UD2. Boletín 6. Prioridades.

1. ¿Qué hace el siguiente programa? ¿qué mostrará el código remarcado?

Ejer1Prioridad.java *

```
1  package Prioridades;
2
3  public class Ejer1Prioridad implements Runnable  {
4      private int contador = 0;
5      public void run()
6      {
7          for (int i = 0; i < 100000000; i++)
8          {
9              contador++;
10         }
11
12         System.out.println(Thread.currentThread().getName() +
13             " Prioridad: " + Thread.currentThread().getPriority() +
14             ". Fin de la ejecución.");
15     }
16 }
```

MainEjer1Prioridad.java *

```
1  package Prioridades;
2
3  public class MainEjer1Prioridad {
4      public static void main(String[] args)
5      {
6          Ejer1Prioridad p = new Ejer1Prioridad();
7
8          Thread hilo2 = new Thread(p);
9
10         hilo2.setPriority(Thread.MAX_PRIORITY);
11         System.out.println("Comenzando " + hilo2.getName() + "...");
12         hilo2.start();
13
14         Thread hilo3 = new Thread(p);
15         hilo3.setPriority(Thread.NORM_PRIORITY + 1);
16         System.out.println("Comenzando " + hilo3.getName() + "...");
17         hilo3.start();
18
19         System.out.println("Ejecutándose " + Thread.activeCount() +
20                 " hilos.");
21         try {
22             hilo2.join();
23         } catch (InterruptedException ex) {  }
24
25         try {
26             hilo3.join();
27         } catch (InterruptedException ex) { }
28     }
29 }
```

Este mostrará el proceso main ejecutandos, despues se ejecutan los otros dos hilos. Deberia de salir primero el hilo 2 debido a su prioridad maxima, pero dependiendo de cada ejecucion saldra de primero o no.

2. Teniendo en cuenta la siguiente aplicación, contesta a las cuestiones que te planteo:

Enteros.java

```
1  package Prioridades;
2
3  public class Enteros extends Thread {
4      int n, tiempo;
5      String nombre;
6
7      public Enteros(int valorN, String nombreHilo, int retardo) {
8          n = valorN;
9          nombre = nombreHilo;
10         tiempo=retardo; }
11
12     public void run() {
13         System.out.println("Prioridad de "+nombre+ " " + getPriority());
14
15       for (int i = 1; i <= n; i++) {
16             System.out.println(nombre);
17             try {
18                 sleep(tiempo);
19             } catch (InterruptedException e) {
20                    System.out.println("Interrupcion "+ nombre);}
21             }
22
23         System.out.println("......Termina "+nombre );
24     }
25 }
```

MainEnteros.java *

```
1  package Prioridades;
2
3  import javax.swing.JOptionPane;
4
5  public class MainEnteros {
6      public static void main (String[] args) {
7          int numero;
8          numero= Integer.parseInt(JOptionPane.showInputDialog("¿Cuantos valores? "));
9
10         Enteros thread1 = new Enteros(numero, "Caracola", 1000);
11         Enteros thread2 = new Enteros(numero, "Cangrejo", 1000);
12
13         thread1.setPriority(Thread.MAX_PRIORITY );
14         thread2.setPriority(Thread.MIN_PRIORITY );
15
16         thread1.start();
17
18         try {
19             thread1.join();
20         } catch (InterruptedException ex) { }
21
22         thread2.start();
23         try {
24             thread2.join();
25         } catch (InterruptedException ex) {  }
26
27         try {
28             for (int i = 1; i <=5; i++) {
29                 System.out.println("Valor i = "+i+" en Hilo principal en main()");
30                 Thread.sleep(3000);
31             }
32         }catch (InterruptedException e){
33             System.out.println("Interrupción del hilo principal en main()");
34         }
35
36         System.out.println("Termina Hilo principal en main()");
37     }
38 }
```

a) ¿Qué hace el código remarcado?

Ese codigo le aplica prioridad a los hilos correspondientes. En este caso se pone para el hilo1 la prioridad maxima, es decir, 10 y para el hilo2 la prioridad minima, es decir, 1. Esto nos dice que el hilo1 debe ejecutarse antes que el 2.

b) ¿Tiene alguna utilidad ese código remarcado si luego lanzo el thread1 y posteriormente llamo al método thread1.join()? Explica tu razonamiento

No, ya que la prioridad es para que se ejecute el hilo1 primero. Si lo ejecutas y despues lo paras para que se ejecute el 2 no tiene sentido ponerles prioridad.

c) ¿Qué salida hubiese obtenido si, manteniendo las prioridades asignadas en el código, lanzo primero thread2.start y thread2.join y luego al thread1? ¿se hubiera ejecutado primero thread1 ya que posee la máxima prioridad, es decir, 10?

No, se ejecuta primero el hilo 2 y despues se ejecuta el hilo 1. No se respetan las prioridades ya que estas forzando a que uno inicie antes que otro

3. **CARRERA DE HILOS.**

Diseña un programa en el que se creen dos hilos que van incrementando un contador (un hilo1 comienza con el contador a 0 y el otro con un valor más alto, por ejemplo 100), cada hilo tendrá una prioridad diferente. Los hilos incrementarán el contador hasta que el hilo que tiene prioridad más alta (y que empezó con el contador a 0) alcance al contador que corresponde a la tarea con ejecución más lenta (prioridad más baja y valor de contador más alto), en ese momento pararemos ambos hilos.

Supuestos:

a) Haz una primera versión en la que ambos hilos partan con la misma prioridad. Lanza primero al hilo1 y luego al hilo2. Explica qué ocurre.

El programa se ejecuta sin problemas y salen numero no muy dispersos entre ellos

b) ¿Qué ocurre si lanzo primero al hilo2, teniendo ambos igual prioridad?

Cuando se lanza primero el hilo con un contador inicial mas grande, el programa tarda mas en ejecutar y terminar, ya que el hilo con el contador pequeño tiene que alcanzar al hilo 2. Los numero que salen, aunque sea aleatorio, suelen ser bastante mas grande.

c) Ahora asigno a hilo1 la prioridad más alta y a hilo2 la más baja ¿qué ocurre en este caso? Lanzando primero hilo2.

El programa se ejecuta mucho mas rapido y los numeros que salen de los contadores son mas cercanos, ya que se le da prioridad al hilo que tiene el contador mas bajo.

d) ¿Y si lanzase primero hilo1?

El programa se ejecuta practicamente igual, ya que es el hilo 1 sigue teniendo la maxima prioridad.

e) Y si hilo1 teniendo máxima prioridad, lo lanzo primero y en la condición le indico que ejecute la aplicación mientras el valor del contador en hilo1 > al valor del contador en hilo2 ¿terminaría alguna vez?

Lo que pasa es que el hilo 1 tiene un contador menor que el hilo 2, por lo tanto nunca entra al bucle y acaba la ejecucion

f) ¿Hay alguna manera de que nunca parase de ejecutarse? (o que fuese muy difícil que parase)

Se puede complicar que el programe termine, haciendo que el contador del hilo que se ejecuta primero se bastante mas alto que el del otro hilo y dandole maxima prioridad y al otro minima. De esta forma es muy complicado que lo alcance pero no es imposible.

La unica forma de que sea imposible es cambiando la condicion del While a que pare cuando sus contadores sean iguales, ya que, que los contadores se igualen, tienen una prioridad tan pequeña que se puede considerar imposible.