1. **Con la clase CyclicBarrier realizar el siguiente programa. Imaginar que 4 personas que están situadas en diferentes ciudades realicen un viaje, que vayan a un punto B tras haberse encontrado los 4 en un punto A, primeramente. Como lo más probable es que alguno llegue antes que otro, los que ya hayan llegado tendrán que esperar en el punto A a que lleguen los restantes, La clase CyclicBarrier se encargará de controlar la espera en el punto A.**
2. **En el siguiente ejercicio, ¿se da una condición de carrera o no? ¿Por qué?**

**Se da cuando uno o más hilos ejecutan secciones que otro hilo podría usar al mismo tiempo, y en estas modifica variables que podrían afectar la ejecución del resto de hilos. Se pueden producir cuando varios threads NO acceden en exclusión mutua a un recurso compartido.**

public class Main implements Runnable{

int contador = 0;

public void incrementarContador(){

try {

Thread.sleep(10);

} catch (InterruptedException e) {

e.printStackTrace();

}

contador++;

}

public int obtenerContador(){

return contador;

}

@Override

public void run() {

incrementarContador();

System.out.println("Valor luego de incrementar el " + Thread.currentThread().getName() + ": " + obtenerContador());

}

public static void main(String[] args) {

Main contador = new Main();

Thread hilo1 = new Thread(contador, "primer hilo");

Thread hilo2 = new Thread(contador, "segundo hilo");

Thread hilo3 = new Thread(contador, "tercer hilo");

//La llamada a start() hace que se ejecute el método run

hilo1.start();

hilo2.start();

hilo3.start();

}

}

1. **Una de las formas de sincronización en hilos es mediante Semáforos. Explica en que consiste dicha forma de sincronización (clase Semaphore) y aporta un ejemplo de código en Java.**