiento

Diseño por Contratos.
Consideraciones.

¿Qué es Verificación de Programas?

“La verificación asegura que cada función del programa trabaja correctamente.”

— Pfleeger/Atlee



Verificación de Programas

¿Qué es ESC/Java?

Extended Static Checker – Verificador estático extendido

Diseñado originalmente en Compaq

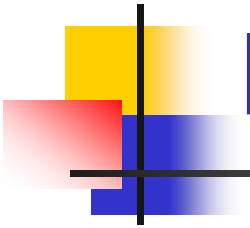
por K. R. M. Leino, Greg Nelson, etc

Diseño por Contratos. Consideraciones.

- Las características principales de **ESC/Java** son:
  - Está basado en el JML

    JML: Java Modeling Language

    Este es un lenguaje de especificación para el Java.
  - El método detrás del JML es el Diseño por Contrato.



Diseño por Contratos

- Las pre y post condiciones definen un tipo de contrato entre un proveedor y sus clientes:
  - Cliente Es un programa que hace uso de un objeto (programa) conoce la interfaz de la clase y no sabe nada de la implementación.
  - Proveedor Es un programa (objeto) que es utilizado por un cliente, construye la documentación, la mantiene y publica una interfaz.
- La motivación principal para desarrollar el diseño por contratos es la reutilización del código.

Diseño por Contratos

	Obligaciones	Beneficios
Cliente	Satisfacer la precondición.	Obtener los resultados de la post condición.
Proveedor	Satisfacer la post condición.	Un proceso mas simple gracias a que se presume que se satisface la precondición.

Diseño por Contratos

- **Precondiciones, Post condiciones e Invariantes en JML.**

Ej.

```
//@ requires 0 <= indice && indice < count();
//@ ensures \result != null;
public Item getItem(int indice) {
   ...
}
```

Ej.

```
private ArrayList items;
//@ loop_invariant items != null;
```

JML - Expresiones

- Una expresión de especificación es una expresión similar a un predicado en lógica, que se evalúa a cierto o falso. Se resume en la siguiente tabla:

Operadores lógicos	&&, ||, !, ==, !=, <, <=, ==>, <==, <==>, <=!=> (y, o, no, igual, distinto, menor, menor igual, implicación, contra implicación, equivalencia, no equivalencia)
Cuantificadores	\forall, \exists, \sum, \product, \num_of, \max, \min (para todo, existe, suma, producto, número de, máximo, mínimo)

JML – Sintaxis de los cuantificadores

`(\sum int I; 0 <= I && I < 5; I)`	`== 0 + 1 + 2 + 3 + 4`
`(\product int I; 0 < I && I < 5; I)`	`== 1 * 2 * 3 * 4`
`(\max int I; 0 <= I && I < 5; I)`	`== 4`
`(\min int I; 0 <= I && I < 5; I-1)`	`== -1`
`(\num_of int I; 0 <= I && I < 5; I*2 < 6)`	`== 3`

JML – seudo variables de las post condiciones.

\result	valor devuelto por el método
\old(E)	valor que tomaba la expresión E al comenzar la ejecución del método

Invariante.

Ejemplo. Obtiene el factorial de un número natural N.

Invariante: $F = I!$

$\{N \geq 0\}$

$F := 1; I := 0;$

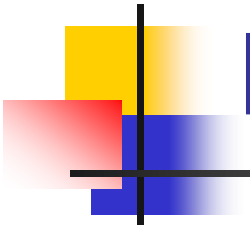
while $N > I$ do

$I := I + 1; F := F * I;$

$\{(F=N!=N*(N-1)*(N-2)*...*3*2*1) \vee (F = 1 \wedge N = 0)\}$

JML - Ejemplo

```
public class Factorial
{
  /*@ requires n >= 0;
    @ ensures  (\result == (\product int i; 0 < i && i <= n; i)) || (\result == 1
    && n == 0);
    @*/

  public static int fact (int n)
  {
    int f = 1, i = 0;

    //@ loop_invariant f == (\product int j; 0 < j && j <= i; j);
    while (n > i)
    {
           i = i + 1; f = f * i;
    }
    return f;
  }

  public static void main(String[] args)
  {
    System.out.println(Factorial.fact(3));
  }
}
```

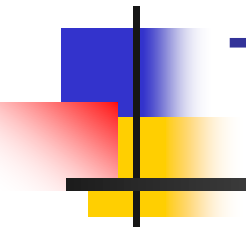


Enlaces Interesantes

- http://www.jmlspecs.org

Bibliografía

- Fundamentos de Programación, Jorge Teran Pomier, 2006.



Taller de Programación

Gracias