

Creación de Threads

Fibonacci:

Podemos hacerlo de dos formas: implementando Runnable o extendiendo Thread. De las dos formas se que el método run() igual.

```
© Main.java  © Fibonacci.java x
1  package es.iespuertodelacruz.odhh.extensionThreats;
2
3  //public class Fibonacci implements Runnable{
4  public class Fibonacci extends Thread{
5      5 usages
6      public int parametro;
7
8      1 usage
9      public Fibonacci(int parametro) {
10         this.parametro = parametro;
11     }
12
13     @Override
14     public void run() {
15         int num1 = 1;
16         int num2 = 1;
17         while (num1 <= parametro || num2 <= parametro) {
18             if (num1 <= parametro) {
19                 System.out.println("num1: " + num1);
20             }
21             if (num2 <= parametro) {
22                 System.out.println("num2: " + num2);
23             }
24             num1 = num1 + num2;
25             num2 = num1 + num2;
26         }
27     }
28 }
```

Y cogemos el parámetro desde el constructor.

Luego le hacemos un new desde el main y le hacemos start o run en el caso de extender, y el new Thread en caso de implementar.

```
© Main.java x  © Fibonacci.java

1  package es.iespuertodelacruz.odhh.extensionThreats;
2
3  ▶ public class Main {
4  ▶      public static void main(String[] args) {
5          Fibonacci f = new Fibonacci( parametro: 20);
6          f.start();
7          //f.run();
8          //new Thread(f).start();
9
10     }
```

Carrera de hilos

```
© Main.java  © Coche.java x

1  package es.iespuertodelacruz.odhh.extensionThreats;
2
3  //public class Coche extends Thread {
4  // 10 usages
5  public class Coche implements Runnable {
6
7      2 usages
8      long time;
9
10     5 usages
11     public void setTime(long time) {
12         this.time = time;
13     }
14
15     @Override
16     public void run() {
17         for (int i = 0; i < 50; i++) {
18             try {
19                 Thread.sleep(time);
20             } catch (Exception e) {
21                 e.printStackTrace();
22             }
23         }
24         System.out.println("Ha llegado el coche: " + /*this*/new Thread().getName());
25     }
26 }
```

Si usas el extends no necesitas el "Thread." en el sleep y en el getName() le puedes hacer "this", haciéndolo con el implements usaremos lo que no está comentado. Y el setTime es para ponerle la cantidad de sleep que va a estar en cada pasada del bucle.



```
1 package es.iespuertodelacruz.odhh.extensionThreats;
2
3 public class Main {
4     public static void main(String[] args) {
5         Coche coche1 = new Coche();
6         Coche coche2 = new Coche();
7         Coche coche3 = new Coche();
8         Coche coche4 = new Coche();
9         Coche coche5 = new Coche();
10
11         coche1.setTime(100);
12         coche2.setTime(2000);
13         coche3.setTime(3000);
14         coche4.setTime(4000);
15         coche5.setTime(5000);
16
17         new Thread(coche1).start();
18         new Thread(coche2).start();
19         new Thread(coche3).start();
20         new Thread(coche4).start();
21         new Thread(coche5).start();
22         /*
23         Fibonacci f = new Fibonacci(20);
24         f.start();
25         //f.run();
26         //new Thread(f).start();
27         */
28     }
29 }
30 }
```

Y en el main si hacemos el extends no hace falta lo de new Thread sino haciendo un start funcionará, de todos modos funcionará así hagas lo que hagas.