Unit 1. Introduction to Concurrent Programming

What is a concurrent program?

Set of threads running in parallel that can perform task over the same data, as they can have access to the same data. Concurrency can be performed in two ways: **Logical Parallelism**(one core), or **Real Parallelism**(multiple cores).

The upsides are improvements in **efficiency**, **scalability**, **communication management**, **flexibility** and **minor semantic gap**.
The
problem comes when taking into account the problems of concurrent programs, as they are more complex and involve taking into account **race conditions** and **Deadlocks**.

Why Concurrency in Java?

It is mainly due to some of the functionalities that Java and its special architecture involve, since there is **library support**, incorporated **Threads**, and it is also a common language.

Creating Threads

To instantiate new threads, we can follow two options:

Implementing Runnable

```java
// We define the method run() containing
// all the code executed by the thread

public class HelloRunnable implements Runnable{
    public void run() { System.out.println("Hello world");}

    public static void main(String args[]) {
        (new Thread(new HelloRunnable())).start();
    }
}

// the start method runs what is inside the run() method.
```

```
// If you need several instances:
Thread t = new Thread(new H()); // H is the current class
t.start();
```

Using anonymous classes

```
new Thread(new Runnable() {
    public void run() {
        System.out.println("Iniciando viaje");
    }
}).start();
```

Extending class Thread(less common)

```
// General way
public class HelloThread extends Thread {
    public void run()
    {
        System.out.println(”Hello world!");
    }

    public static void main(String args[])
    {
        (new HelloThread()).start();
    }
}
```

```
// If you need several instances
public class H extends Thread {
    public void run() {
        System.out.println("Asd");
    }
}
H t = new H(); t.start();
```

```
new Thread() {
    public void run() {
        System.out.println(”execute thread");
    }
}.start();
```

💡 Note that Threads start with the method start(), not the method run()

Naming Threads

When managing different Threads, you will need to identify them. For that purpose you can put them a name:

```
// You can do it in the instantiation
new T("Thread num 1").start();

// Or with setname()
t.setName("Thread num 1");

// The names can be retrieved using
t.getName();
// or
Thread.currentThread.getName();
```

Using Threads

There are many operations beneath Threads, the most common ones are:

Function	Purpose
Thread.sleep(long millis);	Suspends the tread for given time
Thread.interrupt();	Reactivates suspended threads
t.join();	Current Thread waits to other thread to finish(for **t** to finish)

Function	Purpose
Thread.join(long millis)	Give maximum waiting time
Thread.currentThread();	Returns reference to currently running thread
Thread.isAlive();	Returns if Thread has started and not finished yet
Thread.yield();	Leaves the processor.