


# Streams

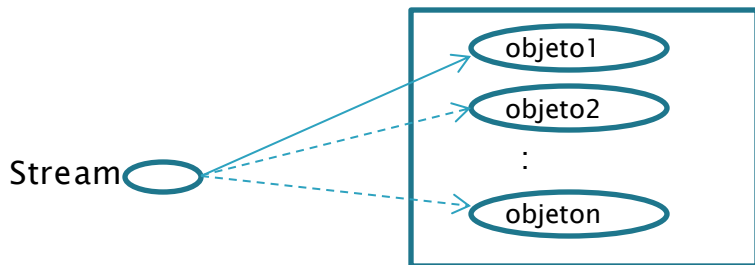


# ¿Qué es un Stream?

- Objeto que permite realizar de forma rápida y sencilla operaciones de búsqueda, filtrado, recolección, etc. sobre un grupo de datos (array, colección o serie discreta de datos)
  - Para manipular un Stream utilizamos la interfaz Stream de `java.util.stream`
  - Otras variantes como `IntStream`, `LongStream` o `DoubleStream` se emplean para trabajar con tipos primitivos
- 

# Funcionamiento

➤ Recorre los datos desde el principio hasta el final y durante el recorrido realiza algún tipo de cálculo u operación



➤ Una vez realizado el recorrido, el stream se cierra y no puede volver a utilizarse

# Creación de un Stream

## ➤ A partir de una colección:

```
ArrayList<Integer> nums=new ArrayList<>();  
nums.add(20);nums.add(100);nums.add(8);  
Stream<Integer> st=nums.stream();
```

## ➤ A partir de un array:

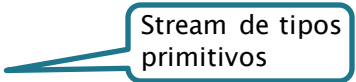
```
String[] cads={"a","xy","jk","mv"};  
Stream<String> st= Arrays.stream(cads);
```

## ➤ A partir de una serie discreta de datos:

```
Stream<Double> st=Stream.of(2.4, 7.4, 9.1);
```

## ➤ A partir de un rango de datos:

```
IntStream stint=IntStream.range(1,10);  
IntStream stint2=IntStream.rangeClosed(1,10);
```



Stream de tipos  
primitivos

# Tipos de métodos de Stream

- Métodos intermedios. El resultado de su ejecución es un nuevo Stream. Ejemplos: filtrado y transformación de datos, ordenación, etc.
  - Métodos finales. Generan un resultado. Pueden ser void o devolver un valor resultado de alguna operación. Ejemplos: calculo (suma, mayor, menor, ...), búsquedas, reducción, etc.
- 