

- 3 pt. 5. En la mayoría de los sistemas operativos existen dos funciones muy útiles para sincronizar hilos (o procesos) concurrentes: wait y signal, que operan sobre estructuras de datos conocidas como semáforos. La forma de operar es la siguiente: un hilo que ejecuta un wait(sem) -- donde semáforo es un objeto del tipo semáforo-- sólo podrá seguir ejecutando si el valor del semáforo es 0 o más. En otro caso, se suspende hasta que otro hilo lo despierte a través de un signal(sem) sobre el mismo objeto semáforo.

Muchas veces los alumnos de POO han manifestado que la programación de hilos en Java sería mucho más fácil si se pudiera disponer de esta funcionalidad. Atendiendo vuestras quejas, hemos decidido pedirlos que implementéis una clase Semaforo que debe proveer dos métodos públicos, wait y signal, con la siguiente interfaz:

```
// Decrementa en uno el valor del semáforo. Si el nuevo valor es
// menor que 0, el hilo se suspende hasta que otro hilo
// ejecute un signal sobre este semáforo.
// Si el valor es mayor o igual que 0, retorna inmediatamente.

public void wait();

// Incrementa en uno el valor del semáforo. Si el nuevo valor es
// menor o igual que 0, se despierta a un hilo de los que está
// suspendido esperando.

public void signal();
```

- 3 pt. 6. Supón el siguiente fragmento de código perteneciente a una clase *Cliente*:

```
...
ClaseA arg = new ClaseA();
ServicioRemoto sr =
    (ServicioRemoto) Naming.lookup("//ordenador.eps.uam.es/servicio1");
sr.servicio(arg);
...
```

Describe las alternativas de implementación que existen para la clase ClaseA para que este código funcione, y cuándo se debe utilizar cada una.