# Práctica 1 - CDS



## Clases de Interés

- Pool: incluye el método para inicilizar los parámetros de la piscina y las operaciones que invoca un nadador cuando quiere un cambio de estado.

```
public abstract void init(int ki, int cap);
public abstract void  kidSwims() throws InterruptedException;
public abstract void kidRests() throws InterruptedException;
public abstract void instructorSwims() throws InterruptedException;
public abstract void instructorRests() throws InterruptedExcpetion;
```

- Log: operaciones que hay que invocar cuando hay un cambio de estado de un nadador para que la aplicación pueda mostrar el estado correcto de los andadores.

```
public void waitingToSwim()
public void swimming()
public void waitingToRest()
public void resting()
```

## Preguntas Piscina Tipo 0 (baño libre)

1. **Compruebe si se ha utilizado la palabra synchronized al implementar las operaciones de la piscina del tipo 0 (kidSwims, kidRests, instructorSwims, instructorRests). ¿Se observaría alguna diferencia de ejecución entre utilizar dicha palabra synchronized o no utilizarla?**

No se ha utilizado la palabra synchronized. Si se utilizara synchronized no se podría ejecutar ningún método de la clase a la vez que otro.

2. **¿Se pueden producir condiciones de carrera? ¿Por qué?**

No se pueden producir condiciones de carrera porque no hay variables compartidas entre los métodos.

3. **¿Cómo se ha representado el estado interno de la piscina? ¿Por qué así?**

Con una tupla del número actual de niños dentro de la piscina y el número actual de instructores dentro de la piscina.

4. **En la piscina de tipo 0,  ¿qué transiciones de estados se pueden producir? ¿Pueden los niños e instructores pasar de “resting” a “swimming” directamente, sin pasar por estados intermedio? ¿Por qué?**

Se producen transiciones directas de resting a swimming porque no hay ninguna restricción que implique la necesidad de esperar a entrar o salir de la piscina.

## Actividad 1 (Pool 1)

### Ejercicio 1.1

Indique cómo representar el estado de la piscina de tipo 1 para poder cumplir las normas de uso asociadas a dicha piscina. Para ello, complete la tabla correpondiente a Pool1:

| Pool1 | hay que esperar si… | modifica estado… | avisa a… |
|---|---|---|---|
| kidSwims() | No hay instructores nadando | waitingToSwim() | No avisa |
| kidRests() | No espera nunca | No hay estado | Avisa a instructorRests() |
| instructorSwims() | No espera nunca | No hay estado | Avisa a kidSwims() |
| instructorRests() | Espera si hay algún niño dentro de la piscina | waitingToRest() | No avisa |

### Ejercicio 1.2

Modifique el método kidSwims de la clase Pool1 para que el hilo invocante espere si no hay instructores nadando (tal y como se indica en la tabla, y siguiendo el esquema anteriormente descrito). Utilice las variables internas que considere oportunas para contabilidar los niños e instructores sque entran/salen de la piscina. Si es necesario, modifique otros métodos de Pool1 para actualizar dichas variables.

```
public class Pool1 extends Pool {   //no kids alone
    private int kids = 0, instructors = 0;
    public void init(int ki, int cap) {}
    //public boolean ok(int nk, int ni){return nk == 0 || ni >= 0;} //pueden entrar niños si no hay niños o si hay más de un instructo
    public synchronized void kidSwims() throws InterruptedException{
        while(instructors == 0){ //espera a que haya al menos un instructor en la piscina para que entren los niños
            log.waitingToSwim(); //para visualizar la posición del nadador
            wait();
        }
        kids++;
        notifyAll();
        log.swimming();
        //log.notifyAll();
    }
    public synchronized void kidRests(){
        kids--;
        notifyAll();
        log.resting();
    }
    public synchronized void instructorSwims(){
        instructors++;
        notifyAll();
        log.swimming();
    }
    public synchronized void instructorRests() throws InterruptedException{
         while(instructors == 1 && kids > 0){
            log.waitingToRest();
            wait();
        }
        instructors--;
        notifyAll();
        log.resting();
    }
}
```

### Ejercicio 1.3

Analice la traza de la ejecución de determine qué problemas o situaciones ilegales aparecen y qué situaciones sí que se han resuelto.

Ya no entran niños si no hay instructores pero se sigue superando la capacidad de la piscina y la cantidad de niños por instructor.

### Ejercicio 1.4

Revise el resto de métodos de la clase Pool1 (es decir, kidRests, instructorSwims, instructorRests) y modifique aquellos que sea necesario, para reflejar las entradas/salidas a la piscina de los niños e instructores de forma apropiada.

Hay que tener en cuenta que los instructores también tienen que esperar a que no haya niños en la piscina para salirse.

### Ejercicio 1.5

Compruebe que las reglas de la piscina tipo 1 se cumplen ahora. Repita la ejecución modificando el número de niños y/o instructores. Comrpuebe que la ejecución continúa siendo correcta.

## Preguntas Piscina Tipo 1 (los niños no pueden nadar solos)

1. Compruebe si se ha utlizado la palabra synchronized al implementar las operaciones de la piscina de tipo 1. ¿Se observaría alguna diferencia de ejecución entre utilizar dicha palabra synchronized o no utilizarla?

Sí que se utiliza, de no ser así podría dar errores debido a las condiciones de carrear provocadas por sus variables compartidas.

2. ¿Se pueden producir condiciones de carrera? ¿Por qué?

Sí, porque los métodos modifican kids e instructor que son variables compartidas entre los hilos.

3. ¿Cómo se ha representado el estado interno de la piscina? ¿Por qué así?

Con una tupla del número actual de niños dentro de la piscina y el número actual de instructores dentro de la piscina.

4. Para avisar al nuevo estado, se ha empleado notifyAll(). ¿Podríamos haber utilizado simplemente notify()? ¿Por qué?

No se podría haber utilizado notify() porque solo avisa a un hilo no a todos los que están en espera.

5. ¿Al final qué métodos de Pool1 ha empleado notifyAll()? ¿Lo ha utilizado al final de todos los métodos? Si es así, analice si es obligatorio emplearlo para todos los métodos, o si sería posible (o más eficiente) utilizarlo solamente en unos cuantos (en los requeridos).

Se ha usado al final de todos pero podría haber sido evitado tanto en kidSwims como en instructorRests.

## Actividad 2 (Pool2)

1. Indique cómo representar el estado de la piscina para poder cumplir las  normas de uso indicadas para la piscina tipo 2 (recuerde que también debe cumplir con las normas de la piscina tipo 1). Para ello, rellene la tabla correspondiente a Pool2:

| Pool2 | hay que esperar si… | modifica estado… | avisa a… |
|---|---|---|---|
| kidSwims() | No hay instructores nadando o ya hay el máximo de niños permitido por instructor en la piscina | waitingToSwim() | No avisa |
| kidRests() | No espera nunca | No hay estado | Avisa a instructorRests() |
| instructorSwims() | No espera nunca | No hay estado | Avisa a kidSwims() |
| instructorRests() | Espera si hay algún niño dentro de la piscina o si hay más niños que el máximo permitido si se va uno de los instructores | waitingToRest() | No avisa |

2. Modifique el código de Pool2 para que se cumplan las reglas de utilización de la piscina tipo 2.

```
public class Pool2 extends Pool{ //max kids/instructor
    private int kids = 0, instructors = 0;
    private int maxKids;
    public void init(int ki, int cap){
        maxKids = ki;
    }
    public synchronized void kidSwims() throws InterruptedException{
        while(instructors == 0 || kids >= maxKids * instructors){
            //espera a que haya al menos un instructor en la piscina para que entren los niños
            log.waitingToSwim(); //para visualizar la posición del nadador
            wait();
        }
        kids++;
        log.swimming();
        //log.notifyAll();
    }
```

```
    public synchronized void kidRests(){
        log.resting();
        kids--;
        notifyAll();
    }
    public synchronized void instructorSwims(){
        log.swimming();
        instructors++;
        notifyAll();
    }
    public synchronized void instructorRests() throws InterruptedException{
         while(kids > maxKids * (instructors - 1)){
            log.waitingToRest();
            wait();
        }
        instructors--;
        notifyAll();
        log.resting();
    }
}
```

3. Verifique el funcionamiento de Pool2 modificando el número de niños y/o instructores.

## Preguntas Piscina Tipo 2 (máximo niños por instructor)

1. Para representar el estado del objeto compartido, ¿bastaría con utilizar una variable de tipo entero (ej. nSwimKids) y una variable de tipo booleano (ej. instInPool)? ¿Por qué?

Basta con  usar una variable que indique el máximo de niños y calcular en la guarda cuántos serían dependiendo del número de instructores. También se podría usar un método auxiliar con una variable booleana.

2. ¿A qué métodos afecta esta nueva regla? (es decir, ¿en qué métodos de Pool2 ha realizado modificaciones?)

Afecta a kidSwims y a instructorRests.

3. cuando un instructor entra en la piscina Pool2, ¿se debe invocar notifyAll()? ¿Por qué?

Si, porque puede haber niños esperando que ahora si puedan entrar al haber aumentado el número máximo de niños permitido en la piscinal

## Actividad 3 (Pool3)

1. Indique cómo representar el estado de la piscina de tipo 3 para poder cumplir con todas sus normas de uso. Para ello, relllene la tabla correspondiente a Pool3.

| Pool3 | hay que esperar si… | modifica estado… | avisa a… |
|---|---|---|---|
| kidSwims() | No hay instructores nadando o ya hay el máximo de niños permitido por instructor en la piscina o la piscina está llena | waitingToSwim() | No avisa |
| kidRests() | No espera nunca | No hay estado | Avisa a instructorRests(), a kidsSwims() y a instructorSwims() |
| instructorSwims() | Espera si la piscina ya está llena | No hay estado | Avisa a kidSwims() |
| instructorRests() | Espera si hay algún niño dentro de la piscina o si hay más niños que el máximo permitido si se va uno de los instructores | waitingToRest() | Avisa a kidsSwims() y a instructorSwims() |

2. Modifique el código de Pool3 para que se cumplan las reglas de utilización de la piscina tipo 3.

```
public class Pool3 extends Pool{ //max kids/instructor
    private int kids = 0, instructors = 0;
    private int maxKids, maxCap;
    public void init(int ki, int cap){
        maxKids = ki;
```

```
        maxCap = cap;
    }
    public synchronized void kidSwims() throws InterruptedException{
        while(instructors == 0 || kids >= maxKids * instructors || (instructors + kids) >= maxCap){
            //espera a que haya al menos un instructor en la piscina para que entren los niños
            log.waitingToSwim(); //para visualizar la posición del nadador
            wait();
        }
        kids++;
        log.swimming();
        //log.notifyAll();
    }
    public synchronized void kidRests(){
        log.resting();
        kids--;
        notifyAll();
    }
    public synchronized void instructorSwims() throws InterruptedException{
        while((instructors + kids) >= maxCap){
            log.waitingToSwim();
            wait();
        }
        log.swimming();
        instructors++;
        notifyAll();
    }
    public synchronized void instructorRests() throws InterruptedException{
         while(kids > maxKids * (instructors - 1)){
            log.waitingToRest();
            wait();
        }
        instructors--;
        notifyAll();
        log.resting();
    }
}
```

3. Verifique el funcionamiento de Pool3 modificando el número de niños y/o instructores.

## Preguntas Piscina Tipo 3 (capacidad máxima)

1. Para representar el estado de la piscina, ¿se requiere añadir alguna obra variable al estado de la piscina respecto a la implementación de Pool2? ¿Cuál? ¿Por qué?

Si, se añade una variable para saber el máximo de personas permitidas en la piscina que después se comparará en las guardas con la suma del número de instructores y niños que hay en ese momento.

2. ¿Podemos decir que la piscina Pool3 está actuando como un “monitor”? ¿Y las piscinas anteriores?

Si, porque los hilos esperan a que se cumplan ciertas condiciones para reactivarse. Todas excepto la piscina 0 funcionan como un monitor.

## Actividad 4 (Pool4)

1. Indique cómo representar el estado de la piscina de tipo 4 para poder cumplir con todas sus normas de uso. Para ello, rellene la tabla correspondiente a Pool4:

| Pool3 | hay que esperar si… | modifica estado… | avisa a… |
|---|---|---|---|
| kidSwims() | No hay instructores nadando, ya hay el máximo de niños permitido por instructor en la piscina, la piscina está llena o hay algún instructor esperando para poder salir | waitingToSwim() | No avisa |
| kidRests() | No espera nunca | No hay estado | Avisa a instructorRests(), a kidsSwims() y a instructorSwims() |
| instructorSwims() | Espera si la piscina ya está llena | No hay estado | Avisa a kidSwims() |

| Pool3 | hay que esperar si… | modifica estado… | avisa a… |
|---|---|---|---|
| instructorRests() | Espera si hay algún niño dentro de la piscina o si hay más niños que el máximo permitido si se va uno de los instructores | waitingToRest() | Avisa a kidsSwims() y a instructorSwims() |

2. Modifique el código de Pool4 para que se cumplan las reglas de utilización de la piscina tipo 4.

```
public class Pool4 extends Pool{ //max kids/instructor
    private int kids = 0, instructors = 0;
    private int maxKids, maxCap;
    private boolean insWait;
    public void init(int ki, int cap){
        maxKids = ki;
        maxCap = cap;
    }
    public synchronized void kidSwims() throws InterruptedException{
        while(instructors == 0 || kids >= maxKids * instructors || (instructors + kids) >= maxCap||!insWait){
            //espera a que haya al menos un instructor en la piscina para que entren los niños
            log.waitingToSwim(); //para visualizar la posición del nadador
            wait();
        }
        kids++;
        log.swimming();
        //log.notifyAll();
    }
    public synchronized void kidRests(){
        log.resting();
        kids--;
        notifyAll();
    }
    public synchronized void instructorSwims() throws InterruptedException{
        while((instructors + kids) >= maxCap){
            log.waitingToSwim();
            wait();
        }
        log.swimming();
        instructors++;
        notifyAll();
    }
    public synchronized void instructorRests() throws InterruptedException{
         while(kids > maxKids * (instructors - 1)){
            log.waitingToRest();
            wait();
            insWait = true;
        }
        instructors--;
        insWait = false;
        notifyAll();
        log.resting();
    }
}
```

3. Verifique el funcionamiento de Pool4 modificando el número de niños y/o instructores.

## Preguntas Piscina Tipo 4 (si instructores esperan, no pueden entrar niños)

1. Para representar el estado de la piscina, ¿se requiere añadir alguna otra variable al estado de la piscina respecto a la implementación de Pool3? ¿Cuál? ¿Por qué?

Se tiene que añadir una variable booleana que indique si hay instructores esperando para descansar o no.

2. ¿Al final de qué métodos de Pool4 ha empleado el notifyAll()? ¿Lo ha utilizado al final de todos los métodos? Si es así, analice si es obligatorio emplearlo para todos los métodos, o si sería posible (o más eficiente) utilizarlo solamente en unos cuantos (en los requeridos).

Se ha empleado al final de todos los métodos.