¿Qué es un Java Infinite Stream? . Los streams infinitos son una de las características más interesantes del API de Java 8. Nos permiten realizar operaciones habituales enfocándolas desde la programación funcional. Por ejemplo supongamos que queremos sumar los primeros 1000 números enteros en Java . Es una operación muy sencilla simplemente construimos un bucle for, lo recorremos y sumamos.

```
package com.arquitecturajava;

public class Principal {

public static void main(String[] args) {


double total=0;
long t1=System.nanoTime();
for(int i=0;i&lt;1_000;i++) {

total=total+(i+1);

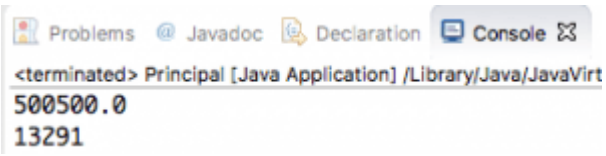
}
long t2=System.nanoTime();

System.out.println(total);
System.out.println(t2-t1);

}

}
```

En este caso he usado también el método System.nanoTime() que permite hacer un calculo preciso del tiempo que esta operación tarda en ejecutarse.



Todo funciona correctamente y Java nos realiza la suma . Ahora bien podemos enfocar de otra manera

Java Infinite Stream

Podemos usar un Stream infinito y realizar la misma operación apoyándonos en programación funcional.

```
package com.arquitecturajava;

import java.util.stream.DoubleStream;

public class Principal2 {

 public static void main(String[] args) {

 long t1=System.nanoTime();

 double resultado=DoubleStream.iterate(0,i->i+1).limit(1_001)
 .sum();

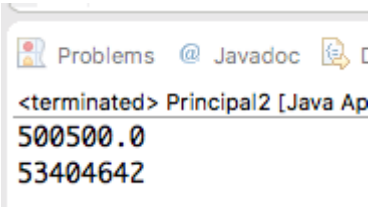
 long t2=System.nanoTime();
```

```
 System.out.println(resultado);
 System.out.println(t2-t1);

 }

}
```

Acabamos de usar la clase DoubleStream que es capaz de generar Streams de Doubles , en este caso hemos usado el método iterate() para generar un stream infinito. Esto quiere decir que se generarán números hasta que decidamos un límite. En este caso como queremos que sean los primeros 1000 elementos ponemos el tope en 1001. El resultado será idéntico con la salvedad del tiempo de ejecución que será mucho mayor.



Acabamos de realizar la operación de suma de n términos con un Java Infinite Stream. El resultado parece anecdótico ya que el bucle for se ejecuta mucho más rápido. Esto es normal ya que es una operación básica . Sin embargo la gran ventaja de los Streams infinitos viene en que se pueden paralelizar utilizando el método parallel() de los streams .

```
package com.arquitecturajava;

import java.util.stream.DoubleStream;

public class Principal3 {
```

```
public static void main(String[] args) {
long t1=System.nanoTime();

double
resultado=DoubleStream.iterate(0,i->i+1).limit(1_001).parallel().sum()
;

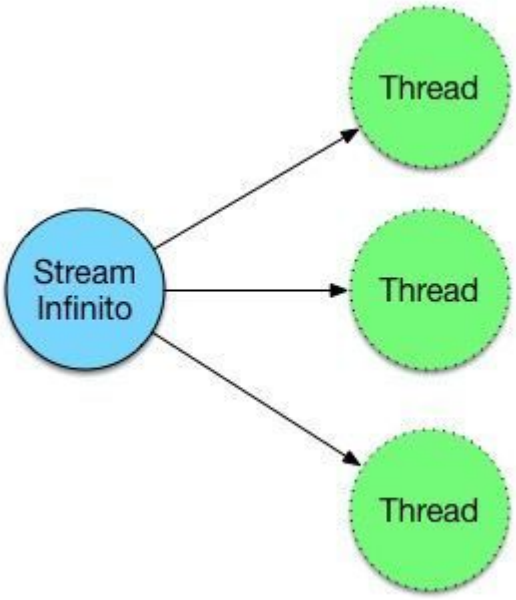
long t2=System.nanoTime();

System.out.println(resultado);
System.out.println(t2-t1);

}

}
```

Esto abre la posibilidad de procesar una operación compleja entre varios hilos de forma concurrente, mejorando el rendimiento algo que un simple bucle for no es sencillo



Los streams infinitos son útiles cuando queremos realizar operaciones complejas sobre un conjunto amplio de datos.

Otros artículos relacionados : Java Lambda , Java Streams , Stream y FlatMap