Hecho por:

- Marco Antonio Cuberos Marín
- José Carlos López Henestrosa

Ejercicio 1

No da el mismo resultado ya que los tiempos de espera de cada hilo son aleatorios.

Está utilizando una clase que extiende de thead y se está probando con tres hilos diferentes.

Ejercicio 2

1. Describir brevemente lo que hace.

Consiste en ejecutar tres hilos, los cuales tienen un tiempo de ejecución aleatorio. Cada uno de ellos se identifica con un número. En el archivo .java viene una explicación más detallada de las líneas de código.

2. Ejecútadlo varias veces y describir lo que sucede.

- Primer caso:

Aquí vemos el reflejo de lo descrito en el primer apartado. Se ejecutan mostrando el nombre y el tiempo de ejecución.

```
Hola Mundo! Thread 2 349
Hola Mundo! Thread 1 1312
Hola Mundo! Thread 3 1471
BUILD SUCCESSFUL (total time: 1 second)
```

- Segundo caso:

Aquí se corrobora lo mencionado anteriormente. El tiempo y el orden respecto a la imagen anterior son completamente distintos.

```
run:
Hola Mundo! Thread 3 130
Hola Mundo! Thread 1 579
Hola Mundo! Thread 2 1859
BUILD SUCCESSFUL (total time: 2 seconds)
```

3. ¿Qué pasa si lo ejecutaís modificando el retardo de cada hilo? ¿y suprimiéndolo? Ejecutadlo haciendo estos cambios.

Depende de si se modifica el tiempo o si quitamos el método random(). En el primer caso, se ejecutaría de forma aleatoria pero tardando, en este caso, más. Esto se debe a que he incrementado el número.

```
t1 = new TestTh( "Thread 1",(int)(Math.random()*3000) );
t2 = new TestTh( "Thread 2",(int)(Math.random()*3000) );
t3 = new TestTh( "Thread 3",(int)(Math.random()*3000) );
```

```
run:
Hola Mundo! Thread 3 1016
Hola Mundo! Thread 2 2077
Hola Mundo! Thread 1 2475
BUILD SUCCESSFUL (total time: 2 seconds)
```

Como podemos apreciar, el tiempo de ejecución se incrementa respecto a las capturas del segundo apartado.

```
run:
Hola Mundo! Thread 2 1733
Hola Mundo! Thread 1 1191
Hola Mundo! Thread 3 1855
BUILD SUCCESSFUL (total time: 0 seconds)
```

Si comentamos el sleep(), los hilos se ejecutarían en orden y en los milisegundos que tengan que tardar para terminar su actividad.

4. ¿Qué mecanismo de creación de Thread utiliza? ¿Cuántos hilos crea?

El mecanismo que utiliza es extender de la clase Thread. Crea 3 hilos, los cuales se declaran en el main.

Ejercicio 3

```
Tarde@DESKTOP-3H43RO6 MINGW64 ~/Desktop/Actividad6PSP/Ejercicio 3
$ java Hilo2
Soy Iv├ín (693)
Soy Vicente (1161)
Soy Isabel (4550)

Tarde@DESKTOP-3H43RO6 MINGW64 ~/Desktop/Actividad6PSP/Ejercicio 3
$ java Hilo2
Soy Isabel (2837)
Soy Vicente (3278)
Soy Iv├ín (4494)

Tarde@DESKTOP-3H43RO6 MINGW64 ~/Desktop/Actividad6PSP/Ejercicio 3
$ java Hilo2
Soy Vicente (1797)
Soy Iv├ín (2860)
Soy Isabel (3602)

Tarde@DESKTOP-3H43RO6 MINGW64 ~/Desktop/Actividad6PSP/Ejercicio 3
$
```

1. ¿Cuál es la diferencia con el ejercicio 1?

El resultado es el mismo, lo único que cambia es que la clase hilo en vez de extender de Thread lo que hace es implementar la interfaz Runnable.

2. ¿Creeis que en este caso es recomendable el cambio realizado con respecto al ejercicio 3? Razonad vuestra respuesta.

En este caso la diferencia estaría en el código de run del hilo, que hace uso del método sleep. Sería más correcto la primera forma ya que estaría ejecutando un método de esa misma clase al extender de Thread. En el caso de la interfaz Runnable, esta no tiene acceso a ese método directamente ya que en ese caso un no es un Thread, y para tener acceso tiene que importar ese método de la clase Thread, por lo que lo mas correcto en este caso seria que la clase extendiera de Thread.

Ejercicio 4

1. Ejecutad el programa varias veces sin modificar el código, ¿Que prioridad tienen los hilos?

```
La prioridad del hilo1 es: 5
La prioridad del hilo2 es: 5
HOLA SOL!!!...: 1
HOLA LUNA!!...: 1
HOLA LUNA!!...: 2
HOLA SOL!!!...: 2
HOLA LUNA!!...: 3
HOLA SOL!!!...: 3
HOLA SOL!!!...: 4
HOLA LUNA!!...: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
HOLA LUNA!!...: 1
HOLA SOL!!!...: 1
La prioridad del hilo1 es: 5
La prioridad del hilo2 es: 5
HOLA LUNA!!...: 2
HOLA SOL!!!...: 2
HOLA LUNA!!...: 3
HOLA SOL!!!...: 3
HOLA LUNA!!...: 4
HOLA SOL!!!...: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

La prioridad de ambos es la misma.

2. Ejecutad el programa quitando los símbolos de comentarios, a las líneas de código que asignan prioridad a los hilos. ¿Qué prioridad tienen ahora los hilos? ¿Se ejecutan los hilos en el orden esperado?

```
La prioridad del hilo1 es: 10
La prioridad del hilo2 es: 1
HOLA SOL!!!...: 1
HOLA LUNA!!...: 1
HOLA LUNA!!...: 2
HOLA SOL!!!...: 2
HOLA SOL!!!...: 3
HOLA LUNA!!...: 3
HOLA LUNA!!...: 4
HOLA SOL!!!...: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
La prioridad del hilo1 es: 10
HOLA SOL!!!...: 1
HOLA LUNA!!...: 1
La prioridad del hilo2 es: 1
HOLA LUNA!!...: 2
HOLA SOL!!!...: 2
HOLA LUNA!!...: 3
HOLA SOL!!!...: 3
HOLA SOL!!!...: 4
HOLA LUNA!!...: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

La prioridad ahora recae en el hilo 1 y siempre se ejecutan en el mismo orden: el hilo 1 es el que comienza siempre.

3. Ejecutad varias veces el programa ¿Cuando se visualiza el mensaje que da la prioridad de los hilos? ¿Que debeís hacer para que éstos se vean al final del programa? Modificad el código necesario para ello.

Se visualiza cuando los hilos comienzan a ejecutarse. Para que se vean al final del programa, tenemos que poner el código cuando se acabe de ejecutarse el bucle *for* que escribe HOLA…

```
@Override
public void run(){
    for (int i=1; i<5; i++) {
        System.out.println(getName()+": "+i);
        try {
            sleep(100);
        } catch(InterruptedException ex) {}
    }
    System.out.println("La prioridad del " + getName() +"es: " + getPriority());
}
```

Poniendo el println después del bucle y usando los métodos sin el objeto.

4. Revisad y corregid el sangrado del archivo modificado.

Está en el .java adjunto

Ejercicio 5

1. Escribid las instrucciones que necesitaríais para conocer su nombre de hilo y su id:

```java
Thread hiloEjer5 = Thread.currentThread();

//Para obtener el nombre

hiloEjer5.getName();

//Para obtener el id

hiloEjer5.getId();
```

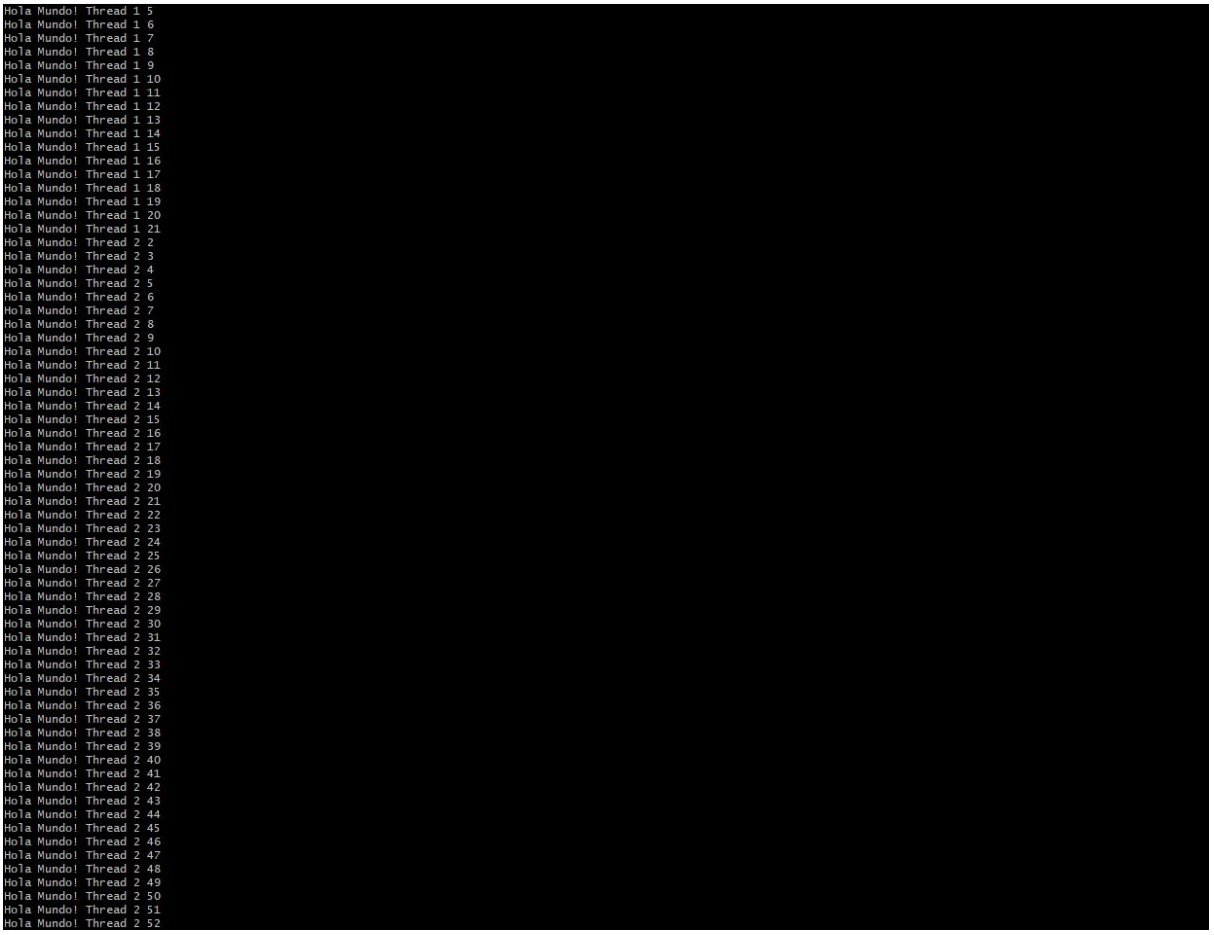
2. Suponiendo que se está ejecutando, escribid las instrucción necesaria para que ceda el paso a otro hilo de forma estática:

otroHilo.join();

Esta instrucción obligaría a esperar a todos los demas hilos en ejecución a que termine el hilo en la que se llama, en este caso ‘otroHilo’. Aqui dejo un código de ejemplo y un ejemplo de ejecución.

```java
class Hilo1 extends Thread {
    private String nombre;
    private int retardo;
    public Hilo1( String s,int d ) {
        nombre = s;
        retardo = d;
    }
    public void run() {
        /*try {
            sleep(retardo);
        }catch( InterruptedException e ) {}*/
        for(int i = 0; i<1000; i++){
            System.out.println( "Hola Mundo! "+nombre+" "+i );
        }
    }
}

public class EjemploJoin {
    public static void main( String args[] ) {
        Hilo1 t1,t2;
        t1 = new Hilo1( "Thread 1",(int)(Math.random()*2000) );
        t2 = new Hilo1( "Thread 2",(int)(Math.random()*2000) );
        t1.start();
        t2.start();
        try{
            t2.join();
        }catch(InterruptedException we){}
    }
}
```



Se ha ejecutado el hilo1 primero y después le ha seguido el hilo2, al cual le hemos hecho un join, por lo que ese hilo se va a ejecutar entero y mientras tanto el hilo1 va a estar esperando a que termine para después continuar.

Ejercicio 6

```java
class HiloEjer6 extends Thread{
     private String s;
     public HiloEjer6(String name) {
        this.s = name;
     }
     public void run() {
        for(int x = 0; x <1000; x++){
            /*try{
                sleep(1000);
            }catch(InterruptedException EE){}*/
            //El sleep dormiria el hilo y el join lo saltaria, ya que considera que el hilo esta parado
            System.out.println(this.s);
        }
    }
}
public class Ejer6 {
     public static void main(String[] args) throws InterruptedException{
         HiloEjer6 h1 = new HiloEjer6("A");
         HiloEjer6 h2 = new HiloEjer6("B");
        try{
            h2.start();
            h1.start();
            h2.join();
        }catch(InterruptedException EE){}
         // Selecciona el hilo actual que se esta ejecutando y lo
        // encadena a h1
        System.out.println("MAIN, THE END");
     }
}
```

Lo único que haría falta es tirar el join al segundo hilo, entonces este se ejecutaría hasta terminar y después seguiría el hilo 1.

El sleep dentro del hilo invalidaría el join, ya que cuando llega al join es probable que el hilo ya esté interrumpido, por lo que considera que el hilo está terminado y libera de nuevo la cola de ejecución.

Hecho por:
Marco Antonio Cuberos Marín
José Carlos López Henestrosa