

Sistemas Operativos

Oscar Bedoya

`oscarbed@eisc.univalle.edu.co`

- * Procesos
- * Hilos en Java
- * Problemas

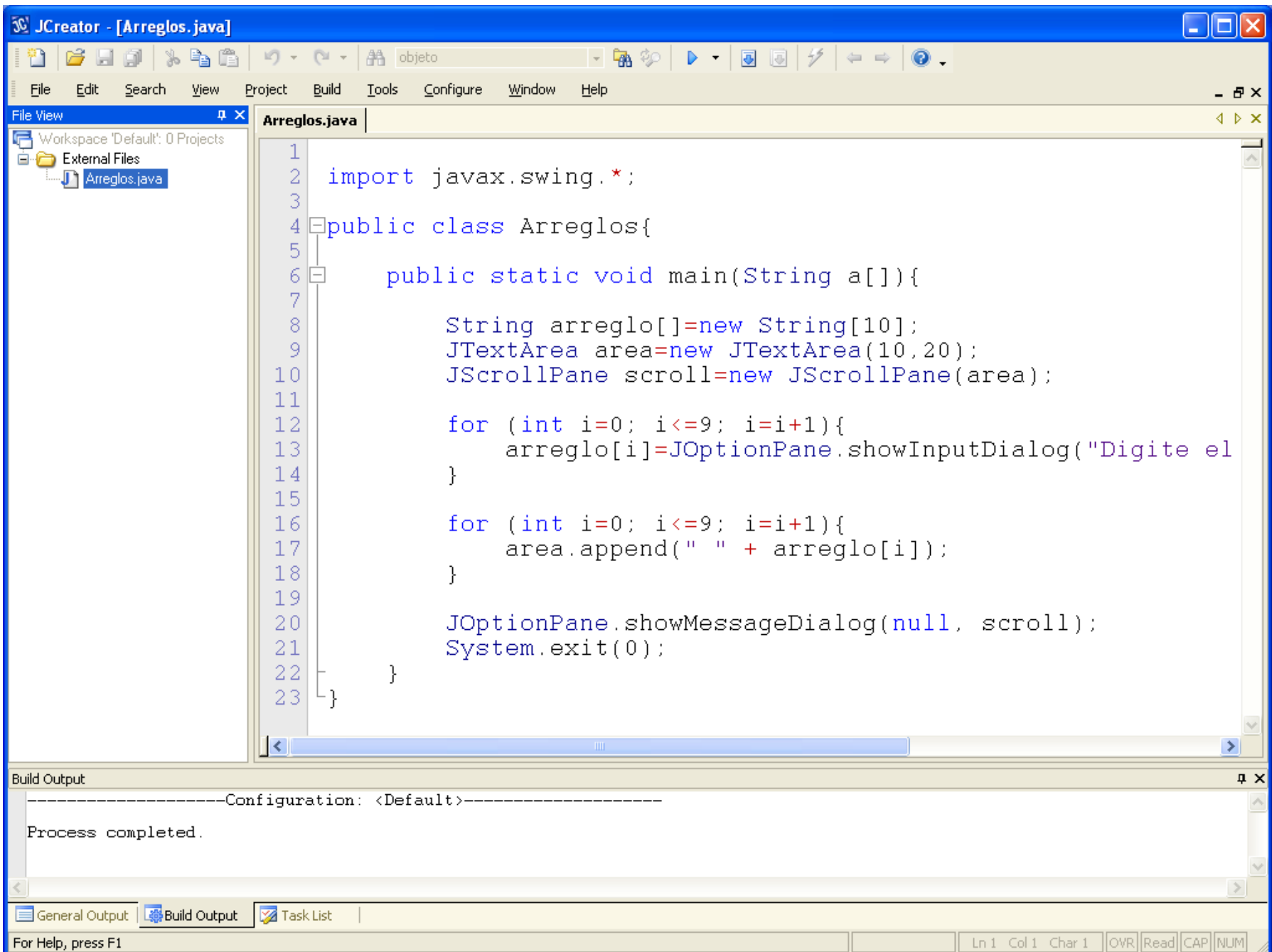
Procesos - Hilos

Proceso

- Es un **programa en ejecución**
- Entidad que **requiere de recursos** para su ejecución (memoria, procesador, dispositivos de E/S)

Procesos - Hilos





```
import javax.swing.*;
public class Arreglos{
    public static void main(String a[]){
        String arreglo[]=new String[10];
        JTextArea area=new JTextArea(10,20);
        JScrollPane scroll=new JScrollPane(area);

        for (int i=0; i<=9; i=i+1){
            arreglo[i]=JOptionPane.showInputDialog("Digite un dato para la posición " + i);
        }

        for (int i=0; i<=9; i=i+1){
            area.append(" " + arreglo[i]);
        }

        JOptionPane.showMessageDialog(null, scroll);
        System.exit(0);
    }
}
```

```
import javax.swing.*;
public class Arreglos{
    public static void main(String a[]){
        → String arreglo[]=new String[10];
        JTextArea area=new JTextArea(10,20);
        JScrollPane scroll=new JScrollPane(area);

        for (int i=0; i<=9; i=i+1){
            arreglo[i]=JOptionPane.showInputDialog("Digite un dato para la posición " + i);
        }

        for (int i=0; i<=9; i=i+1){
            area.append(" " + arreglo[i]);
        }

        JOptionPane.showMessageDialog(null, scroll);
        System.exit(0);
    }
}
```

```
import javax.swing.*;
```

```
public class Arreglos{
```

```
    public static void main(String a[]){
```

```
        → String arreglo[]=new String[10];
```

```
        JTextArea area=new JTextArea(10,20);
```

```
        JScrollPane scroll=new JScrollPane(area);
```

```
        for (int i=0; i<=9; i=i+1){
```

```
            arreglo[i]=JOptionPane.showInputDialog("Digite un dato para la posición " + i);
```

```
        }
```

```
        for (int i=0; i<=9; i=i+1){
```

```
            area.append(" " + arreglo[i]);
```

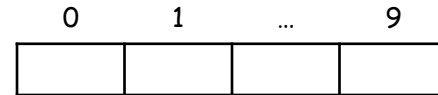
```
        }
```

```
        JOptionPane.showMessageDialog(null, scroll);
```

```
        System.exit(0);
```

```
    }
```

```
}
```




```

import javax.swing.*;
public class Arreglos{
    public static void main(String a[]){
        String arreglo[]=new String[10];
        JTextArea area=new JTextArea(10,20);
        JScrollPane scroll=new JScrollPane(area);

        for (int i=0; i<=9; i=i+1){
            arreglo[i]=JOptionPane.showInputDialog("Digite un dato para la posición " + i);
        }

        for (int i=0; i<=9; i=i+1){
            area.append(" " + arreglo[i]);
        }

        JOptionPane.showMessageDialog(null, scroll);
        System.exit(0);
    }
}

```

0	1	...	9
Sarah			

```

import javax.swing.*;
public class Arreglos{
    public static void main(String a[]){
        String arreglo[]=new String[10];
        JTextArea area=new JTextArea(10,20);
        JScrollPane scroll=new JScrollPane(area);

        for (int i=0; i<=9; i=i+1){
            arreglo[i]=JOptionPane.showInputDialog("Digite un dato para la posición " + i);
        }

        for (int i=0; i<=9; i=i+1){
            area.append(" " + arreglo[i]);
        }

        JOptionPane.showMessageDialog(null, scroll);
        System.exit(0);
    }
}

```

0	1	...	9
Sarah	Oscar		Adriana



Procesos – Hilos

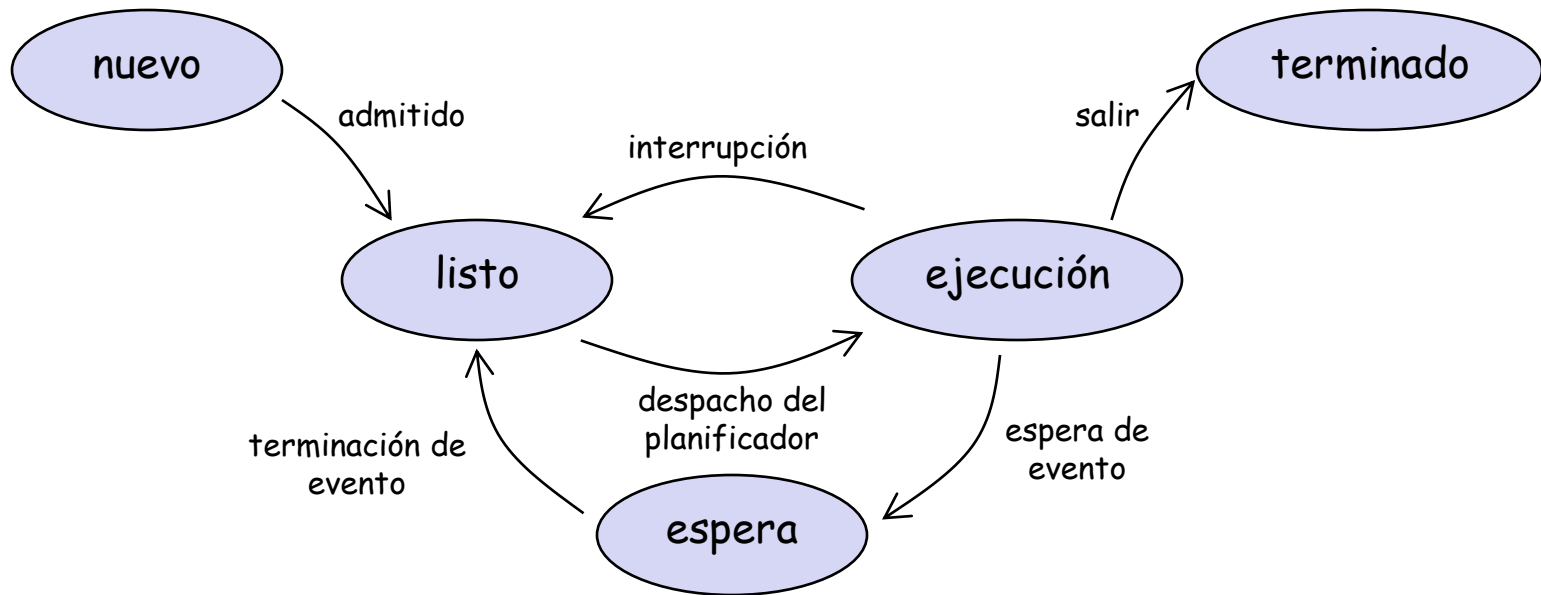
Bloque de control de proceso (PCB, *Process Control Block*)

- Un bloque de control de proceso es la forma como se representa cada proceso dentro del sistema operativo

apuntador	estado del proceso
número de proceso	
contador de programa	
registros*	
prioridad	-
...	

Procesos - Hilos

Estado de un proceso



```
import javax.swing.*;
```

```
public class Arreglos{
```

```
    public static void main(String a[]){
```

→

```
        String arreglo[]=new String[10];
```

```
        JTextArea area=new JTextArea(10,20);
```

```
        JScrollPane scroll=new JScrollPane(area);
```

```
        for (int i=0; i<=9; i=i+1){
```

```
            arreglo[i]=JOptionPane.showInputDialog("Digite un dato para la posición " + i);
```

```
        }
```

```
        for (int i=0; i<=9; i=i+1){
```

```
            area.append(" " + arreglo[i]);
```

```
        }
```

```
        JOptionPane.showMessageDialog(null, scroll);
```

```
        System.exit(0);
```

```
    }
```

```
}
```

apuntador	ejecución
034	
línea 4	
0	1 ... 9
<input type="text"/>	<input type="text"/>
5	-

```
import javax.swing.*;
public class Arreglos{
    public static void main(String a[]){
        String arreglo[]=new String[10];
        JTextArea area=new JTextArea(10,20);
        JScrollPane scroll=new JScrollPane(area);
```

apuntador		ejecución	
034			
línea 8			
0	1	...	9
Sarah			
5		-	

```
        for (int i=0; i<=9; i=i+1){
            arreglo[i]=JOptionPane.showInputDialog("Digite un dato para la posición " + i);
        }
```

```
        for (int i=0; i<=9; i=i+1){
            area.append(" " + arreglo[i]);
        }
```

```
        JOptionPane.showMessageDialog(null, scroll);
        System.exit(0);
```

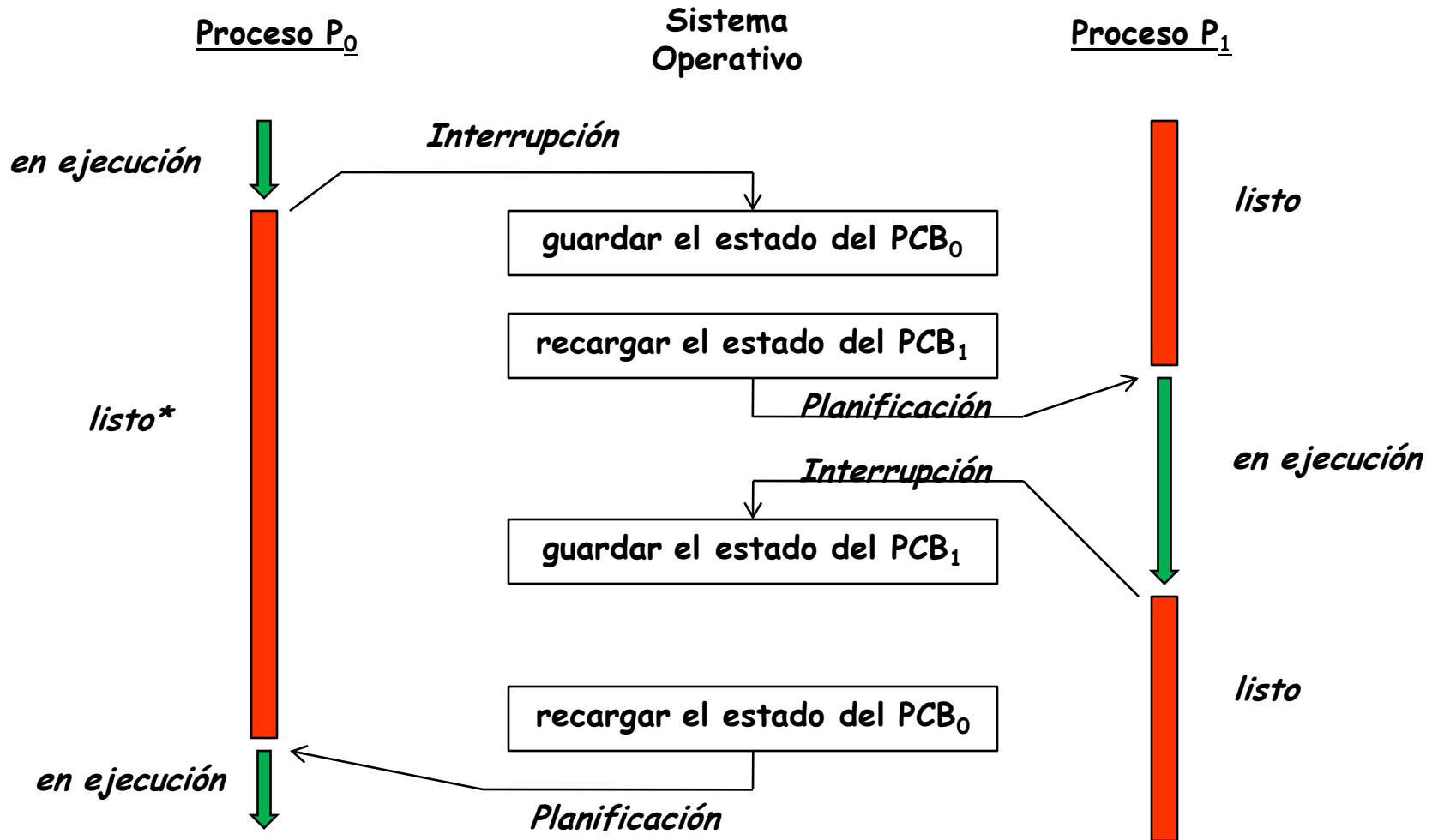
```
    }
```

```
}
```

Procesos - Hilos

Sistema Operativo	
Procesos del sistema (ejecutan código del sistema)	Procesos de usuario (ejecutan código de usuario)

Procesos - Hilos



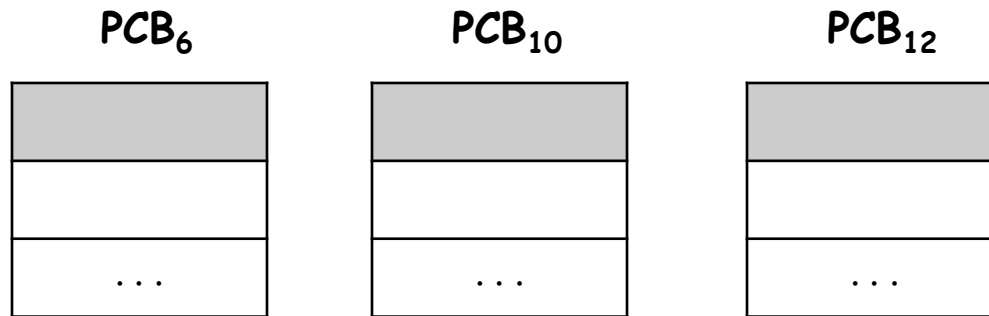
* El estado en el PCB_0 pasa a listo

Procesos - Hilos

- Mientras los procesos esperan para ser atendidos se colocan en colas de planificación

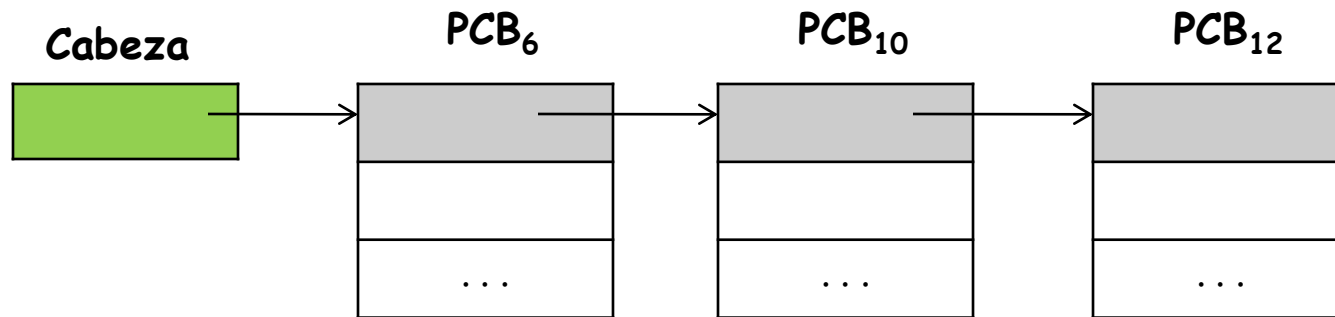
Procesos - Hilos

Colas



Procesos - Hilos

Colas

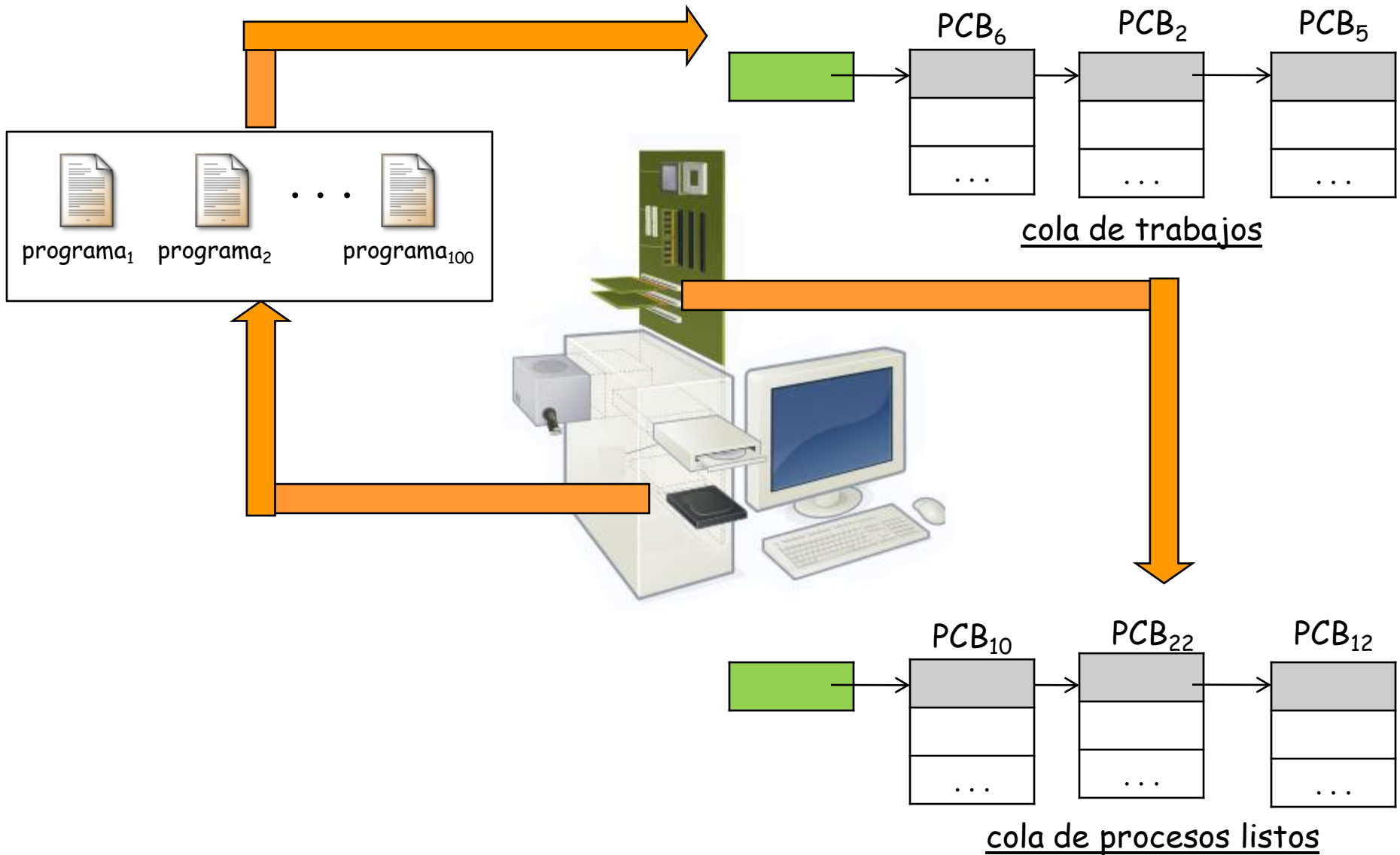


Procesos - Hilos

Colas de planificación

- Cola de trabajos
- Cola de procesos listos
- Colas de dispositivos

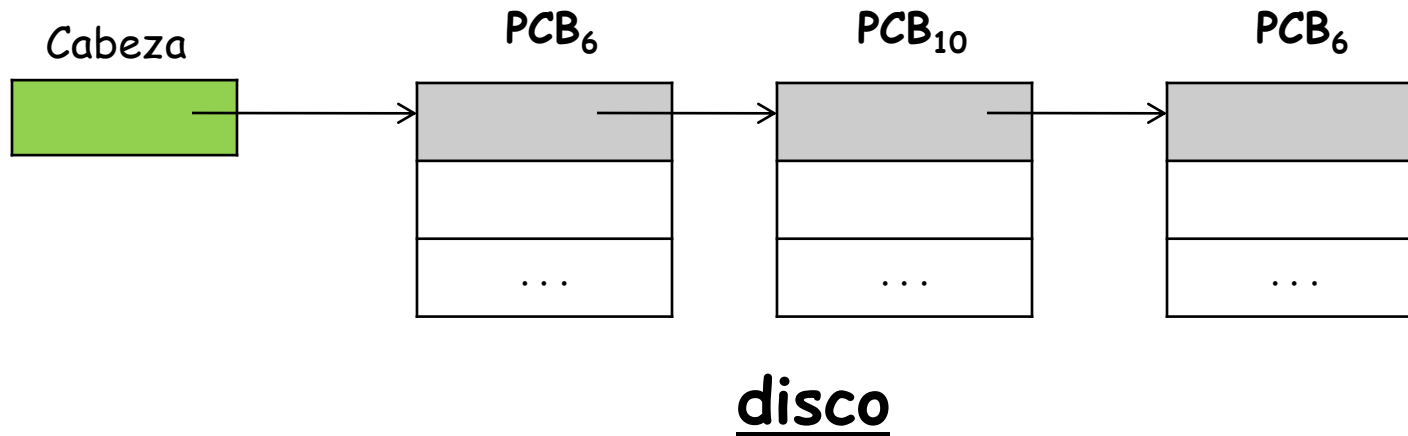
Procesos - Hilos



Procesos - Hilos

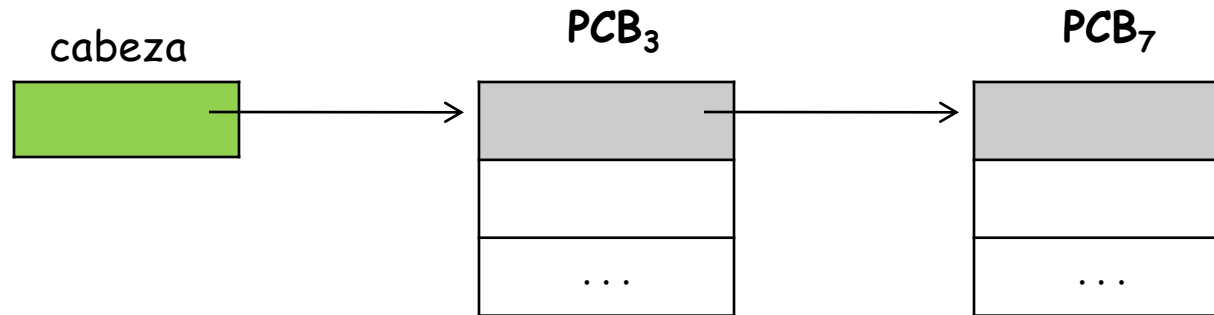
Cola de trabajos

- Reside en disco y contiene los PCB de los procesos que acaban de ser creados



Procesos - Hilos

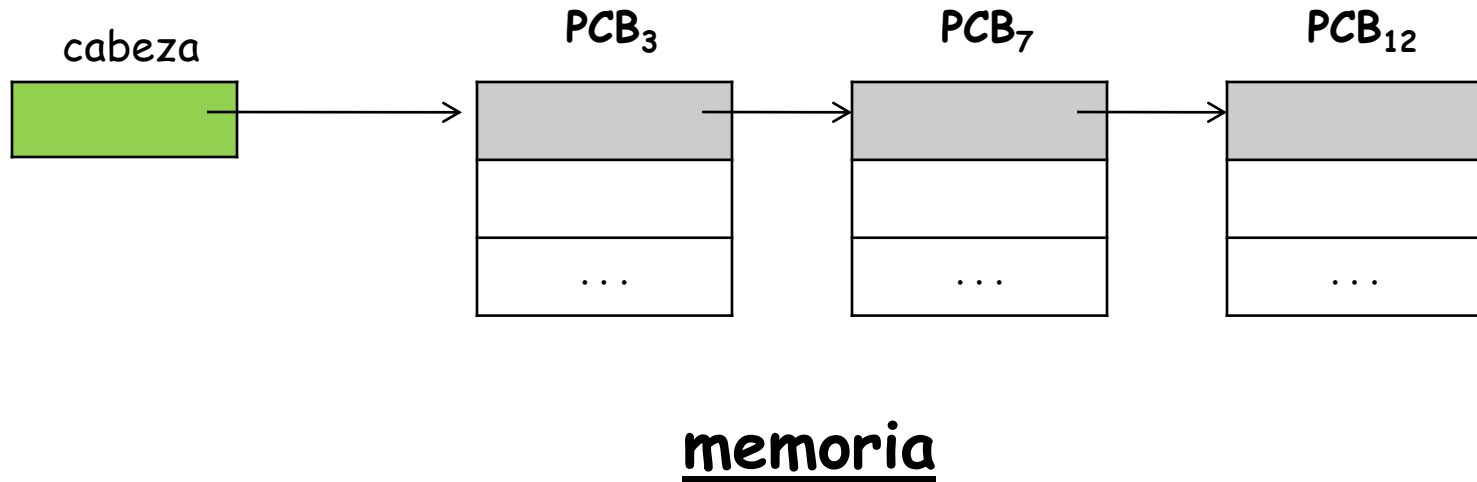
Colas de procesos listos



memoria

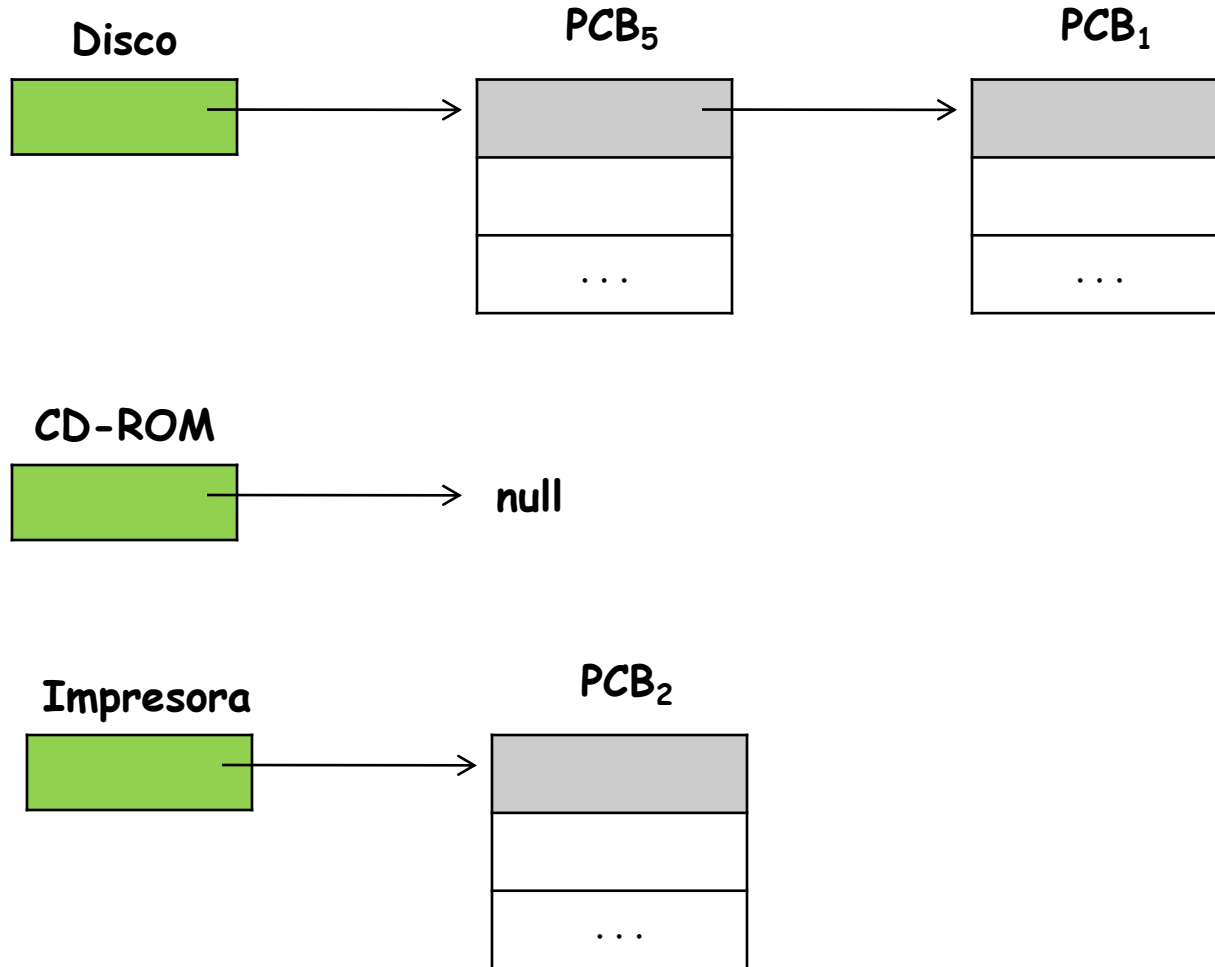
Procesos - Hilos

Colas de procesos listos



Procesos - Hilos

Colas de dispositivos



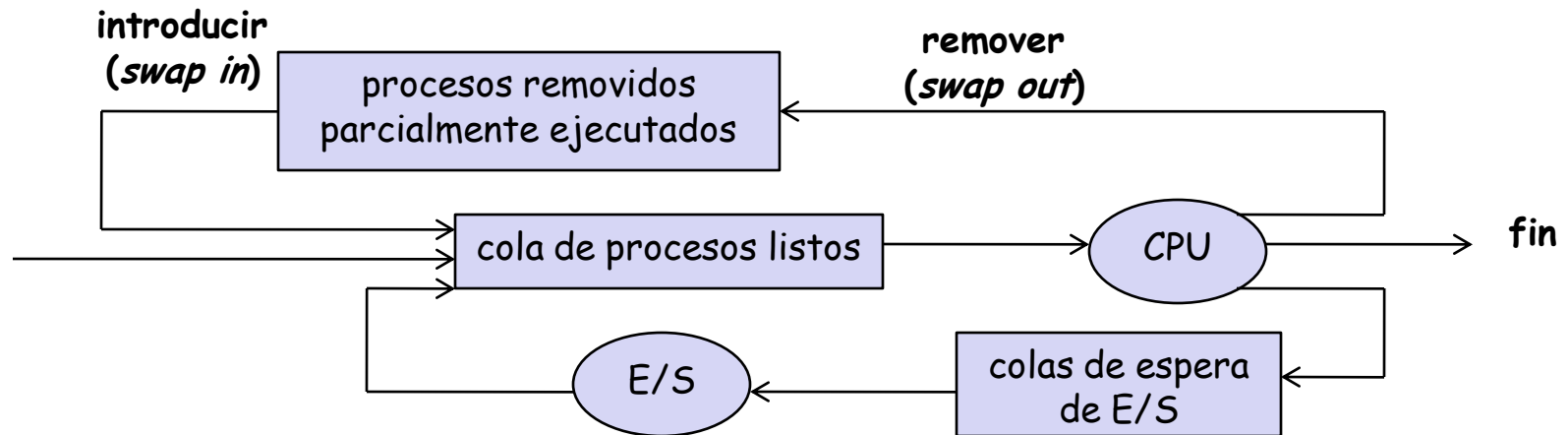
Procesos – Hilos

Planificadores

- **Planificador de largo plazo.** Se utiliza sobre la cola de trabajos que reside en disco. Su frecuencia de utilización puede ser de minutos. Utiliza mezcla de procesos
- **Planificador de mediano plazo.** *Swapping*
- **Planificador de corto plazo.** Se utiliza sobre la cola de procesos listos para decidir quién usa la CPU. Se ejecuta una vez cada 100 milisegundos

Procesos - Hilos

Diagrama de un planificador de mediano plazo



Procesos – Hilos

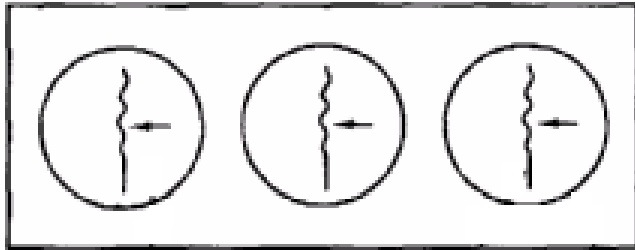
Hilos y procesos

- Hay procesos que pueden tener más de una tarea asociada
- Word ejecuta al tiempo:
 - Corrector ortográfico
 - Papelera
 - Interfaz gráfica
- Procesos que manejan más de un hilo. **Multihilo**

Procesos - Hilos

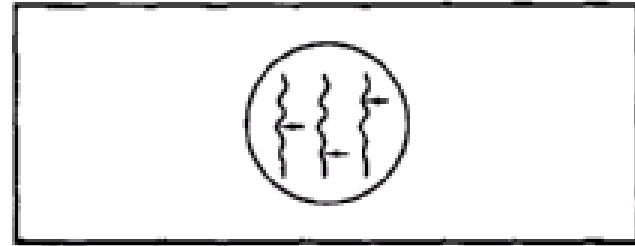
Hilos (Threads)

Computadora



- Tres procesos cada uno con un hilo

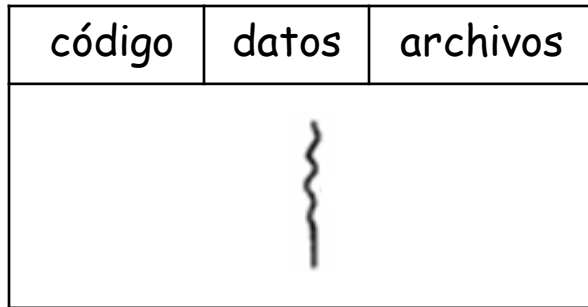
Computadora



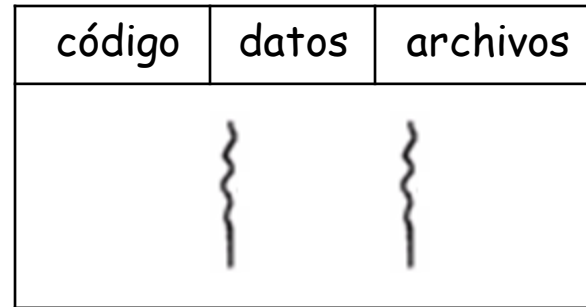
- Un proceso con tres hilos

Procesos - Hilos

Hilos (Threads)



proceso con un solo hilo



proceso multihilo

- Los hilos comparten su sección de código, de datos y otros recursos del sistema operativo

Procesos – Hilos

Beneficios

Los beneficios de la programación multihilos se puede clasificar en cuatro categorías:

- Grado de respuesta
- Compartir recursos
- Utilización de arquitecturas de multiprocesadores

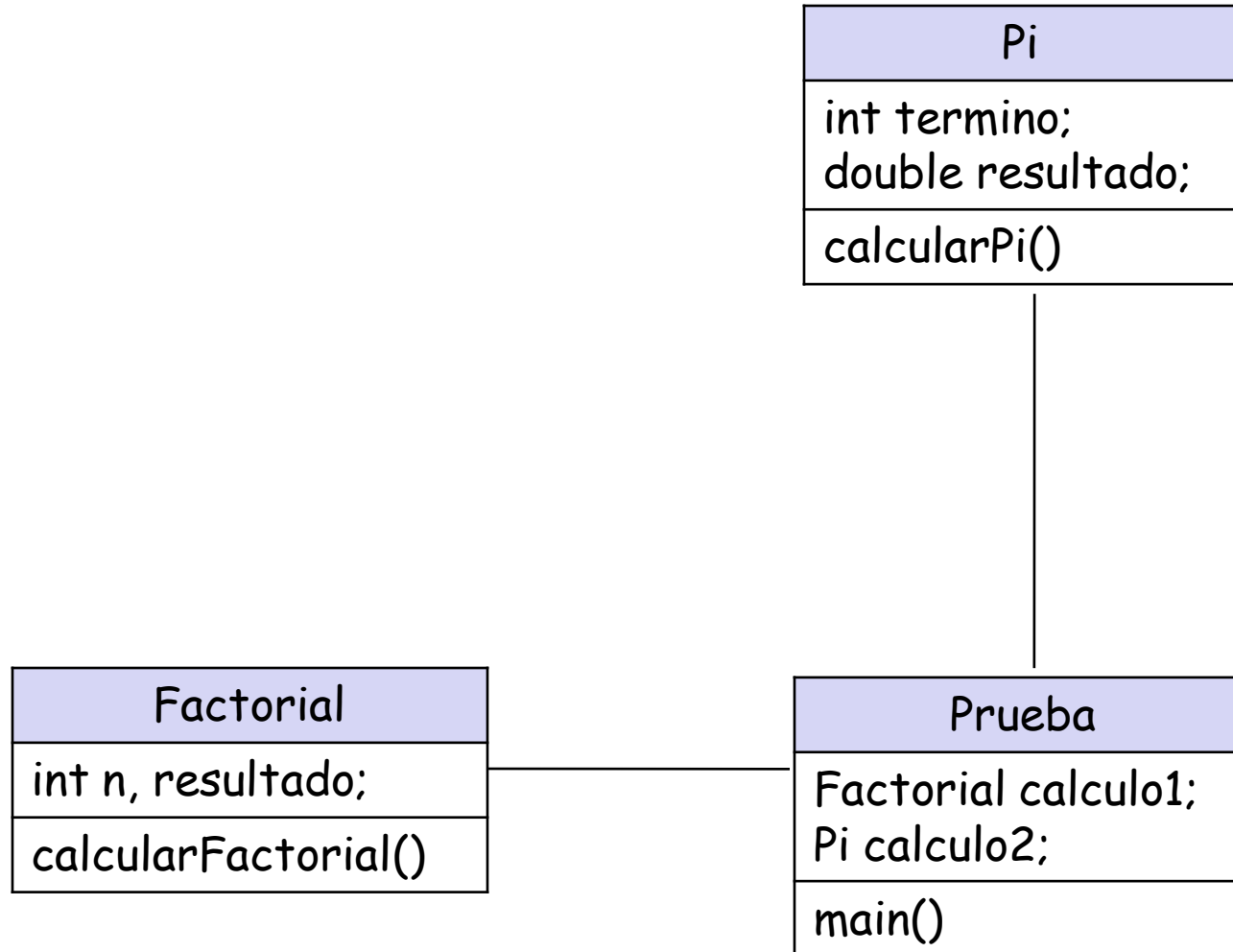
Procesos – Hilos

Hilos en Java

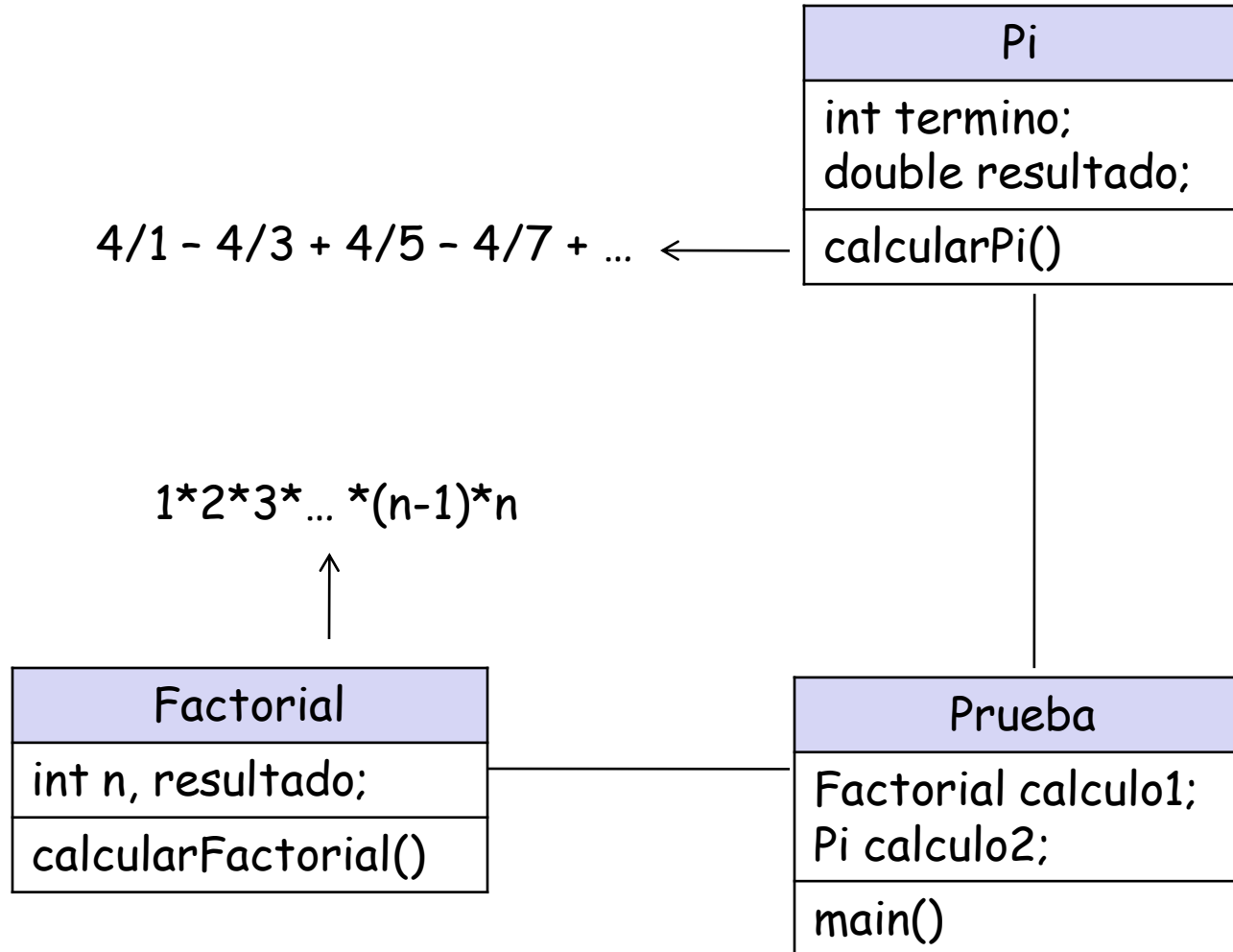
Existen dos formas de trabajar con **hilos en Java**

- Por medio de la clase **Thread**
- Por medio de la interfaz **Runnable**

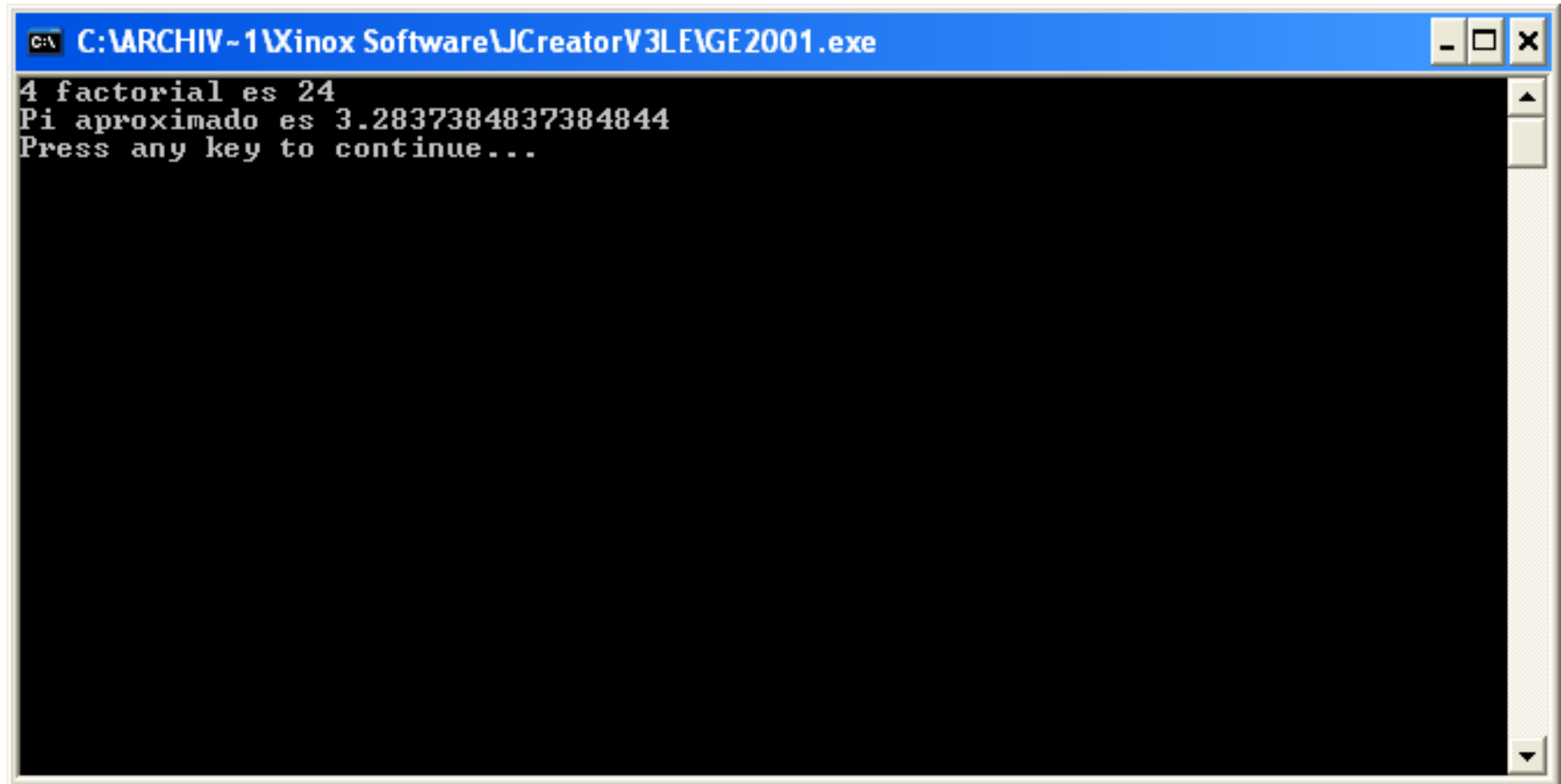
Procesos - Hilos



Procesos - Hilos



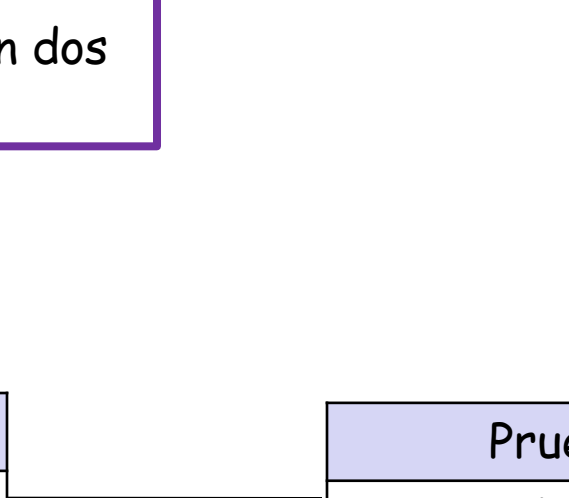
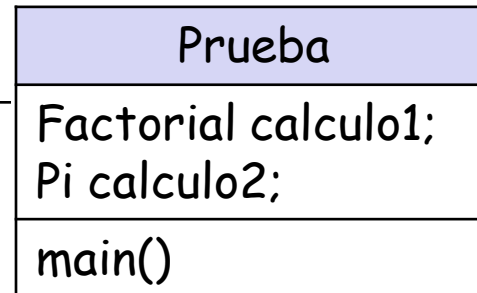
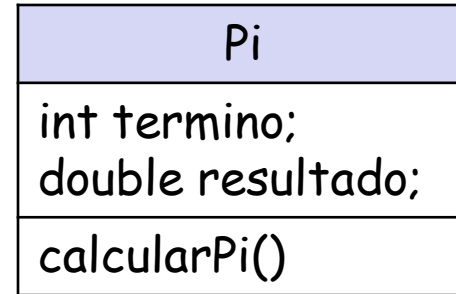
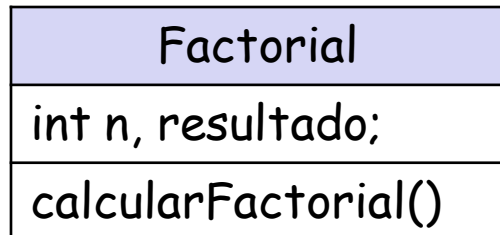
Procesos - Hilos



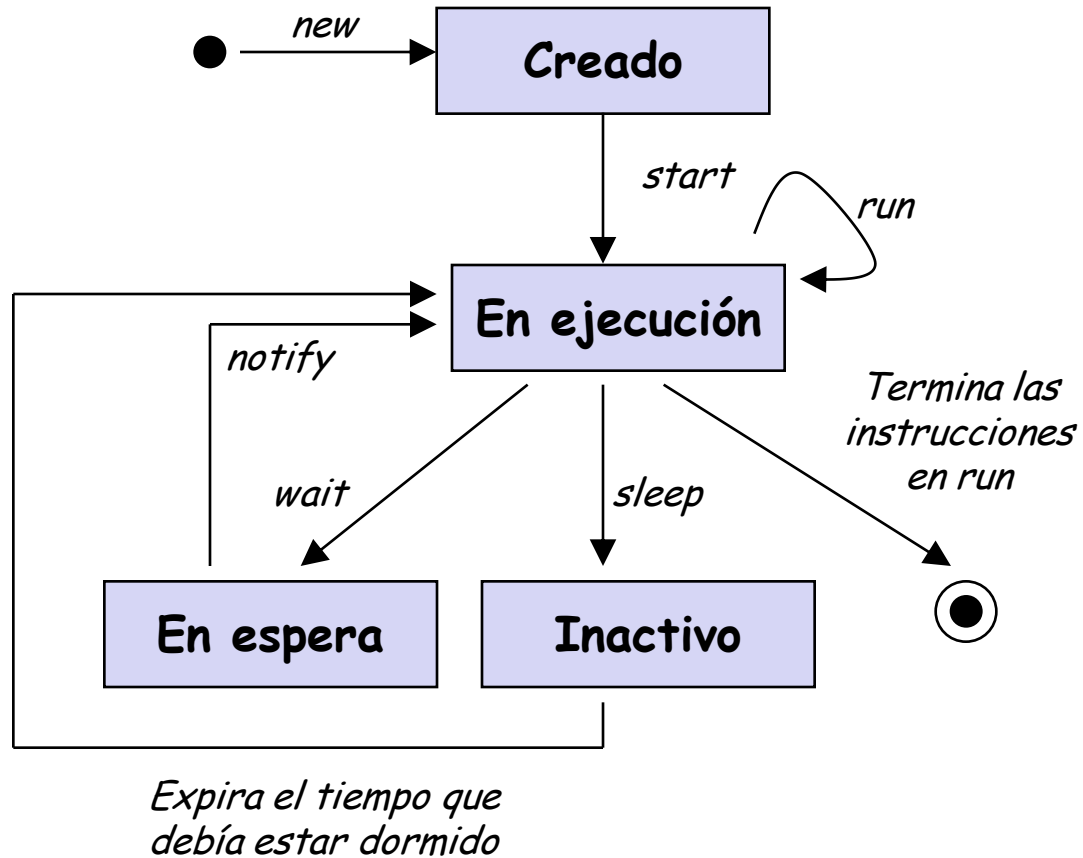
```
C:\ARCHIV-1\Xinox Software\JCreatorV3LE\GE2001.exe
4 factorial es 24
Pi aproximado es 3.2837384837384844
Press any key to continue...
```

Procesos - Hilos

- Los hilos en Java permiten que se ejecuten tareas al mismo tiempo (en paralelo)
- En este caso se tendrían dos hilos de ejecución



Procesos - Hilos



Procesos - Hilos

Thread
start() run() notify() wait() sleep()

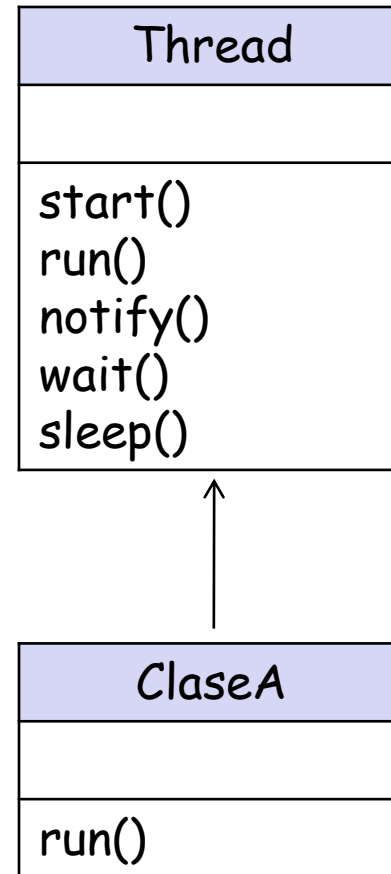
Procesos - Hilos

Hilos en Java por medio de la clase `Thread`

- Desarrollar una clase que **herede** de `Thread` y sobrescribir el método `run` (lo que hace el hilo)

Procesos - Hilos

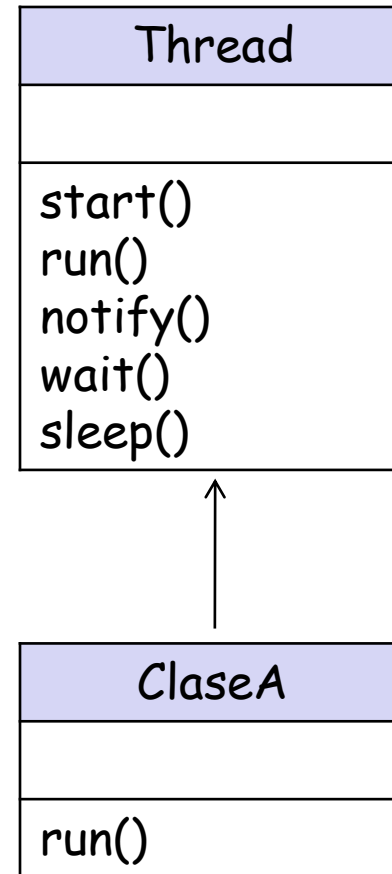
ClaseA hereda de Thread
todos los métodos, se
sobrescribe el método run()
para indicar lo que la ClaseA
va a hacer



Procesos - Hilos

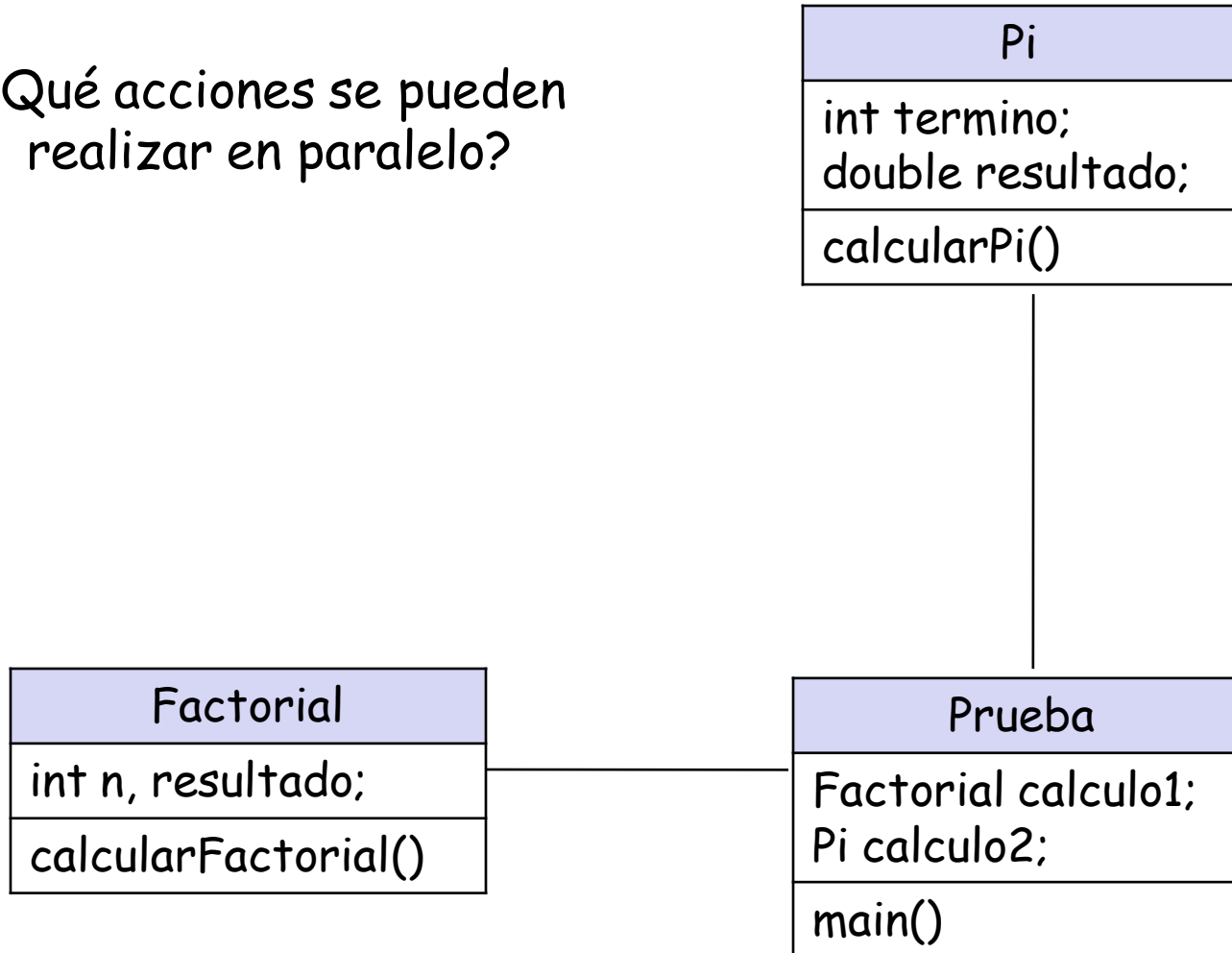
ClaseA hereda de Thread
todos los métodos, se
sobrescribe el método run()
para indicar lo que la ClaseA
va a hacer

```
public class ClaseA extends Thread{  
    ...  
}
```

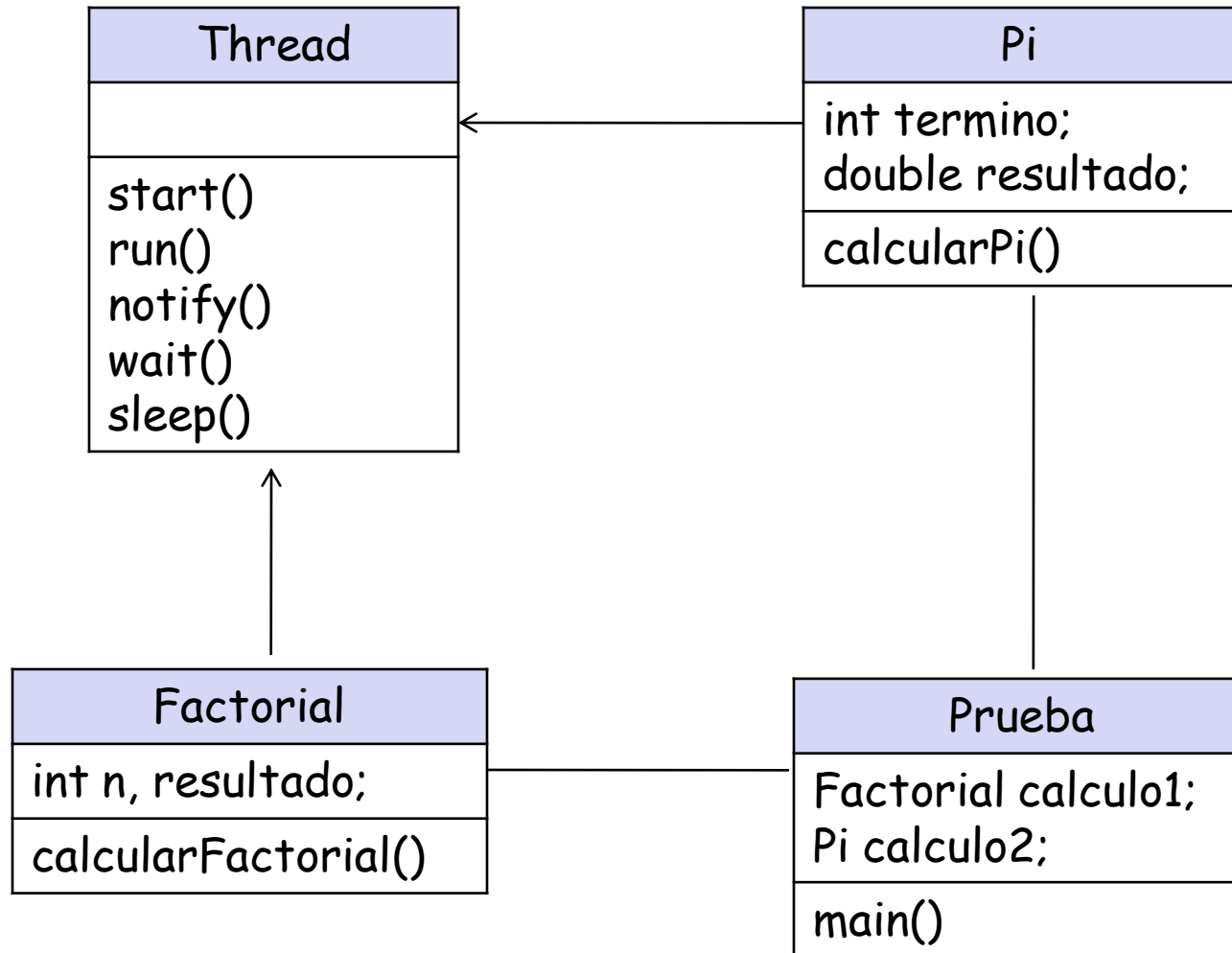


Procesos - Hilos

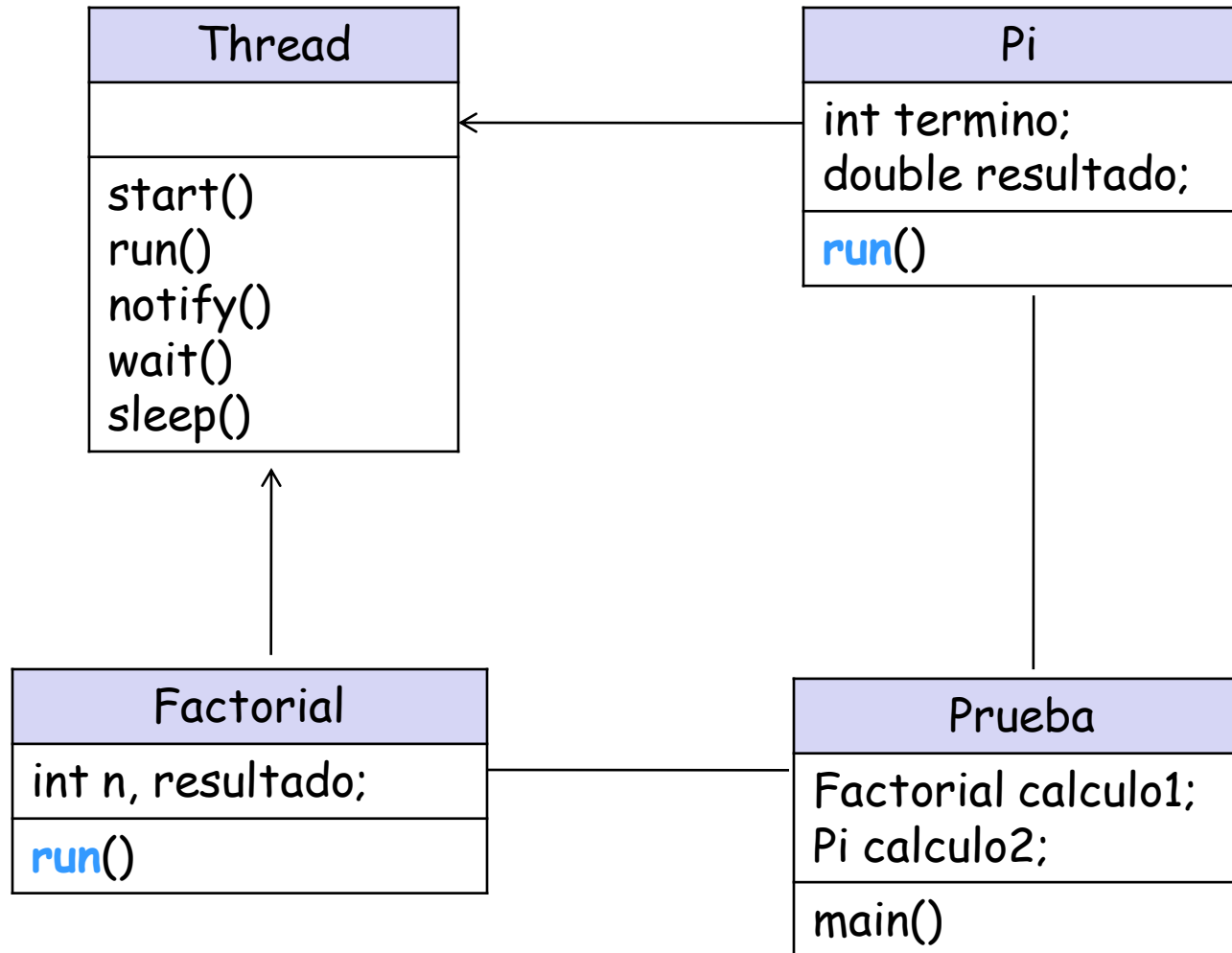
¿Qué acciones se pueden realizar en paralelo?



Procesos - Hilos



Procesos - Hilos

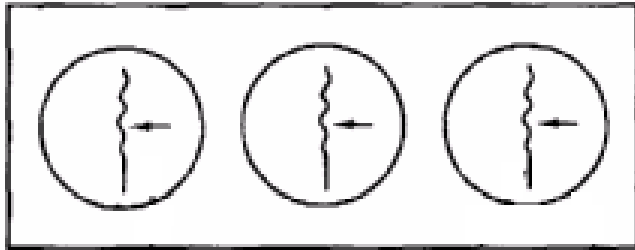


ver Calculos/*.java

Procesos - Hilos

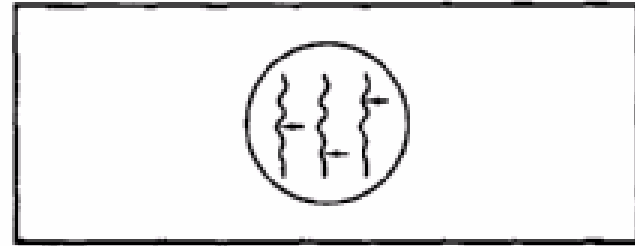
Hilos (Threads)

Computadora



- Tres procesos cada uno con un hilo

Computadora

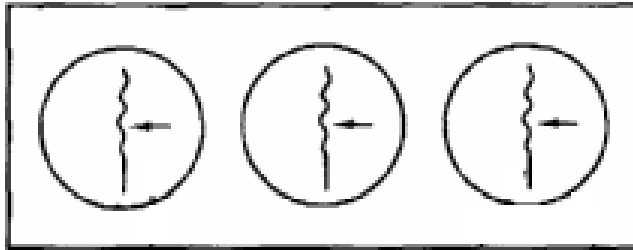


- Un proceso con tres hilos

Procesos - Hilos

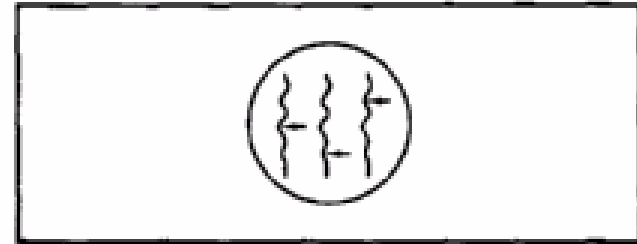
Hilos (Threads)

Computadora



- Tres procesos cada uno con un hilo

Computadora



- Un proceso con tres hilos

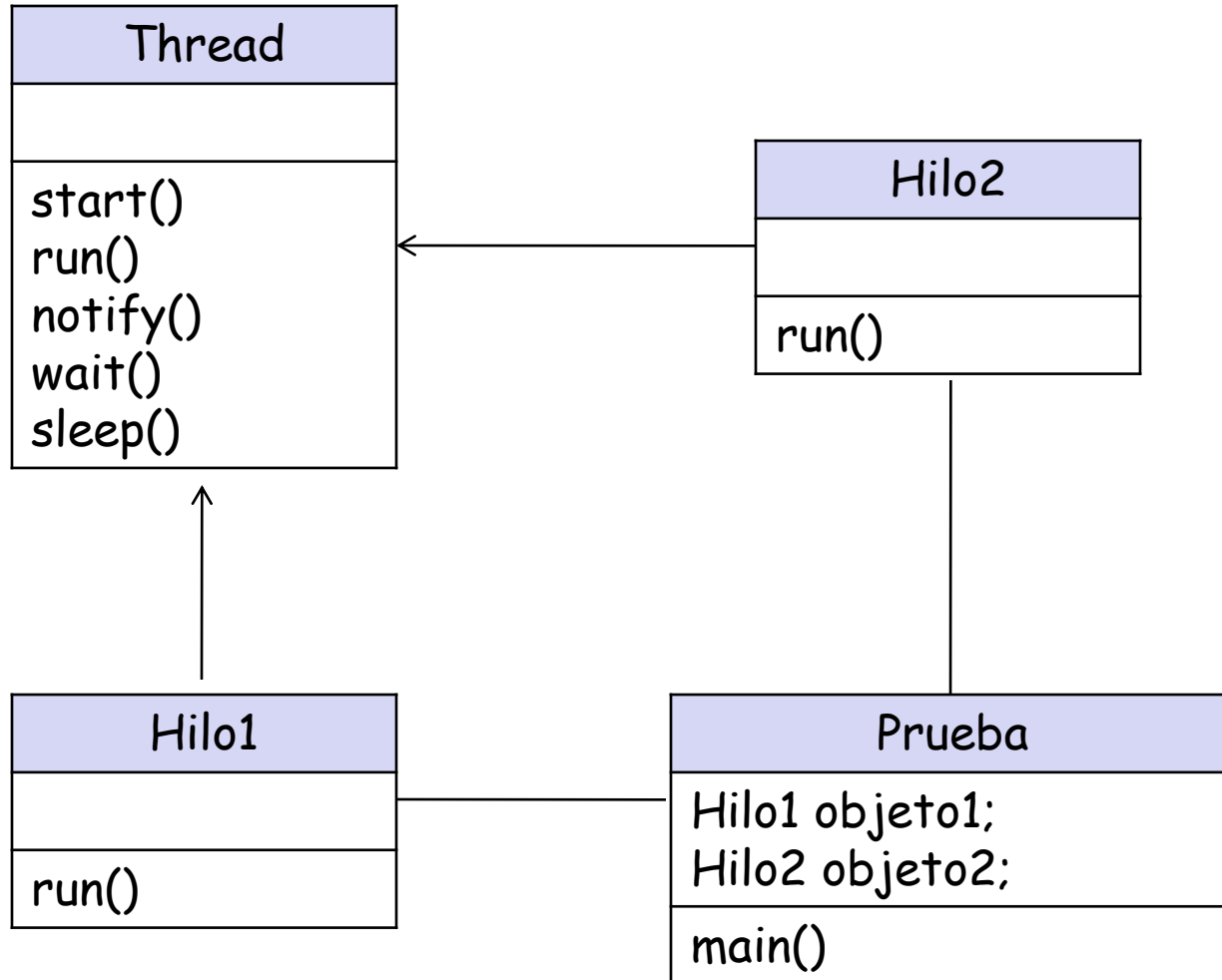
Computadora



- Un proceso con dos hilos:
 - Calcular Pi
 - Calcular Factorial



Procesos - Hilos



Procesos - Hilos

```
public class Hilo1 extends Thread{

    public void run(){
        System.out.println("Mensaje 1 del hilo 1");
        System.out.println("Mensaje 2 del hilo 1");
        System.out.println("Mensaje 3 del hilo 1");
    }

}
```

Procesos - Hilos

```
public class Hilo2 extends Thread{

    public void run(){
        System.out.println("Mensaje 1 del hilo 2");
        System.out.println("Mensaje 2 del hilo 2");
        System.out.println("Mensaje 3 del hilo 2");
    }

}
```

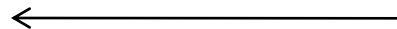
Procesos - Hilos

```
public class PruebaHilos{
```

```
    public static void main(String a[]){
```

```
        Hilo1 objeto1= new Hilo1();
```

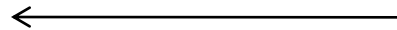
```
        Hilo2 objeto2= new Hilo2();
```



Se crean dos hilos

```
        hilo1.start();
```

```
        hilo2.start();
```

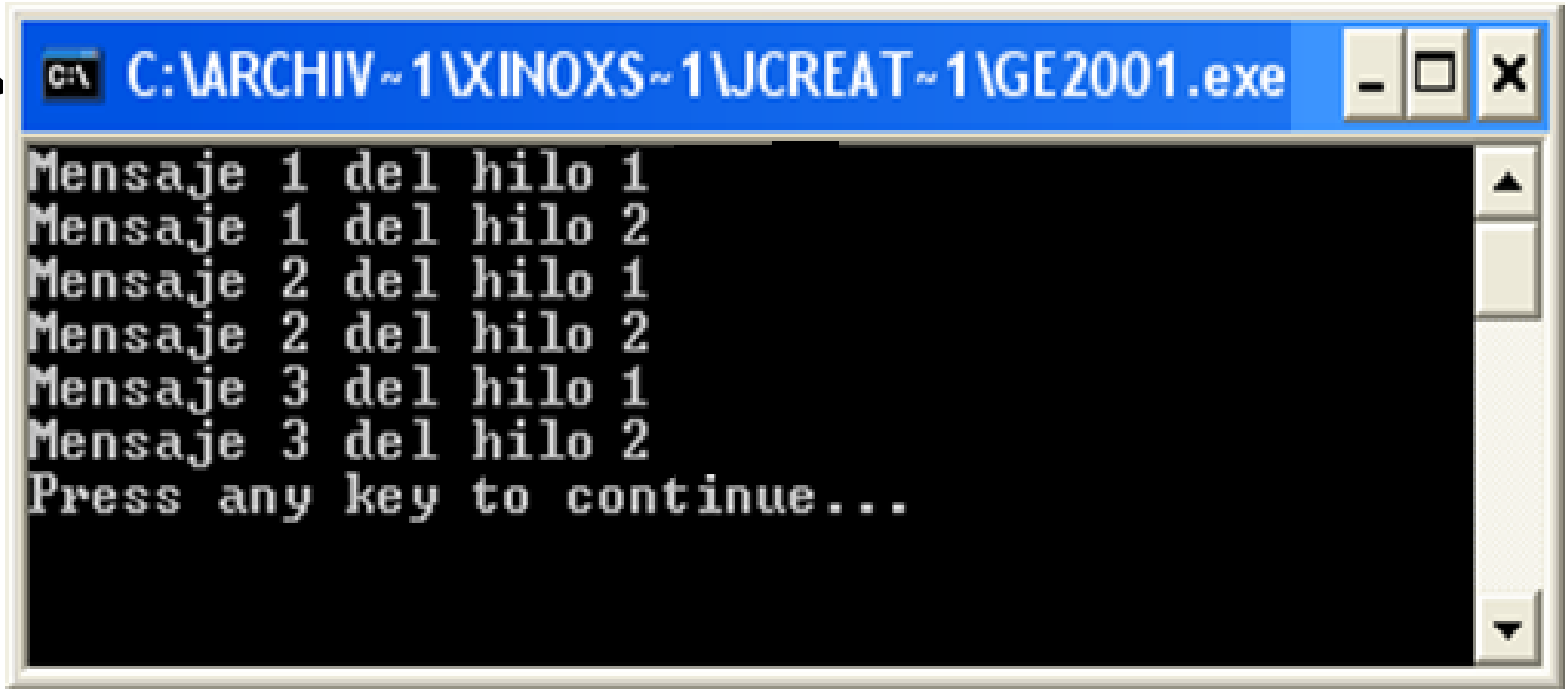


Se inician los hilos

```
    }
```

```
}
```

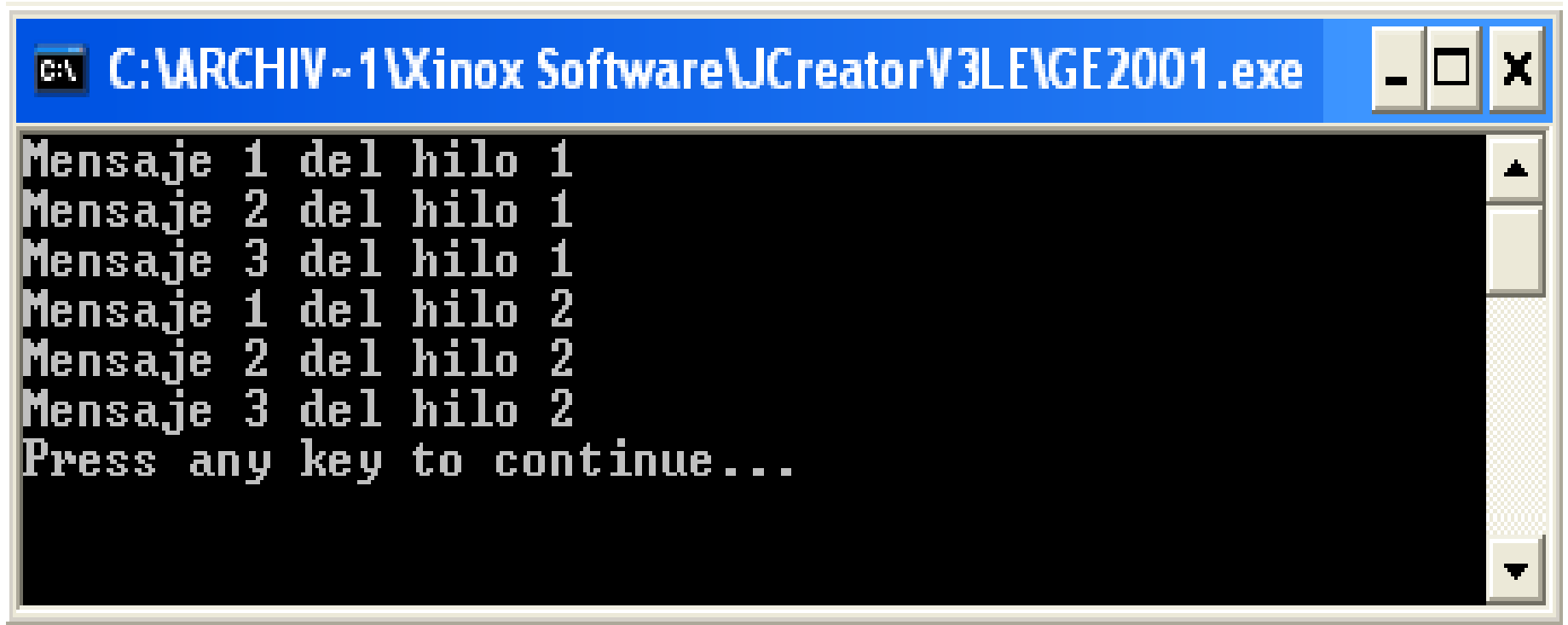
Procesos - Hilos



```
C:\ARCHIV~1\XINOXS~1\JCREAT~1\GE2001.exe
Mensaje 1 del hilo 1
Mensaje 1 del hilo 2
Mensaje 2 del hilo 1
Mensaje 2 del hilo 2
Mensaje 3 del hilo 1
Mensaje 3 del hilo 2
Press any key to continue...
```

Note que no se ejecuta el método
run completo para un hilo y luego
el otro, son procesos
concurrentes!!!

Procesos - Hilos



```
C:\ARCHIV-1\Xinox Software\JCreatorV3\LENGE2001.exe
```

Mensaje 1 del hilo 1
Mensaje 2 del hilo 1
Mensaje 3 del hilo 1
Mensaje 1 del hilo 2
Mensaje 2 del hilo 2
Mensaje 3 del hilo 2
Press any key to continue...

ver Hilos/
Hilo1.java
Hilo2.java
PruebaHilos1.java

Procesos - Hilos

Método `sleep`

- Se especifica la cantidad de milisegundos que estará dormido el hilo
- Se debe manejar con un try-catch una posible excepción

Procesos - Hilos

```
public class Hilo3 extends Thread{
```

```
    public void run(){
```

```
    }
```

```
}
```

Procesos - Hilos

```
public class Hilo3 extends Thread{
```

```
    public void run(){
```

```
        System.out.println("El hilo va a dormir 5 segundos");
```

```
        //Dormir el hilo 5 segundos
```

```
        System.out.println("El hilo despertó");
```

```
    }
```

```
}
```


Procesos - Hilos

```
public class Hilo3 extends Thread{
```

```
    public void run(){
```

```
        System.out.println("El hilo va a dormir 5 segundos");
```

```
        sleep(???);
```

```
        System.out.println("El hilo despertó");
```

```
    }
```

```
}
```

Procesos - Hilos

```
public class Hilo3 extends Thread{  
  
    public void run(){  
  
        System.out.println("El hilo va a dormir 5 segundos");  
  
        sleep(5000);  
  
        System.out.println("El hilo despertó");  
    }  
}
```

Procesos - Hilos

```
public class Hilo3 extends Thread{

    public void run(){

        System.out.println("El hilo va a dormir 5 segundos");
        try{
            sleep(5000);
        }catch(InterruptedException e){ }
        System.out.println("El hilo despertó");
    }
}
```

Procesos - Hilos

```
public class Hilo3 extends Thread{

    public void run(){

        System.out.println("El hilo va a dormir 5 segundos");
        try{
            sleep(5000);
        }catch(InterruptedException e){ }
        System.out.println("El hilo despertó");
    }
}
```

ver Hilos/
-Hilo3.java
-PruebaHilos3.java

Procesos - Hilos

Problema: crear un hilo que esté inactivo una cantidad aleatoria de milisegundos

Procesos - Hilos

- Método Math.random()

Genera un número real (double) en el rango [0.0-1.0)

Procesos - Hilos

- Método Math.random()

$0.0 \leq \text{Math.random()} < 1.0$

0.0, 0.3, 0.5, 0.9, 0.9999

Procesos - Hilos

- Método Math.random()

$0.0 \leq \text{Math.random()} < 1.0$

0.0, 0.3, 0.5, 0.9, 0.9999

```
double numero=Math.random();
```

```
System.out.println("El número es: " + numero);
```


Procesos - Hilos

- Método Math.random()

$0.0 \leq \text{Math.random()} < 1.0$

0.0, 0.3, 0.5, 0.9, 0.9999

$0.0 \leq (\text{Math.random()} * 6) < 6.0$

0.0, 1.8, 3.0, 5.4, 5.9999

Procesos – Hilos

- Método Math.random()

$0.0 \leq \text{Math.random()} < 1.0$

0.0, 0.3, 0.5, 0.9, 0.9999

$0.0 \leq (\text{Math.random()} * 6) < 6.0$

0.0, 1.8, 3.0, 5.4, 5.9999

$0 \leq (\text{int})(\text{Math.random()} * 6) < 6$

0, 1, 3, 5, 5

Procesos – Hilos

- Método Math.random()

$0.0 \leq \text{Math.random()} < 1.0$ 0.0, 0.3, 0.5, 0.9, 0.9999

$0.0 \leq (\text{Math.random()} * 6) < 6.0$ 0.0, 1.8, 3.0, 5.4, 5.9999

$0 \leq (\text{int})(\text{Math.random()} * 6) < 6$ 0, 1, 3, 5, 5

$(\text{int})(\text{Math.random()} * 6)$ genera número entero aleatorio entre 0 y 5

```
int numero=(int) (Math.random() *6);
```

```
System.out.println("El número es: " + numero);
```

Procesos - Hilos

- Cómo generaría números aleatorios entre 0 y 10?

Procesos - Hilos

- Cómo generaría números aleatorios entre 0 y 10?

```
int numero=(int) (Math.random() *11);
```

```
System.out.println("El número es: " + numero);
```

Procesos - Hilos

Problema: crear un hilo que esté inactivo una cantidad aleatoria de milisegundos en el rango [0-19]

Procesos - Hilos

```
public class Hilo4 extends Thread{
```

```
    public void run(){
```

```
        int tiempoDormir= ???
```

```
        System.out.println("El hilo va a dormir " + tiempoDormir + " segundos");
```

```
        try{
```

```
            sleep( ??? );
```

```
        }catch (InterruptedException e){ }
```

```
        System.out.println("El hilo despertó");
```

```
    }
```

```
}
```

Procesos - Hilos

```
public class Hilo4 extends Thread{
```

```
    public void run(){
```

```
        int tiempoDormir=(int)(Math.random()*20);
```

```
        System.out.println("El hilo va a dormir " + tiempoDormir + " segundos");
```

```
        try{
```

```
            sleep( ??? );
```

```
        }catch (InterruptedException e){ }
```

```
        System.out.println("El hilo despertó");
```

```
    }
```

```
}
```


Procesos - Hilos

```
public class Hilo4 extends Thread{
```

```
    public void run(){
```

```
        int tiempoDormir=(int)(Math.random()*20);
```

```
        System.out.println("El hilo va a dormir " + tiempoDormir + " segundos");
```

```
        try{
```

```
            sleep(tiempoDormir*1000);
```

```
        }catch (InterruptedException e){ }
```

```
        System.out.println("El hilo despertó");
```

```
    }
```

```
}
```

Procesos - Hilos

```
public class Hilo4 extends Thread{
```

```
    public void run(){
```

```
        int tiempoDormir=(int)(Math.random()*20);
```

```
        System.out.println("El hilo va a dormir " + tiempoDormir + " segundos");
```

```
        try{
```

```
            sleep(tiempoDormir*1000);
```

```
        }catch (InterruptedException e){ }
```

```
        System.out.println("El hilo despertó");
```

```
    }
```

```
}
```

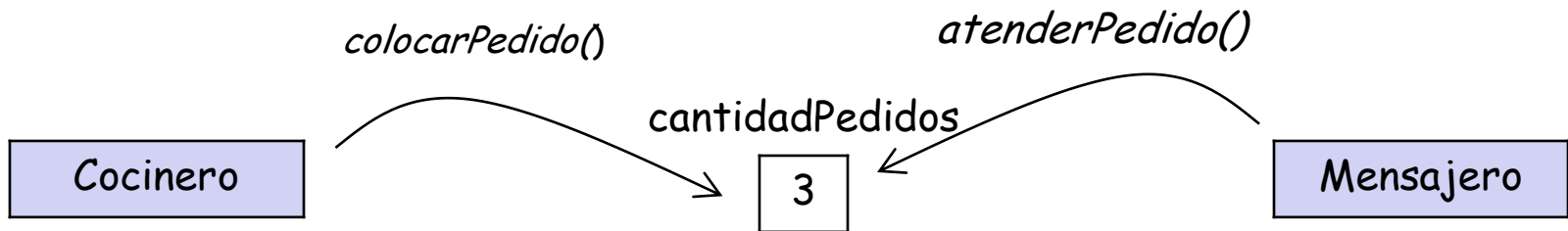
ver Hilos/

-Hilo4.java

-PruebaHilos4.java

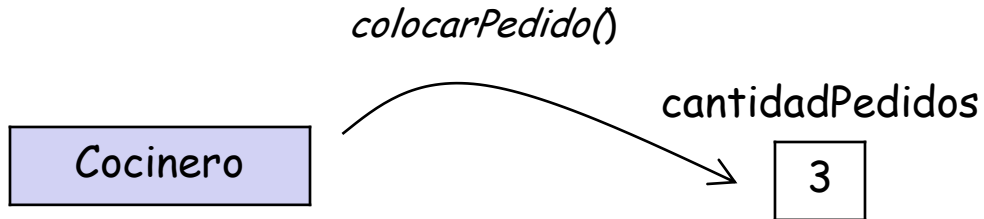
Procesos - Hilos

Problema del restaurante Chino



Procesos - Hilos

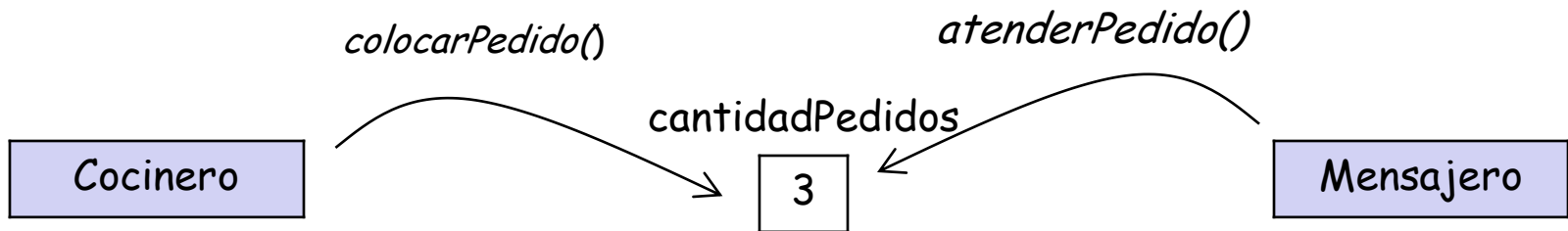
Problema del restaurante Chino



- El proceso Cocinero siempre coloca de uno en uno los pedidos en la mesa
- Se aumenta en 1 la cantidad de pedidos que actualmente hay en la mesa por atender

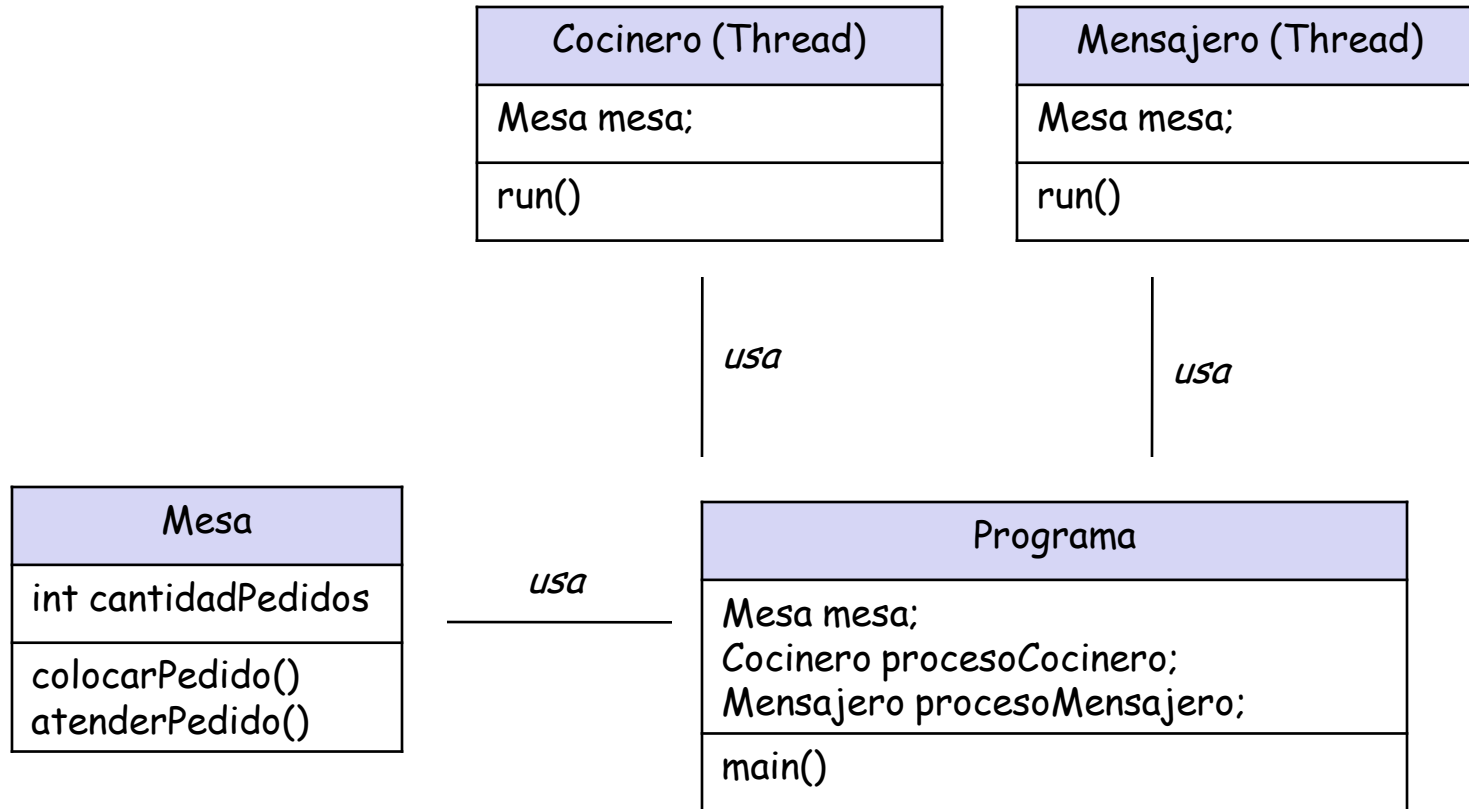
Procesos - Hilos

Problema del restaurante Chino



El proceso Mensajero intenta tomar 2 pedidos. Si solo hay 1 ó 0 espera hasta que por lo menos hayan dos

Procesos - Hilos



```
public class Mesa{  
    int cantidadPedidos;  
  
    public Mesa(){  
        cantidadPedidos=0;  
    }  
  
    public void colocarPedido(){  
    }  
  
    public void atenderPedido(){  
    }  
  
}
```



```
public class Mesa{  
    int cantidadPedidos;  
  
    public Mesa(){  
        cantidadPedidos=0;  
    }  
  
    public void colocarPedido(){  
        cantidadPedidos = cantidadPedidos+1;  
    }  
  
    public void atenderPedido(){  
  
    }  
  
}
```

```
public class Mesa{  
    int cantidadPedidos;  
  
    public Mesa(){  
        cantidadPedidos=0;  
    }  
  
    public void colocarPedido(){  
        cantidadPedidos = cantidadPedidos+1;  
    }  
  
    public void atenderPedido(){  
        cantidadPedidos = cantidadPedidos-2;  
    }  
  
}
```

```
public class Mesa{  
    int cantidadPedidos;  
  
    public Mesa(){  
        cantidadPedidos=0;  
    }  
  
    public void colocarPedido(){  
        cantidadPedidos = cantidadPedidos+1;  
    }  
  
    public void atenderPedido(){  
        while ( condicionEspera )  
            ;  
        cantidadPedidos = cantidadPedidos-2;  
    }  
}
```

```
public class Mesa{  
    int cantidadPedidos;  
  
    public Mesa(){  
        cantidadPedidos=0;  
    }  
  
    public void colocarPedido(){  
        cantidadPedidos = cantidadPedidos+1;  
    }  
  
    public void atenderPedido(){  
        while (cantidadPedidos==0 || cantidadPedidos==1)  
            ;  
        cantidadPedidos = cantidadPedidos-2;  
    }  
}
```

```
public class Mesa{  
    int cantidadPedidos;  
  
    public Mesa(){  
        cantidadPedidos=0;  
    }  
  
    public void colocarPedido(){  
        cantidadPedidos = cantidadPedidos+1;  
    }  
  
    public void atenderPedido(){  
        while (cantidadPedidos<2)  
            ;  
        cantidadPedidos = cantidadPedidos-2;  
    }  
}
```

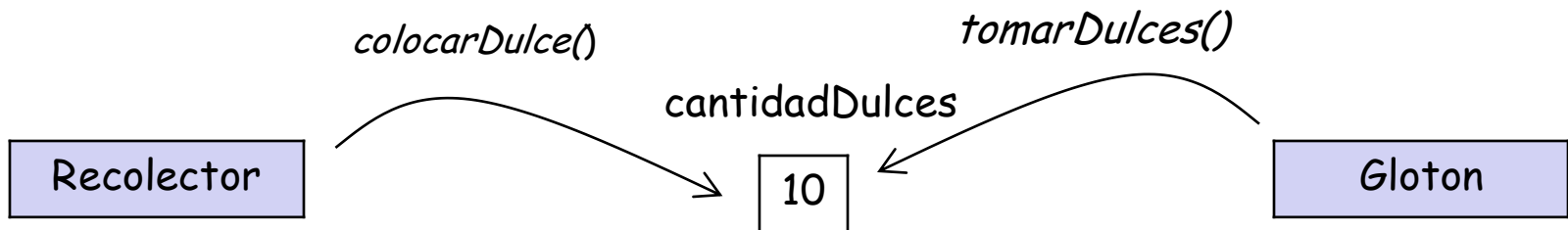
```
public class Cocinero extends Thread{  
    Mesa mesa;  
    public Cocinero(Mesa m){  
        mesa=m;  
    }  
    public void run(){  
  
        int tiempoCocinando=(int)(Math.random()*10);  
        System.out.println("Cocinero preparando comida por " + tiempoCocinando + " segundos");  
        try{  
            sleep(tiempoCocinando*1000);  
        }catch(InterruptedException e){}  
        System.out.println("Cocinero trae un nuevo pedido a la mesa");  
        mesa.colocarPedido();  
  
    }  
}
```

```
public class Cocinero extends Thread{
    Mesa mesa;
    public Cocinero(Mesa m){
        mesa=m;
    }
    public void run(){
        while(true){
            int tiempoCocinando=(int)(Math.random()*10);
            System.out.println("Cocinero preparando comida por " + tiempoCocinando + " segundos");
            try{
                sleep(tiempoCocinando*1000);
            }catch(InterruptedException e){}
            System.out.println("Cocinero trae un nuevo pedido a la mesa");
            mesa.colocarPedido();
        }
    }
}
```

ver Restaurante/*.java

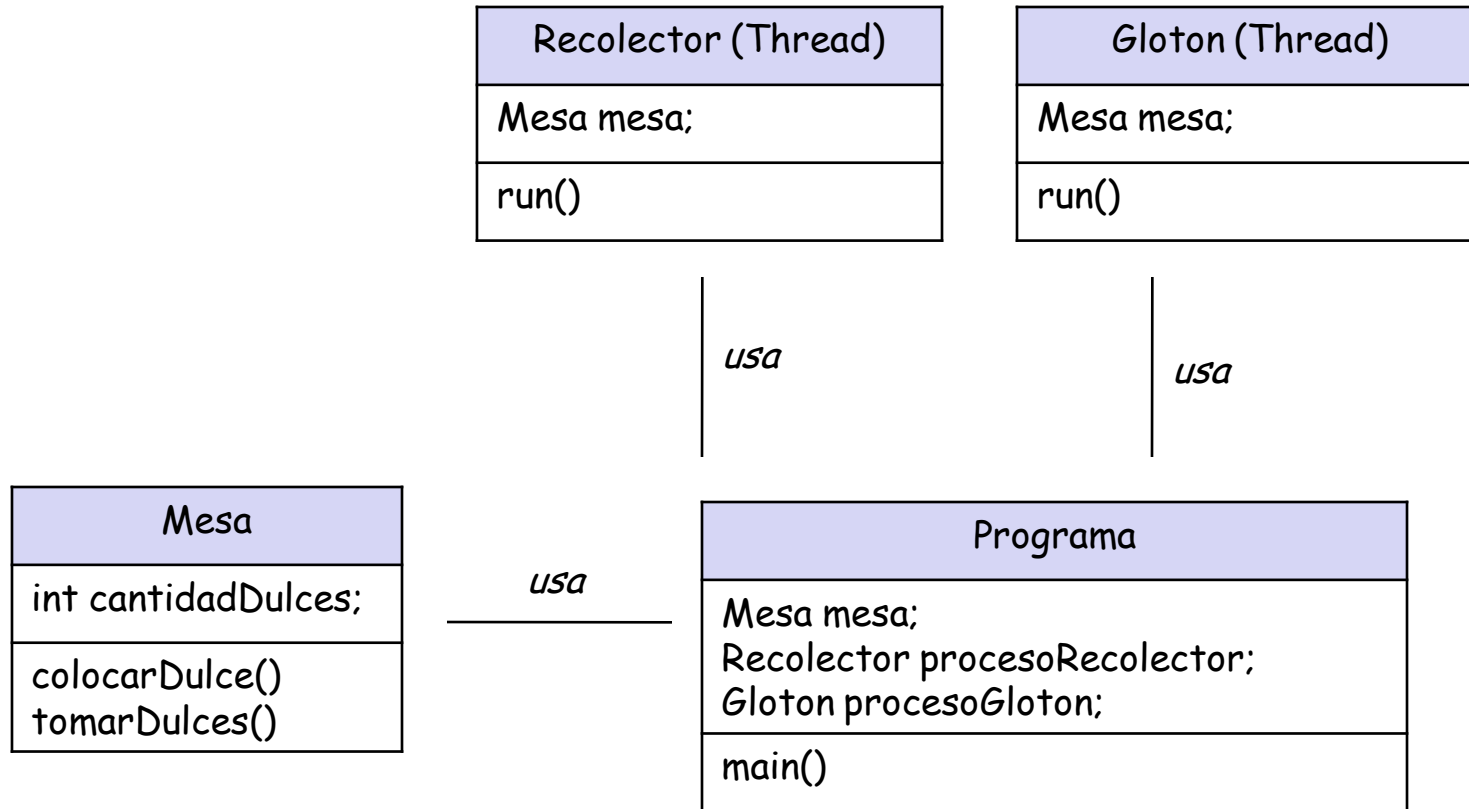
Procesos - Hilos

Problema del recolector y el glotón



- El Recolector siempre coloca de a un dulce en la mesa
- En la mesa caben máximo 100 dulces
- El Glotón toma siempre 5 dulces, si no hay suficientes debe esperar

Procesos - Hilos

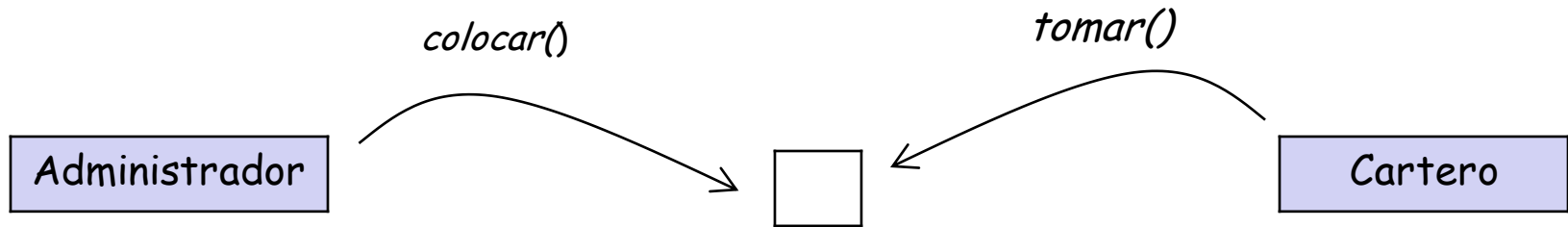


```
public class Mesa{  
    int cantidadDulces;  
  
    public Mesa(){  
        cantidadDulces =0;  
    }  
  
    public void colocarDulce(){  
        while (cantidadDulces==100)  
            ;  
        cantidadDulces=cantidadDulces+1;  
    }  
    public void tomarDulces(){  
        while (cantidadDulces<5)  
            ;  
        cantidadDulces=cantidadDulces-5;  
    }  
}
```

```
public class Recolector extends Thread{  
    Mesa mesa;  
    public Recolector(Mesa m){  
        mesa=m;  
    }  
    public void run(){  
        while(true){  
            int tiempoRecogiendo=(int)(Math.random()*10);  
            System.out.println("Recolector va a recoger dulce por " + tiempoRecogiendo + " segundos");  
            try{  
                sleep(tiempoRecogiendo*1000);  
            }catch(InterruptedException e){}  
            System.out.println("Recolector trae un dulce a la mesa");  
            mesa.colocarDulce();  
        }  
    }  
}
```

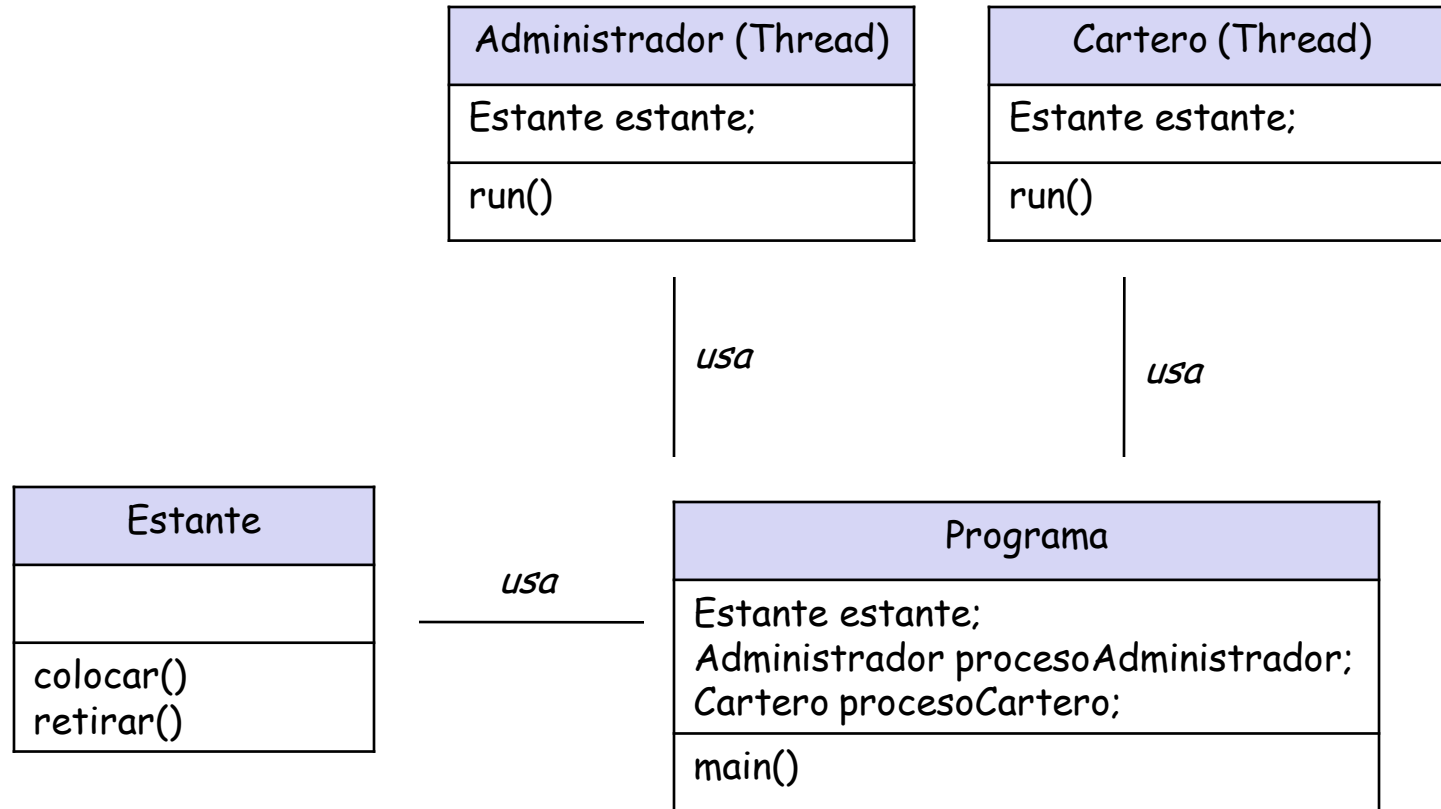
Procesos - Hilos

Problema de la oficina de correo



- Se tiene un estante con capacidad máxima para 50 cartas
- El Administrador siempre coloca 4 cartas, espera si no caben
- El Cartero toma siempre 12 cartas, espera si no las hay

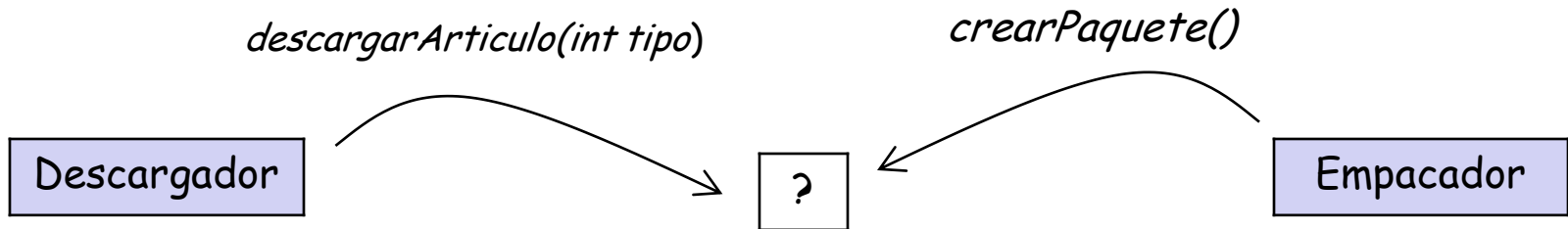
Procesos - Hilos



```
public class Estante{  
    int cantidadCartas;  
  
    public Estante(){  
        cantidadCartas=0;  
    }  
  
    public void colocar(){  
        while (cantidadCartas>46)  
            ;  
        cantidadCartas=cantidadCartas+4;  
    }  
    public void tomar(){  
        while (cantidadCartas<12)  
            ;  
        cantidadCartas=cantidadCartas-12;  
    }  
}
```

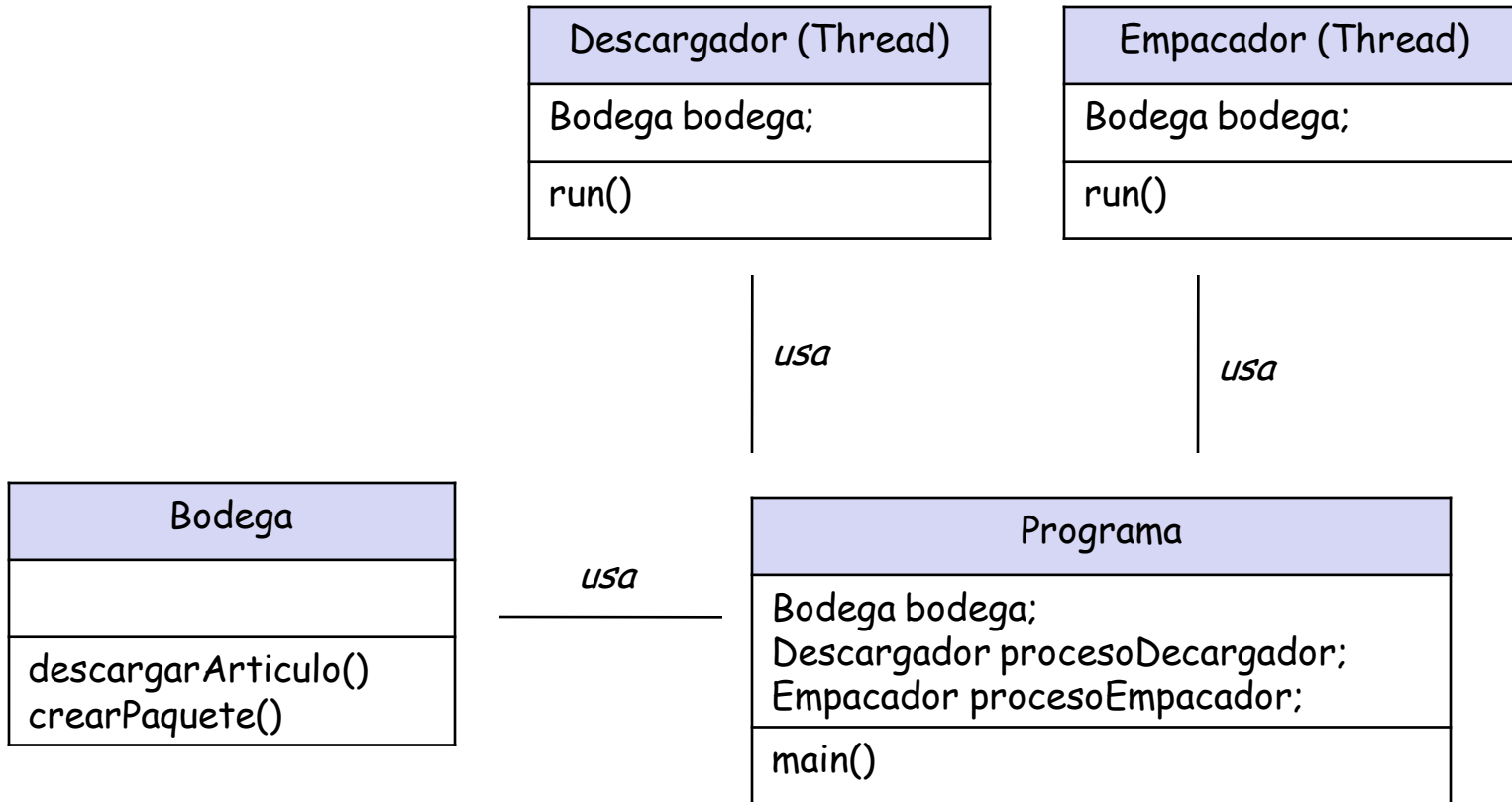
Procesos - Hilos

Problema de la bodega



- La capacidad de la bodega es de 200m^3
- En la bodega se pueden colocar dos tipos de artículos, tipo1 de volumen 10m^3 y tipo 2 de 15m^3
- Cada vez se coloca un solo artículo, puede ser *descargarArticulo(1)* o *descargarArticulo(2)*
- Para crear un paquete el empacador necesita 3 artículos tipo 1 y 4 tipo 2, si no los hay debe esperar

Procesos - Hilos




```
public class Bodega{
```

```
    ???
```

```
    public Bodega(){
```

```
        ???
```

```
    }
```

```
    public void descargarArticulo(int tipo){
```

```
        if (tipo==1){
```

```
            ???
```

```
        }
```

```
        if (tipo==2){
```

```
            ???
```

```
        }
```

```
    }
```

```
    public void crearPaquete(){
```

```
    }
```

```
}
```