**ACTIVIDAD 14**

**Enunciado**

Modifica lo necesario en la función main del siguiente programa Java para que sus hilos h1 y h2 salgan de la situación de DeadLock en la que se encuentran al bloquear los recursos “a” y “b” dentro de la clase derivada Hilo.

**//Ejemplo8.java**

**public** **class** Ejemplo8 **extends** Thread {

**public** **static** **void** main(String args[]) {

Recurso a = **new** Recurso();

Recurso b = **new** Recurso();

Hilo h1 = **new** Hilo(a, b, "uno");

Hilo h2 = **new** Hilo(b, a, "dos");

h1.start();

h2.start();

}

}

**//Hilo.java**

**class** Hilo **extends** Thread {

Recurso a;

Recurso b;

**public** Hilo(Recurso a, Recurso b,String nombre) {

**super**(nombre);

**this**.a = a;

**this**.b = b;

}

**public** **void** run(){

System.***out***.println("Hilo " + **this**.getName() + " comienza");

**synchronized**(a) {

**try** {

Thread.*sleep*(100);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**synchronized**(b) {

}

System.***out***.println("Hilo " + **this**.getName() + " ha terminado");

}

}

}

**class** Recurso{

}

En este programa no se imprimen los mensajes que indican que los hilos *uno* y *dos* han terminado, porque dichos hilos se quedan para siempre esperando en la sentencia synchronized(b). Es decir, lo que se imprime es lo siguiente:

Hilo uno comienza

Hilo dos comienza

Cambia el nombre de la clase principal por el de Actividad14 y ten en cuenta que para comprobar que el DeadLock ha desaparecido, cualquiera de los siguientes resultados esperados es válido:

Hilo uno comienza

Hilo dos comienza

Hilo uno ha terminado

Hilo dos ha terminado

Hilo dos comienza

Hilo uno comienza

Hilo dos ha terminado

Hilo uno ha terminado

**SOLUCION**

**//Hilo.java**

class Hilo extends Thread {

Recurso a;

Recurso b;

public Hilo(Recurso a, Recurso b,String nombre) {

super(nombre);

this.a = a;

this.b = b;

}

public void run(){

System.out.println("Hilo " + this.getName() + " comienza"); //mensaje del hilo que comienza

synchronized(a) {

try {

Thread.sleep(100); //Sleep de 100 milisegundos

} catch (InterruptedException e) {

e.printStackTrace();

}

synchronized(b) {

}

System.out.println("Hilo " + this.getName() + " ha terminado"); //mensaje del hilo que termina

}

}

}

class Recurso{

}

**//Actividad14.java**

public class Actividad14 extends Thread {

public static void main(String args[]) {

Recurso a = new Recurso();

Recurso b = new Recurso();

//Se cambia el orden en el que se acceden a los recursos para evitar el deadlock

Hilo h1 = new Hilo(b, a, "uno");

Hilo h2 = new Hilo(b, a, "dos");

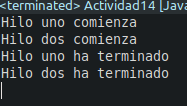
//Ejecutamos los hilos

h1.start();

h2.start();

}

}

****

**REQUISITOS**

1. Entregar las soluciones de las actividades en un único archivo comprimido donde tendremos:
   1. Un archivo de extensión .odt o .docx que contendrá
      1. Una copia del código del programa
      2. Una captura de pantalla del resultado de la ejecución del programa
   2. Los archivos \*.java empleados
2. El único archivo con la soluciones contendrá el nombre original de la actividad, seguido de un guión bajo y del primer apellido del alumno.

**CRITERIOS DE CORRECCIÓN**

1. Cumplimiento de los requisitos (15%).
2. Hacer lo que se indica en el enunciado (55%).
3. Claridad del código Java (15%).
4. Documentación del código Java (15%).
5. Explicación detallada de cómo se han resuelto problemas que hayan podido surgir en la realización de la actividad (hasta un 15% adicional sobre la nota obtenida). Dicha explicación se incluirá al principio de los archivos de código fuente Java mediante líneas de comentarios. La primera línea de comentarios sólo incluirá lo siguiente: // Resolución de problemas.
6. Variables no usadas (-15%).
7. Librerías no usadas (-15%).