UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

etsinf

Tema 1. Recursividad

Programación (PRG)

Jorge González Mollá

Departamento de Sistemas Informáticos y Computación

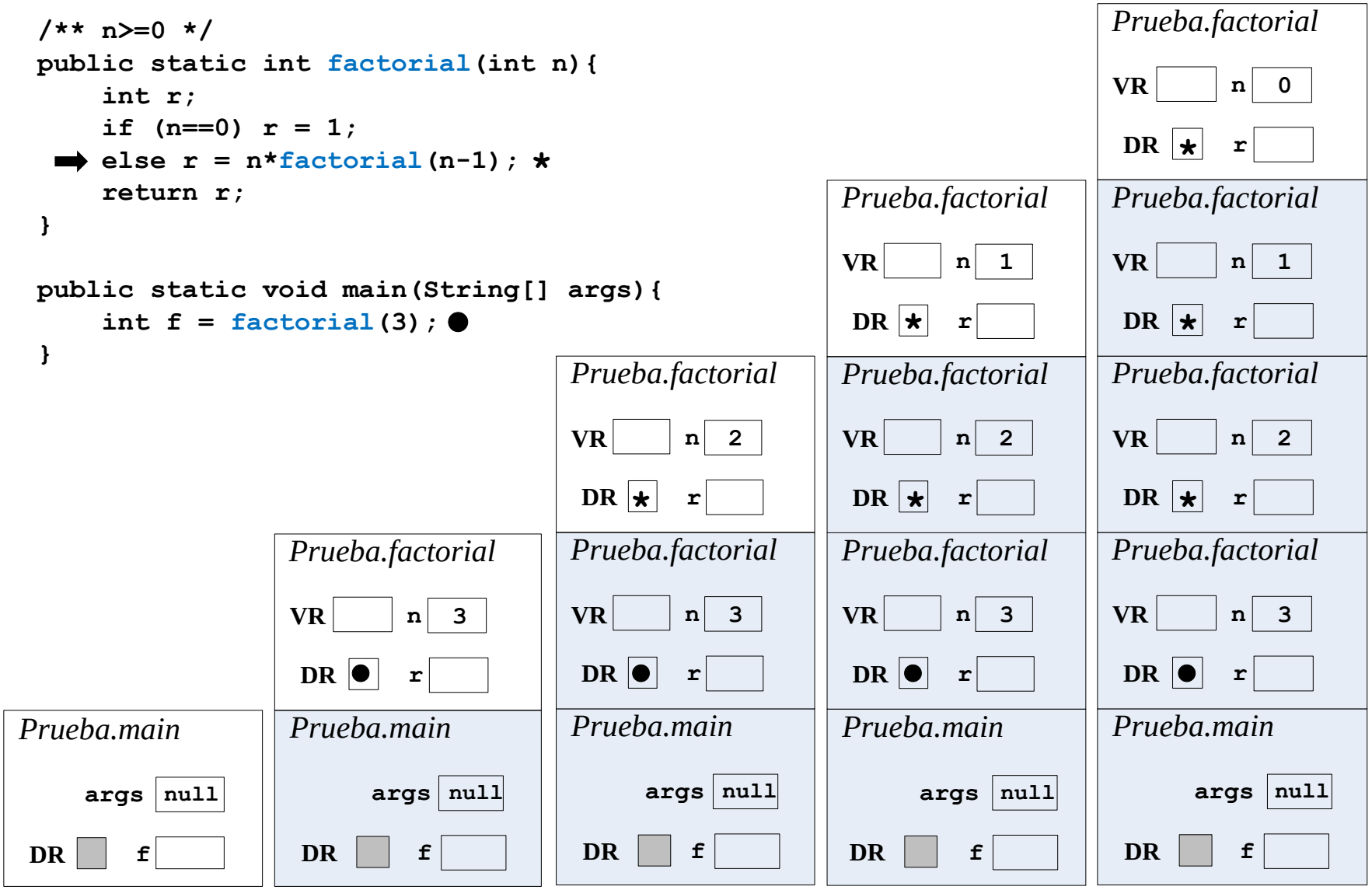
Índice

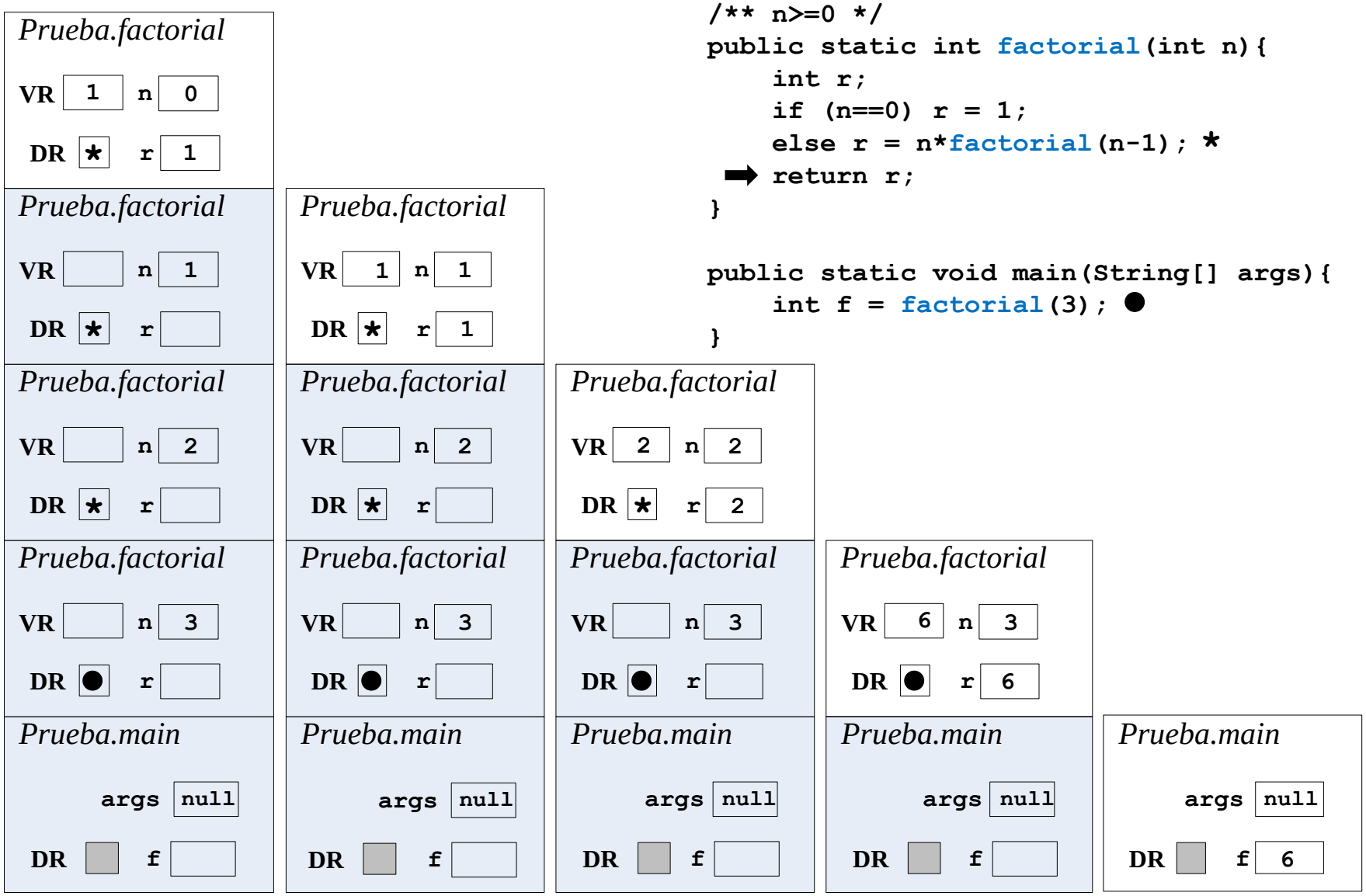
Recursividad y pila de llamadas

- Cada llamada recursiva a un mismo método está asociada a un registro de activación propio que se apila en la pila de llamadas. Es decir, existen tantos registros de activación como llamadas pendientes. De todos ellos, sólo está activo el que está en la cima de la pila.
- Cuando la ejecución de un método finaliza, deja de existir su registro de activación (se desapila). En el caso recursivo, la ejecución puede reanudarse en una ejecución inmediatamente anterior del mismo método, que habrá dejado de estar pendiente.
- En recursividad se hace un uso intensivo de la pila de llamadas.
- Puede llegar a provocar serios problemas por agotamiento de la memoria, produciéndose un desbordamiento de la pila (stack overflow).
- La causa habitual del desbordamiento de la pila es la recursividad infinita, provocando la excepción `StackOverflowError`.

```
/** n>=0 */
public static int factorial(int n){
    int r;
    if (n==0) r = 1;
 ➡  else r = n*factorial(n-1); *
    return r;
}

public static void main(String[] args){
    int f = factorial(3); ●
}
```




Prueba.factorial
VR 1 n 0
DR * r 1
Prueba.factorial
VR n 1
DR * r
Prueba.factorial
VR 1 n 1
DR * r 1
Prueba.factorial
VR n 2
DR * r
Prueba.factorial
VR n 2
DR * r
Prueba.factorial
VR 2 n 2
DR * r 2
Prueba.factorial
VR n 3
DR ● r
Prueba.factorial
VR n 3
DR ● r
Prueba.factorial
VR n 3
DR ● r
Prueba.factorial
VR 6 n 3
DR ● r 6
Prueba.main
args null
DR f
Prueba.main
args null
DR f
Prueba.main
args null
DR f
Prueba.main
args null
DR f
Prueba.main
args null
DR f 6
/** n>=0 */
public static int factorial(int n){
int r;
if (n==0) r = 1;
else r = n*factorial(n-1); *
return r;
}
public static void main(String[] args){
int f = factorial(3); ●
}

Recursividad y pila de llamadas

- Si se compara el uso de la pila que provocan las llamadas `factorial(n)` en sus versiones iterativa y recursiva del citado método, se puede concluir que dicho uso es mucho mayor en la versión recursiva que en la iterativa.
- En la versión iterativa de `factorial`, la coexistencia simultánea de registros en la pila es de sólo 2 (el registro asociado al método `factorial` y el registro de activación correspondiente al método `main`).
- En la pila de llamadas de la versión recursiva de `factorial`, pueden llegar a coexistir simultáneamente hasta n+2 registros (los n+1 registros asociados a las distintas llamadas recursivas sobre el método `factorial`, además del registro de activación correspondiente al método `main`).
- Como resumen, el consumo de memoria del método `factorial` iterativo es siempre el mismo (constante); no obstante, el del `factorial` recursivo es variable y depende linealmente del valor de su argumento n.