Capítulo 5
Primitivas de sincronización en Java

Exclusión mutua en Java

Por defecto en Java un objeto no está protegido, esto quiere decir que cualquier número de hilos puede estar ejecutando código dentro de un objeto. La exclusión mutua en Java se consigue mediante la palabra reservada *synchronized*.

Un método en Java puede estar precedido por **synchronized**, todos los métodos que lleven este modificador se ejecutarán en **exclusión mutua**, cuando un método sincronizado se está ejecutando se garantiza que ningún otro método sincronizado del mismo objeto podrá ejecutarse. Sin embargo, cualquier número de métodos no sincronizados se puede estar ejecutando dentro del objeto.

```
public class Sincronizados {
   public void m1() {
   }
   public synchronized void m2() {
   }
   public void m3() {
   }
   public synchronized void m4() {
   }
   public void m5() {
   }
}
```

Bien, ahora vamos a interpretar un poco más como funciona *synchronized*, el mismo también se lo puede aplicar a un bloque de código, para ello necesita hacer referencia a un objeto. En el siguiente ejemplo se aplica *synchronized* sobre el mismo objeto sobre el que se está invocando el método *m1()*. Esto hace que este bloque de código se ejecute en exclusión mutua con cualquier otro bloque sincronizado del mismo objeto, incluido los métodos que son precedidos por *synchronized*.

```
public class Sincronizados2 {
   public void m1() {
      // cualquier código. Será no sincronizado
      synchronized (this) {
         // sólo esta parte es sincronizada
      }
      // cualquier código. Será no sincronizado
   }
}
```

Podemos decir entonces que cuando se ejecuta *synchronized* se hace un ***Lock*** sobre el objeto (hecha un cerrojo) y cada vez que un *thread* intenta acceder a un bloque sincronizado le pregunta a ese objeto si no hay otro *thread* que ya tenga el *Lock* para ese objeto. El *Lock* es liberado cuando el *thread* que lo tiene tomado termina la ejecución normalmente, ejecuta un return o lanza exception.

Condición de sincronización en Java

Aún falta resolver el problema de la sincronización y comunicación entre los distintos hilos. Para ello Java proporciona tres métodos: ***wait()***, ***notify()*** y ***notifyAll()***, todos ellos definidos el clase Object de manera que cualquier objeto puede invocarlos. Antes de invocar a estos métodos debemos hacer que el hilo en curso haga el *Lock* sobre el mismo objeto, por lo tanto estos métodos deben ser invocados en un bloque *synchronized*. Veamos ahora la funcionalidad de cada uno de ellos:

wait(): le indica al hilo en curso que abandone la exclusión mutua sobre el objeto y se vaya al estado *"espera por wait"*.

notify(): un hilo arbitrario del conjunto de espera es seleccionado por el planificador para pasar al estado *"listo"*.

notifyAll(): todos los hilos del conjunto de espera son puestos en estado *"listo"*.

Ahora bien, cuando se invoca *wait* desde un objeto (llamamos al mismo "pera") indica al hilo que tiene el *Lock* sobre "pera" que vaya al estado *"espera por wait"*, dicho hilo para volver al estado *"listo"* deber esperar que se invoque un *notify* o *notifyAll* desde el mismo objeto por el cual fue enviado a espera, el objeto "pera" en este caso.