

5 Programación funcional:

Una expresión lambda caracterizada por el símbolo "->" en la asignación de una lógica, es decir código, a una función genérica definida en una interfaz.

Los objetos que reciben esta asignación, pertenecientes a la clase de la interfaz, se pueden llamar como argumentos de otras funciones.

Por ejemplo el objeto suma se recoge mediante la expresión

```
NumeroEnteros suma=(a,b)->a+b;
```

Con la funcionalidad de sumar dos números.

La programación funcional es un nuevo paradigma de programación en el cual las funciones se pueden gestionar como objetos, así por ejemplo se pueden asignar funciones a variables así poderlas enviar como argumentos en la llamada a un método.

Las expresiones lambda se pueden almacenar en variables si el tipo de variable es una interfaz que tiene un solo método. La expresión lambda debe tener la misma cantidad de parámetros y el mismo tipo de retorno que ese método. **Java tiene muchos de estos tipos de interfaces integrados, como la Consumer interfaz (que se encuentra en el java.util.paquete) utilizada por las listas.**

Se recogen las siguientes interfaces implementadas en la librería Java.util.function

Consumer<T> representa una función genérica que acepta un dato simple y no retorna resultado.

Function<T,R> Representa una función genérica que acepta un dato y retorna un resultado

Predicate<T> representa una función booleana que acepta un dato

Supplier<T> representa un proveedor de resultados.

5.46 Haz un ejemplo usando la expresión lambda Consumer y BiConsumer

Ejercicio

Las lambdas de tipo **Consumer** reciben por parámetro un elemento y no devuelven nada.

Las lambdas de tipo **BiConsumer** reciben por parámetro dos elementos y no devuelven nada.

```
1 package Ejercicio_46_Consumer_BiConsumer;
2 import java.time.LocalDate;
3 import java.time.Period;
4 import java.util.function.*;
5
6 public class LambdaConsumer {
7
8
9
10 public static void main (String[] args) {
11     System.out.println("*****CONSUMER*****");
12
13     Consumer <String> imprimirNombre =(nombre)->System.out.println("Tu nombre es :"+nombre);
14     Consumer <LocalDate> imprimirEdad=fecha->{
15         LocalDate hoy=LocalDate.now();
16         LocalDate aux=LocalDate.of(hoy.getYear(), fecha.getMonthValue(), fecha.getDayOfMonth());
17         if(aux.compareTo(hoy)<0) {
18             System.out.println("Este año "+aux.getYear()+" ya has cumplido años tienes "+Period.between(fecha, LocalDate.now()).getYears());
19         }else if(Period.between(aux,LocalDate.now()).getDays()==0) {
20             System.out.println("Felicitades es tu cumpleaños de los "+Period.between(fecha, LocalDate.now()).getYears());
21         }
22     }else {
23         System.out.println("Este año "+aux.getYear()+" aun no has cumplido años tienes "+Period.between(fecha, LocalDate.now()).getYears());
24     }
25 };
26
27     imprimirNombre.accept("cesar");
28     imprimirEdad.accept(LocalDate.of(1977, 1, 1));
29     imprimirEdad.accept(LocalDate.of(1977, LocalDate.now().getMonth(), LocalDate.now().getDayOfMonth()));
30     imprimirEdad.accept(LocalDate.of(1977, 7, 4));
31     imprimirEdad.accept(LocalDate.of(1977, 12, 31));
32
33     System.out.println("*****BiCONSUMER*****");
34
35     BiConsumer <String,String> imprimirNombreApellidos=(nombre,apellidos)->System.out.println("Tu nombre es "+apellidos+", "+nombre);
36
37     BiConsumer<String,LocalDate> imprimirNombreDiaNac=(nombre,fNac)->System.out.println(nombre+" naciste del día de la semana "+fNac.getDayOfWeek());
38     imprimirNombreApellidos.accept("Cesar","Bouzas Soto");
39     imprimirNombreDiaNac.accept("Cesar",LocalDate.of(1977, 7, 4));
40 }
41 }
42 }
```

Problemas @ Javadoc Declaración Consola SonarLint On-The-Fly Preparación

<terminado> LambdaConsumer [Aplicación Java] C:\Program Files\ eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.2.v20220201-1208\jre\bin\javaw.exe (15 Jun 2022 20:28:19 - 20:28:19) [pid: 4156]

```
*****CONSUMER*****
Tu nombre es :cesar
Este año 2022 ya has cumplido años tienes 45
Felicitades es tu cumpleaños de los 45
Este año 2022 aun no has cumplido años tienes 44
Este año 2022 aun no has cumplido años tienes 44
*****BiCONSUMER*****
Tu nombre es Bouzas Soto,Cesar
Cesar naciste del día de la semana MONDAY
```

Activar Win
Ve a Configura

5.47 Crea una clase PersonaJava8 donde tenga un nombre y apellidos. Desde el main crea un método que le cambie el nombre a una persona. Crea un método que te pinte por pantalla un String introducido por parámetro. Llama a este método para mostrar el contenido de un arrayList Haz un método que devuelva un numero al azar. Crea una PersonaJava8 con la expresión Supplier y cambiale el nombre.

```

1 package Ejercicio_47_PersonaJava8;
2 import java.util.*;
3 import java.util.function.BiConsumer;
4 import java.util.function.Consumer;
5 import java.util.function.Function;
6 import java.util.function.Supplier;
7
8 public class Principal {
9
10     public static void main(String[] args) {
11         PersonaJava8 p1=new PersonaJava8("Cesar","Roucas Soto");
12         PersonaJava8 p2=new PersonaJava8("Natividad","Núñez Conde");
13         PersonaJava8 p3=new PersonaJava8("Manuel","Roucas Núñez");
14         PersonaJava8 p4=new PersonaJava8("Mateo","Roucas Núñez");
15         PersonaJava8 p5=new PersonaJava8("Pedro","Flying Tractor");
16         ArrayList<PersonaJava8> personas=new ArrayList<>();
17         personas.add(p1);
18         personas.add(p2);
19         personas.add(p3);
20         personas.add(p4);
21         personas.add(p5);
22         BiConsumer<PersonaJava8,String> cambiarNombre=(persona,nombre)->{
23             System.out.println("Cambiamos el nombre.....");
24             System.out.println("El nombre de la persona es "+persona.getNombre());
25             persona.setNombre(nombre);
26             System.out.println("El nombre de la persona es "+persona.getNombre());
27         };
28         Consumer<PersonaJava8> imprimirNombre=persona->System.out.println("El nombre de la persona es "+persona.getNombre());
29         imprimirNombre.accept(p1);
30         cambiarNombre.accept(p5,"IceMan");
31         imprimirNombre.accept(p5);
32         System.out.println("*****Imprimir String Consumer*****");
33         Consumer<String> imprimirPalabra->System.out.println(palabra);
34         for(PersonaJava8 p:personas) {
35             imprimirNombre.accept(p);
36             imprimir.accept(p.getApellidos());
37         }
38         System.out.println("*****Aleatorio BiFunction con max,result*****");
39         BiFunction<Integer,Integer,Integer> numAleatorio=(min,max)->{return (int)(Math.round(Math.random()*(max-min))+min)};
40         int suma=0;
41         for(int i=0;i<100;i++) {
42             int num=numAleatorio.apply(0, 9);
43             suma+=num;
44             System.out.print(num+" ");
45             if(i%10==0 && i!=0) System.out.print("\n");
46         }
47         System.out.println("\nEl promedio de todos estos numeros es "+suma/(100));
48
49         System.out.println("*****Supplier salida PersonaJava8*****");
50         Supplier<PersonaJava8> crearPersona=()->{return new PersonaJava8();};
51         PersonaJava8 p6=crearPersona.get();
52         imprimirNombre.accept(p6);
53         imprimir.accept(p6.getApellidos());
54     }
55 }

```

Trastear con dos puntos.....

5.48 Crea una expresión Function que devuelva un valor aleatorio entre 1 y un valor introducido por parámetro. Crea una expresión Function que reciba una cadena y la devuelva convertida en mayúsculas. Crea una expresión BiFunction que reciba 2 números, que compare cual es mayor y devuelva un número al azar entre ambos.

```

1 package Ejercicio_48_Funciones;
2 import java.util.*;
3 import java.util.function.BiFunction;
4 import java.util.function.Function;
5
6 public class Funciones {
7
8     public static void main(String[] args) {
9
10         // Función que devuelva un valor aleatorio entre 1 y un valor introducido por parámetro
11         Function<Integer,Integer> numAleatorio=(numero)->{return (int)(Math.round(Math.random()*(numero-1))+1)};
12
13         // Función que reciba una cadena y la devuelva convertida en mayúsculas
14         Function<String,String> ponerMayusculas=(cadena)->{return cadena.toUpperCase()};
15
16         // Función que reciba 2 números, que compare cual es mayor y devuelva un número al azar entre ambos
17         BiFunction<Integer,Integer,Integer> numAleatorioEntre=(n1,n2)->{
18             int max=(Integer.compare(n1,n2)>0)?n1:n2;
19             int min=(Integer.compare(n1,n2)<0)?n1:n2;
20             System.out.println("El máximo es "+max+" y el número aleatorio entre "+max+" y "+min+" es ");
21             return (int)(Math.round(Math.random()*(max-min))+min);
22         };
23
24         // Ejecución de las funciones
25         int n1=1;
26         String txt="";
27         for(int i=0;i<10;i++) {
28             txt+=numAleatorio.apply(n1)+" ";
29             n1=numAleatorio.apply(100);
30         }
31         System.out.println(txt);
32
33         // Ejecución de la función ponerMayusculas
34         String txt2="hola mundo";
35         System.out.println(ponerMayusculas.apply(txt2));
36
37         // Ejecución de la función numAleatorioEntre
38         int n1=10;
39         int n2=100;
40         System.out.println(numAleatorioEntre.apply(n1,n2));
41
42         // Ejecución de la función numAleatorioEntre con valores aleatorios
43         int n1=numAleatorio.apply(100);
44         int n2=numAleatorio.apply(100);
45         System.out.println(numAleatorioEntre.apply(n1,n2));
46     }
47 }

```

- 5.49 Crea un Predicate que reciba un número y te diga si es divisible entre 2.
Crea un BiPredicate que te diga si dos cadenas introducidas son iguales ignorando mayúsculas/minúsculas.

```
DivisibleCadenasIguales.java X
1 package Ejercicio_49_Predicate_divisibles_iguales;
2
3 import java.util.function.Predicate;
4 import java.util.function.BiPredicate;
5
6 public class DivisibleCadenasIguales {
7
8
9
10 public static void main(String[] args) {
11     System.out.println("*****Predicate Par*****");
12     Predicate<Integer> esPar=numero->numero%2==0;
13     String txt;
14     for(int i=0;i<10;i++) {
15         txt=(esPar.test(i))?" es par":" es impar";
16         System.out.println("El numero "+i+txt);
17     }
18     System.out.println("***** BiPredicate Comparar dos Strings Equals ignorando mayúsculas*****");
19     BiPredicate<String,String> sonIguales=(cadena1,cadena2)->return cadena1.equalsIgnoreCase(cadena2);
20     System.out.println("¿Es lo mismo Tuberculo que ver tu Culo? " +sonIguales.test("Tuberculo","vertuculo"));
21     System.out.println("¿Es lo mismo Mar flores que flores en el Mar? " +sonIguales.test("Mar flores","flores en el mar"));
22     System.out.println("¿Es lo mismo cesar bouzas soto que Cesar Bouzas Soto? " +sonIguales.test("cesar bouzas soto","Cesar B
23
24
25 }
```

```
<terminado> DivisibleCadenasIguales [Aplicación Java] C:\Program Files\ eclipse\plugins\org.eclipse.justi.openjdk.hotspot
*****Predicate Par*****
El numero 0 es par
El numero 1 es impar
El numero 2 es par
El numero 3 es impar
El numero 4 es par
El numero 5 es impar
El numero 6 es par
El numero 7 es impar
El numero 8 es par
El numero 9 es impar
El numero 10 es par
El numero 11 es impar
El numero 12 es par
El numero 13 es impar
El numero 14 es par
El numero 15 es impar
El numero 16 es par
El numero 17 es impar
El numero 18 es par
El numero 19 es impar
***** BiPredicate Comparar dos Strings Equals ignorando mayúsculas*****
¿Es lo mismo Tuberculo que ver tu Culo? false
¿Es lo mismo Mar flores que flores en el Mar? false
¿Es lo mismo cesar bouzas soto que Cesar Bouzas Soto? true
```

- 5.50 Crea una interfaz funcional que contenga un método “operacion” al que se le pasen dos números (o doubles). Crea una clase calculadora que tenga un método al que se le pasen dos enteros (o doubles) y un objeto tipo Arithmetic y realice la operación entre ambos valores. Crea otro método que haga lo mismo pero en vez de pasarle un objeto tipo Arithmetic se le pase una BiFuncional. En el main, crea dos operaciones diferentes (suma y resta, por ejemplo) de tipo Arithmetic y creando un objeto tipo calculadora mira las diferentes posibilidades que hay de llamar a sus métodos.

```
*Calculadora.java X InterfaceCalculadora.java
4
5 public int sumar(int a,int b){return a+b;}
6 public int restar(int a,int b){return a-b;}
7 public int multiplicar(int a,int b){return a*b;}
8 public int dividir(int a,int b){return a/b;}
9
10
11 public class Calculadora {
12
13
14
15 public void sumar(int a, int b,Arithmetic arit) {
16     System.out.println("La suma aritmetica de a= "+a+" y b= "+b+" es "+arit.sumar(a, b));
17 }
18 public void restar(int a, int b,Arithmetic arit) {
19     System.out.println("La resta aritmetica de a= "+a+" y b= "+b+" es "+arit.restar(a, b));
20 }
21
22 public void multiplicar(int a, int b,Arithmetic arit) {
23     System.out.println("La multiplicacion Aritmetica de a= "+a+" y b= "+b+" es "+arit.multiplicar(a, b));
24 }
25 public void dividir(int a, int b,Arithmetic arit) {
26     System.out.println("La division Aritmetica de a= "+a+" y b= "+b+" es "+arit.dividir(a, b));
27 }
28
29 public void sumar(int a, int b) {
30     InterfaceCalculadora sumar = (n1,n2)->n1+n2;
31     System.out.println("La suma de a= "+a+" y b= "+b+" es "+sumar.operacion(a, b));
32 }
33 public void restar(int a, int b) {
34     InterfaceCalculadora restar = (n1,n2)->n1-n2;
35     System.out.println("La resta de a= "+a+" y b= "+b+" es "+restar.operacion(a, b));
36 }
37 public void multiplicar(int a,int b) {
38     InterfaceCalculadora multiplicar=(n1,n2)->n1*n2;
39     System.out.println("La multiplicacion de "+a+" y "+b+" es "+multiplicar.operacion(a, b));
40 }
41 public void dividir(int a,int b) {
42     InterfaceCalculadora dividir=(n1,n2)->n1/n2;
43     System.out.println("La division de "+a+" y "+b+" es "+dividir.operacion(a, b));
44 }
45
46
47 public static void main (String args[]) {
48     Arithmetic ar=new Arithmetic();
49     Calculadora casio=new Calculadora();
50     casio.sumar(10, 20, ar);
51     casio.restar(10, 20, ar);
52     casio.multiplicar(10, 20,ar);
53     casio.dividir(10,20,ar);
54
55     casio.sumar(10, 20);
56     casio.restar(10, 20);
57     casio.multiplicar(10,20 );
58     casio.dividir(10,20 );
59
60 }
61
62 }
63 }
```

```
<terminado> Calculadora [Aplicación Java] C:\Program Files\ eclipse\plugins\org.eclipse.justi.openjdk.hotspot
La suma aritmetica de a= 10 y b= 20 es 30
La resta aritmetica de a= 10 y b= 20 es -10
La multiplicacion Aritmetica de a= 10 y b= 20 es 200
La division Aritmetica de a= 10 y b= 20 es 0
La suma de a= 10 y b= 20 es 30
La resta de a= 10 y b= 20 es -10
La multiplicacion de 10 y 20 es 200
La division de 10 y 20 es 0
```

5.51 Crea un Stream de nombres de personas y muestra sus datos por consola (en 2 líneas de código) .

Hay muchas formas de crear un Stream todo stream tiene un origen source, es decir fuente de datos. La fuente puede ser de naturaleza diversa, desde un array , a datos procedentes de un disco o de la tarjeta de red.

Como Stream es una interface, no podemos hacer directamente un new , hay muchas formas de crear un stream , un ejemplo sencillo es el método static of de interface Stream. Desde java 8 las interfaces pueden tener ciertos métodos con código, concretamente son los métodos default y los static. Stream.of() es un método estático de Stream y se usa para crear objetos Stream. Se le puede pasar por parámetro un conjunto de valores separados por comas o un array. En este ejemplo utilizaré la primera de las dos opciones.



```
1 package Ejercicio_51_Stream;
2 import java.util.stream.Stream;
3 public class CrearStram {
4
5     public static void main(String[] args) {
6
7         Stream<String> nombres= Stream.of("Cesar","Nati","mateo","manuel");
8         nombres.forEach(n->System.out.println(n));
9
10        //nombres.forEach(System.out::println);
11    }
12
13 }
14
```

<terminado> Crear
Cesar
Nati
mateo
manuel

5.52 Crea un Stream de nombre de personas en mayúsculas, conviértelo en nombres con solo la primera en mayúscula (el resto en minúsculas), muéstralo por pantalla y luego mételo dentro de una lista.

Otra forma interesante de crear un Stream es crear un Stream con el método stream() que desde java 8 tienen los objetos Collection. Devuelve un Stream secuencial también hay un método para **parallelStream()**.

En resumen un método stream que implementa un interface Collection devuelve un objeto Stream que da un flujo de datos ligado a la potencia de la programación funcional.

El método de programación tradicional y funcional son mejor según el contexto pero para el procesamiento de datos (filtrado ordenado etc..) la programación declarativa (funcional) parece la más conveniente.

El estilo de programación declarativa tipo SQL relacionada con la funcional de java:

Select->map()

From->stream

Where->filter()

Order by->sorted()

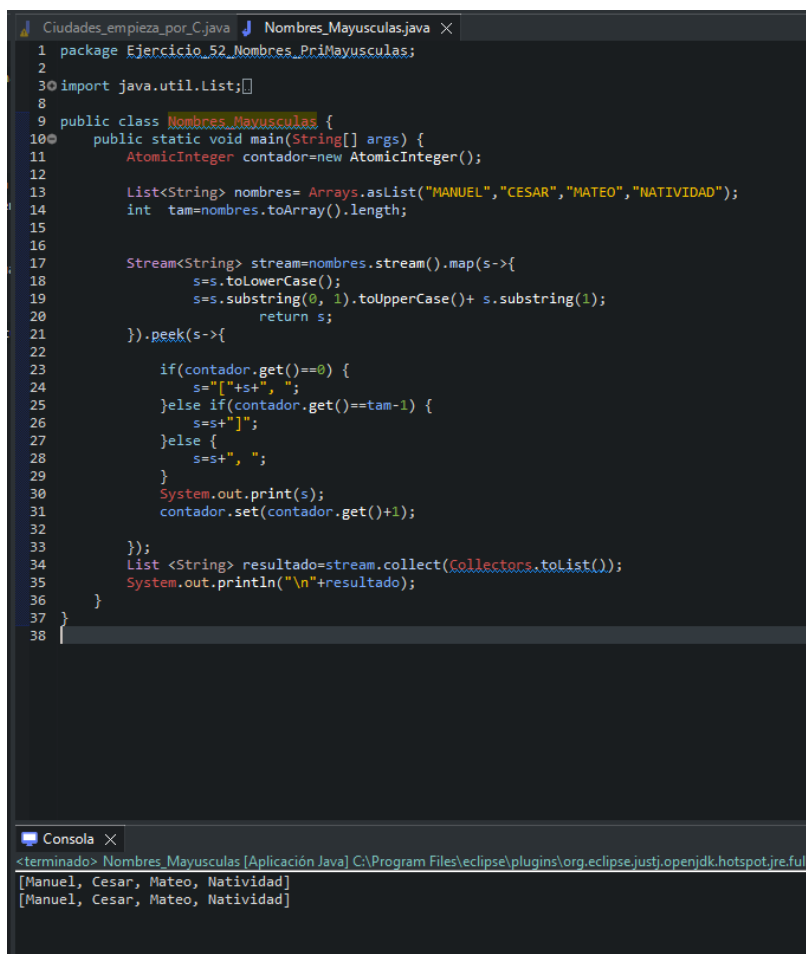
Además observa que en existe una interacción automática sobre cada miembro resultante del from similar a la interacción que se hace en el Stream.

Para este ejemplo necesitamos el mapeo de datos el SELECT , podemos transformar los elementos de un Stream , para lograrlo usamos algunos de los siguientes métodos:

- *map(Function<T,R>):Stream<R>*
- *mapToDouble(TodoubleFunction<T>:DoubleStream*
- *mapToInt(TointFunction<T>:IntStream*
- *mapToLong(ToLongFunction<T>):LongStream*

Map (Function<T,R>:Stream<R>

Retorna un Stream que contiene el resultado de aplicar la función pasada por parámetro a todo los elementos del Stream. **Transforma los elementos de Tipo T a tipo R**



```
1 package Ejercicio_52_Nombres_PriMayusculas;
2
3 import java.util.List;
4
5
6
7
8
9 public class Nombres_Mayusculas {
10     public static void main(String[] args) {
11         AtomicInteger contador=new AtomicInteger();
12
13         List<String> nombres= Arrays.asList("MANUEL","CESAR","MATEO","NATIVIDAD");
14         int tam=nombres.toArray().length;
15
16
17         Stream<String> stream=nombres.stream().map(s->{
18             s=s.toLowerCase();
19             s=s.substring(0, 1).toUpperCase()+ s.substring(1);
20             return s;
21         }).peek(s->{
22             if(contador.get()==0) {
23                 s="["+s+", ";
24             }else if(contador.get()==tam-1) {
25                 s=s+"]";
26             }else {
27                 s=s+", ";
28             }
29             System.out.print(s);
30             contador.set(contador.get()+1);
31         });
32
33         List <String> resultado=stream.collect(Collectors.toList());
34         System.out.println("\n"+resultado);
35     }
36 }
37
38 }
```

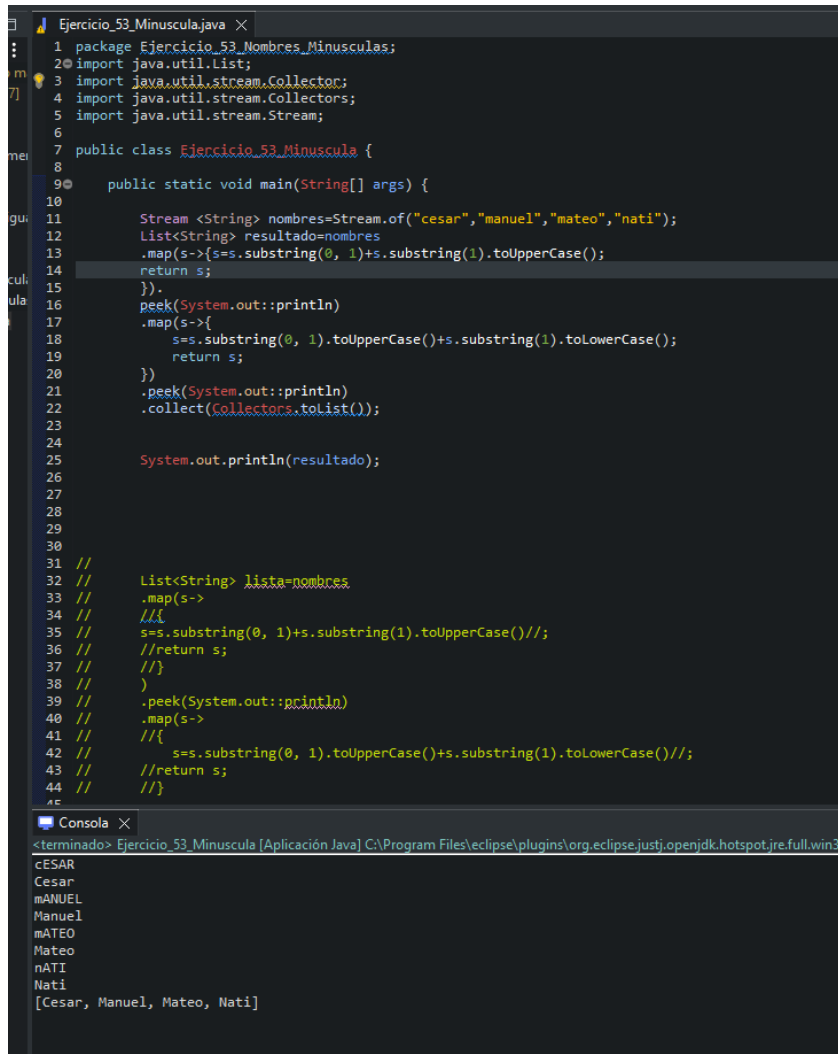
Consola X

<terminado> Nombres_Mayusculas [Aplicación Java] C:\Program Files\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full...

[Manuel, Cesar, Mateo, Natividad]

[Manuel, Cesar, Mateo, Natividad]

- 5.53 Crea un Stream de nombre de personas en minúscula, conviértelo en nombres con solo la primera en minúscula , muéstralo por pantalla , luego haz a la inversa (primera mayúscula y las siguientes minúsculas), muéstralo por pantalla y luego mételo dentro de una lista.



```
1 package Ejercicio_53_Nombres_Minusculas;
2 import java.util.List;
3 import java.util.stream.Collectors;
4 import java.util.stream.Collectors;
5 import java.util.stream.Stream;
6
7 public class Ejercicio_53_Minuscula {
8
9     public static void main(String[] args) {
10
11         Stream<String> nombres=Stream.of("cesar","manuel","mateo","nati");
12         List<String> resultado=nombres
13             .map(s->{s=s.substring(0, 1)+s.substring(1).toUpperCase();
14                 return s;
15             })
16             .peek(System.out::println)
17             .map(s->{
18                 s=s.substring(0, 1).toUpperCase()+s.substring(1).toLowerCase();
19                 return s;
20             })
21             .peek(System.out::println)
22             .collect(Collectors.toList());
23
24         System.out.println(resultado);
25
26
27
28
29
30
31 //
32 // List<String> lista=nombres
33 // .map(s->
34 // //{
35 // s=s.substring(0, 1)+s.substring(1).toUpperCase();//;
36 // //return s;
37 // //}
38 // )
39 // .peek(System.out::println)
40 // .map(s->
41 // //{
42 // s=s.substring(0, 1).toUpperCase()+s.substring(1).toLowerCase();//;
43 // //return s;
44 // //}
45 // }
```

Console

```
<terminado> Ejercicio_53_Minuscula [Aplicación Java] C:\Program Files\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
cESAR
Cesar
mANUEL
Manuel
mATEO
Mateo
nATI
Nati
[Cesar, Manuel, Mateo, Nati]
```

No imprime por orden todo junto , Se pude????? Sorted

Peek permite ver el estado del Stream en un determinado momento de la aplicación.

5.54 Ejercicio 54: Crea un Stream de nombre de personas en mayúsculas, conviértelo en nombres con solo la primera en mayúscula (el resto en minúsculas), y muestra por pantalla, sólo los nombres que empiecen por J.

```

1 package Ejercicio_54.Nombres_distintos_54;
2 import java.util.stream.Stream;
3 import java.util.List;
4 import java.util.stream.Collectors;
5
6 public class Didtintos_54 {
7
8
9     public static void main(String args[]) {
10         Stream<String> personas=Stream.of("PEPE", "JuAnjo", "CESAR", "MATEO", "MANUEL", "JUAN", "NATIVIDAD", "JAÍMOLÁS");
11         List<String> personas=personas
12             .map(s->Character.toUpperCase(s.charAt(0)) + s.substring(1).toLowerCase())
13             .filter(s->s.startsWith("J")).collect(Collectors.toList());
14
15         System.out.println(personas);
16     }
17 }
18
19
20
21
22

```

Console: <terminado> Didtintos_54 [Aplicación Java] C:\Program Files\ eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.2.v20220201-1208\jre\bin\javaw.exe (19 jun 2022 19:34:42 - 19:34:42) [pid: 10220]
[Juanjo, Juan, Jaímolás]

Filter(Predicate<T>):Stream<T> devuelve solo el Stream que contiene los elementos que cumplen el predicado en este caso startsWith("J");

5.55 Ejercicio 55: Crea un Stream de nombre de personas en mayúsculas, conviértelo en nombres con solo la primera en mayúscula (el resto en minúsculas), y usa las siguientes operaciones terminales (puedes inventarte tu la condición... que empiece por L, que acabe en "o",...): `anyMatch` `noneMatch` `allMatch`

```

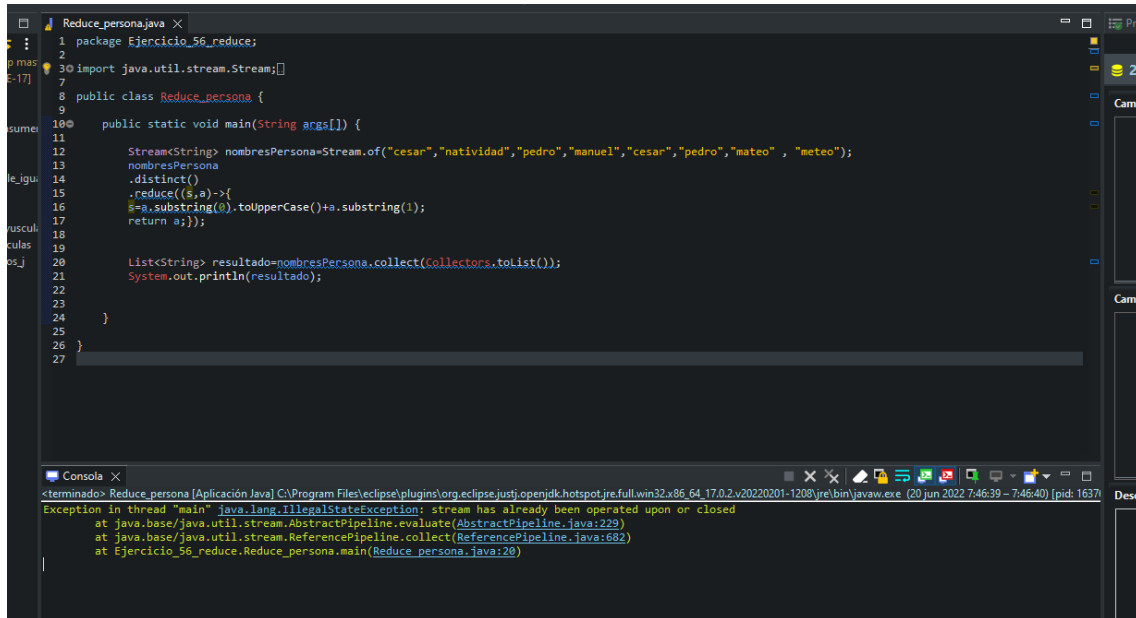
1 package Ejercicio_55.terminales;
2 import java.util.List;
3 import java.util.stream.Stream;
4
5 public class Terminales {
6
7     public static void main(String[] args) {
8
9
10         Stream<String> nombres=Stream.of("CESAR", "MANUEL", "LOLO", "LOLOQ");
11
12         boolean empiezaPorM=nombres
13             .map(s->s.substring(0, 1).toUpperCase()+s.substring(1).toLowerCase())
14             .anyMatch(s->s.startsWith("M"));
15
16         System.out.println("Alguna de los nombres empieza por M? "+empiezaPorM);
17
18         boolean masDe5letras=nombres
19             .peek(s->System.out.println(s+" "))
20             .map(s->s.substring(0, 1).toUpperCase()+s.substring(1).toLowerCase()).peek(System.out::println)
21             .allMatch(s->s.length()>5);
22
23         System.out.println("Todos los nombres son mayores o iguales a 4 letras? "+masDe5letras);
24     }
25 }
26
27
28
29

```

Console: <terminado> Terminales [Aplicación Java] C:\Program Files\ eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.2.v20220201-1208\jre\bin\javaw.exe (19 jun 2022 21:36:21 - 21:36:22) [pid: 6192]
CESAR
MANUEL
LOLO
LOLOQ
Todos los nombres son mayores o iguales a 4 letras? true

CERRAR el Stream mostrar el original no se puede, volver a usar un stream

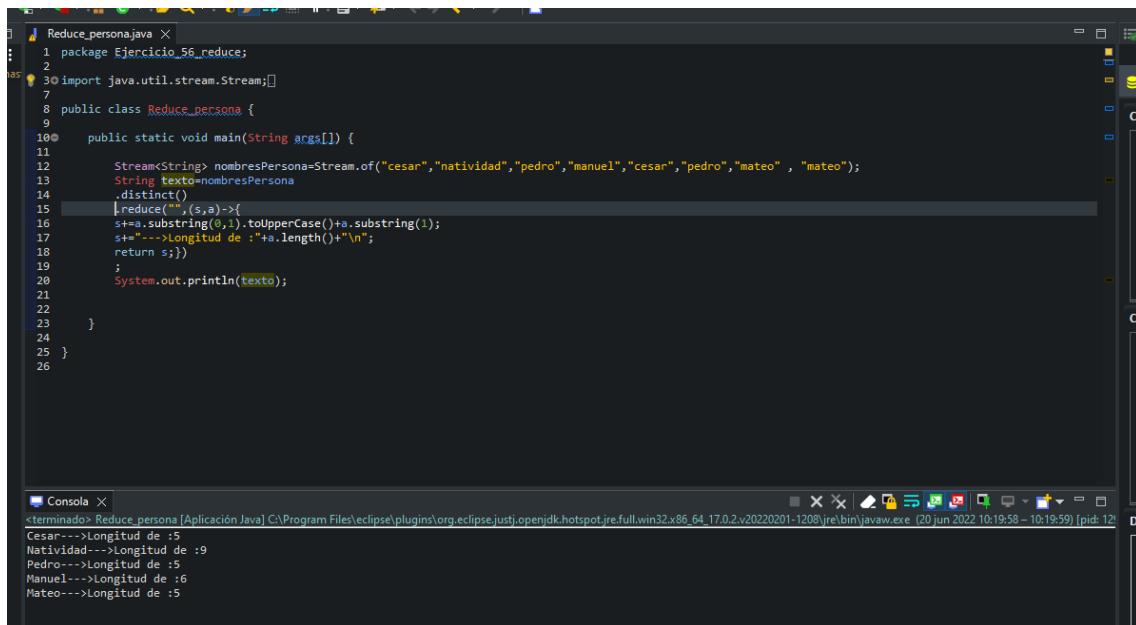
5.56 Crea un Stream de nombre de personas en minúscula, usa la operación `distinct()` y muéstralo por pantalla el nombre con la primera en mayúsculas
🔗 Añádele la operación `reduce()`



```
1 package Ejercicio_56_reduce;
2
3 import java.util.stream.Stream;
4
5 public class Reduce_persona {
6
7     public static void main(String args[]) {
8
9         Stream<String> nombresPersona=Stream.of("cesar","natividad","pedro","manuel","cesar","pedro","mateo" , "mateo");
10        nombresPersona
11        .distinct()
12        .reduce((s,a)->{
13            s=s.substring(0,1).toUpperCase()+a.substring(1);
14            return s;});
15
16        List<String> resultado=nombresPersona.collect(Collectors.toList());
17        System.out.println(resultado);
18
19    }
20
21 }
```

Console

```
<terminado> Reduce_persona [Aplicación Java] C:\Program Files\ eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.2.v20220201-1208\jre\bin\javaw.exe (20 jun 2022 7:46:39 - 7:46:40) [pid: 1637]
Exception in thread "main" java.lang.IllegalStateException: stream has already been operated upon or closed
    at java.base/java.util.stream.AbstractPipeline.evaluate(AbstractPipeline.java:229)
    at java.base/java.util.stream.ReferencePipeline.collect(ReferencePipeline.java:682)
    at Ejercicio_56_reduce.Reduce_persona.main(Reduce_persona.java:20)
```



```
1 package Ejercicio_56_reduce;
2
3 import java.util.stream.Stream;
4
5 public class Reduce_persona {
6
7     public static void main(String args[]) {
8
9         Stream<String> nombresPersona=Stream.of("cesar","natividad","pedro","manuel","cesar","pedro","mateo" , "mateo");
10        String texto=nombresPersona
11        .distinct()
12        .reduce("",(s,a)->{
13            s+=a.substring(0,1).toUpperCase()+a.substring(1);
14            s+="--->Longitud de :"+a.length()+"\n";
15            return s;});
16
17        System.out.println(texto);
18
19    }
20
21 }
```

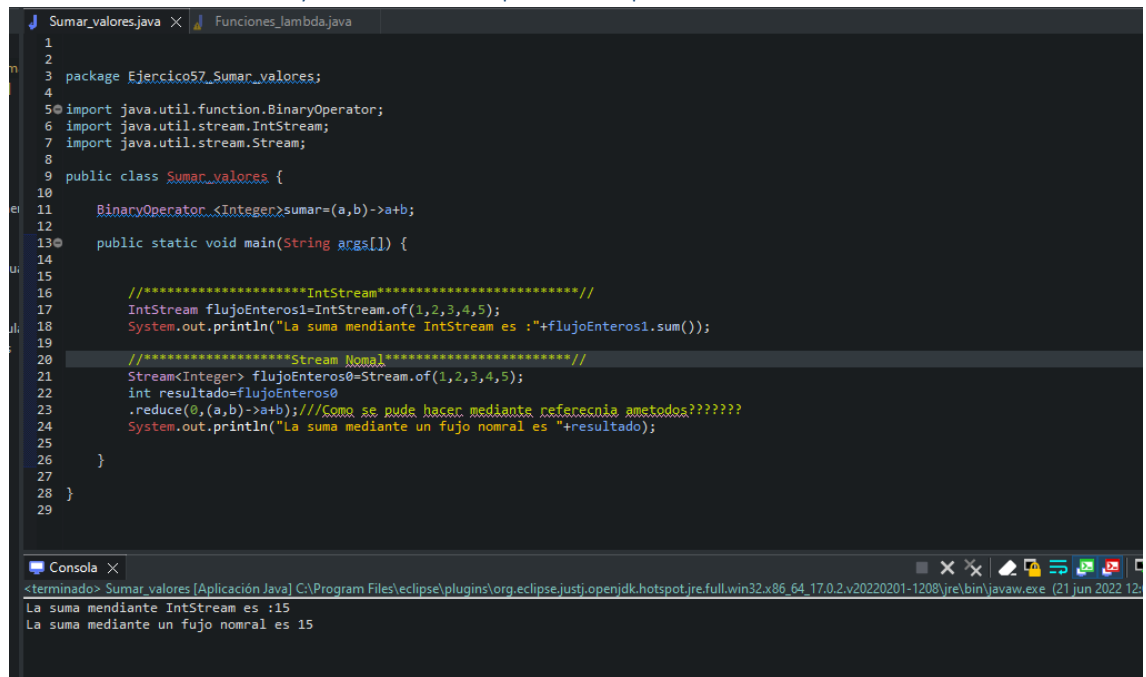
Console

```
<terminado> Reduce_persona [Aplicación Java] C:\Program Files\ eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.2.v20220201-1208\jre\bin\javaw.exe (20 jun 2022 10:19:58 - 10:19:59) [pid: 125]
Cesar--->Longitud de :5
Natividad--->Longitud de :9
Pedro--->Longitud de :5
Manuel--->Longitud de :6
Mateo--->Longitud de :5
```

5.57 Ejercicio 57

Desde java 8 se ofrece la posibilidad de crear flujos a partir de tres tipos primitivos: int, Long y double. Como `Stream<T>` es una interfaz genérica y no hay forma de usar primitivas como parámetro de tipo genérico, se crearon tres tipos de interfaces especiales: `IntStream`, `LongStream`, `DoubleStream`

5.57.1 Crea un Stream de int y añádele una operación que sume sus valores.



```
1 Sumar_valores.java x Funciones_lambda.java
2
3 package Ejercicio57_Sumar_valores;
4
5 import java.util.function.BinaryOperator;
6 import java.util.stream.IntStream;
7 import java.util.stream.Stream;
8
9 public class Sumar_valores {
10
11     BinaryOperator<Integer> sumar=(a,b)->a+b;
12
13     public static void main(String args[]) {
14
15         //*****IntStream*****
16         IntStream flujoEnteros1=IntStream.of(1,2,3,4,5);
17         System.out.println("La suma mediante IntStream es :"+flujoEnteros1.sum());
18
19         //*****Stream Normal*****
20         Stream<Integer> flujoEnteros0=Stream.of(1,2,3,4,5);
21         int resultado=flujoEnteros0
22             .reduce(0,(a,b)->a+b); //Como se puede hacer mediante referencia a metodos????
23         System.out.println("La suma mediante un flujo normal es "+resultado);
24     }
25 }
26
27
28
29
```

Console x

```
<terminado> Sumar_valores [Aplicación Java] C:\Program Files\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.exe (21 jun 2022 12:
La suma mediante IntStream es :15
La suma mediante un flujo normal es 15
```

5.57.2 Crea un IntStream con números entre el 1 y el 50 .

El método ***range(int startInclusive, int endExclusive)*** crea un flujo ordenado desde el primer parámetro hasta el segundo parámetro. Incrementa el valor de los elementos posteriores con el paso igual a 1. El resultado no incluye el último parámetro, es solo un límite superior de la secuencia.

El ***método rangeClosed(int startInclusive, int endInclusive)*** hace lo mismo con solo una diferencia, se incluye el segundo elemento. Podemos usar estos dos métodos para generar cualquiera de los tres tipos de flujos de primitivas.

```
IntStream flujoEnteros50=IntStream.rangeClosed(0, 50);
flujoEnteros50
    .forEach(s->System.out.print(s+", "));
```

5.57.3 A partir de IntStream anterior crea un IntStream con número aleatorios entre el valor y el valor + 50. Con el IntStream resultado, muestra por pantalla algunas de sus estadísticas (máximo valor, mínimo, sumatorio, contar, media,...) .

```

1 package Ejercicio57_Sumar_valores;
2 import java.util.List;
3 import java.util.OptionalDouble;
4 import java.util.OptionalInt;
5 import java.util.function.BinaryOperator;
6 import java.util.stream.Collectors;
7 import java.util.stream.IntStream;
8 import java.util.stream.Stream;
9 import Ejercicio00_Aleatorios;
10
11 public class Sumar_valores {
12
13     BinaryOperator<Integer> sumar=(a,b)->a+b;
14
15     public static void main(String args[]) {
16         //*****IntStream*****
17         IntStream flujoEnteros1=IntStream.of(1,2,3,4,5);
18         System.out.println("La suma mediante IntStream es : "+flujoEnteros1.sum());
19
20         //*****Stream Normal*****
21         Stream<Integer> flujoEnteros0=Stream.of(1,2,3,4,5);
22         int resultado=flujoEnteros0
23             .reduce(0,(a,b)->a+b); //Como se puede hacer mediante referencia a metodos????
24         System.out.println("La suma mediante un flujo normal es : "+resultado);
25
26         //*****Crear un Stream del 0 al 50*****
27         List<Integer> lista=IntStream.rangeClosed(1, 50).boxed().collect(Collectors.toList());
28
29         String txt="";
30         for(int i=0;i<lista.size();i++) {
31             txt+=lista.get(i)+ ((i<lista.size()-1)?", ":"");
32         }
33         System.out.println(txt);
34
35         List<Integer> enteros=lista.stream().mapToInt(x->x)
36             .map(x->(int)((Math.random()*(50-x))+x)).boxed().collect(Collectors.toList());
37
38         enteros.stream().mapToInt(x->x).forEach(x->
39             System.out.print(x+" "));
40
41         int suma=enteros.stream().mapToInt(x->x).sum();
42         Double avg=enteros.stream().mapToInt(x->x).average().getAsDouble();
43         int max=enteros.stream().mapToInt(x->x).max().getAsInt();
44         long contar=enteros.stream().mapToInt(x->x).count();
45         OptionalInt min=enteros.stream().mapToInt(x->x).min();
46         System.out.print("\nLa suma es : "+suma);
47         System.out.print("\nEl máximo es : "+max);
48         System.out.print("\nEl mínimo es : "+min);
49         System.out.print("\nLa promedio es : "+avg);
50         System.out.print("\nEl numero es : "+contar);
51         System.out.print("\nEl promedio x el numero : "+(avg*contar));
52     }
53 }
54
55

```

Consola

```

terminado> Sumar_valores [Aplicación Java] C:\Program Files\ eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.2.v20220201-1208\jre\bin\javaw.exe (22 jun 2022 0:33:55 - 0:33:56) [pid: 12868]
La suma mediante IntStream es : 15
La suma mediante un flujo normal es : 15
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50
13,45,3,12,48,17,16,45,31,23,39,27,35,24,34,47,33,47,39,31,30,34,38,28,43,31,38,47,42,47,31,41,36,37,49,44,41,49,46,45,46,49,48,47,46,46,47,48,49,50,
La suma es : 1912
El máximo es : 50
El mínimo es : OptionalInt[3]
La promedio es : 38.24
El numero es : 50
El promedio x el numero : 1912.0

```

Buscar OptionalInt, Boxed()

5.58 Crea un Stream de nombres y obtén un IntStream con la longitud de cada uno de ellos. Crea un IntSummaryStatistics para obtener las estadísticas (máximo valor, mínimo, sumatorio, contar, media,...)

```
Nombres_IntStream.java X
1 package Ejercicio_58_Stream_IntStream;
2
3 import java.util.List;
4 import java.util.concurrent.atomic.AtomicInteger;
5 import java.util.ArrayList;
6 import java.util.IntSummaryStatistics;
7 import java.util.stream.IntStream;
8 public class Nombres_IntStream {
9
10     public static void main (String args[]) {
11         List<String> nombres=new ArrayList<>();
12         nombres.add("Cesar");
13         nombres.add("Mateo");
14         nombres.add("Natividad");
15         nombres.add("Manuel");
16         AtomicInteger contador = new AtomicInteger(0);
17
18         contador.set(0);
19
20         System.out.println(nombres);
21         nombres.stream()
22             .map(x->x.length())
23             .mapToInt(x->x)
24             .forEach(x->{
25                 String txt;
26                 if(contador.get()==0) {
27                     txt="["+Integer.toString(x) + (((nombres.size()==contador.incrementAndGet())?"":","));}else {
28                         txt=Integer.toString(x) + (((nombres.size()==contador.incrementAndGet())?"":","));
29                     }
30                 System.out.print(txt);
31             });
32         System.out.println("\n");
33         IntSummaryStatistics estadisticas=nombres.stream().mapToInt(x->x.length()).summaryStatistics();
34         System.out.println(estadisticas);
35
36     }
37 }
38
39 }
40
```

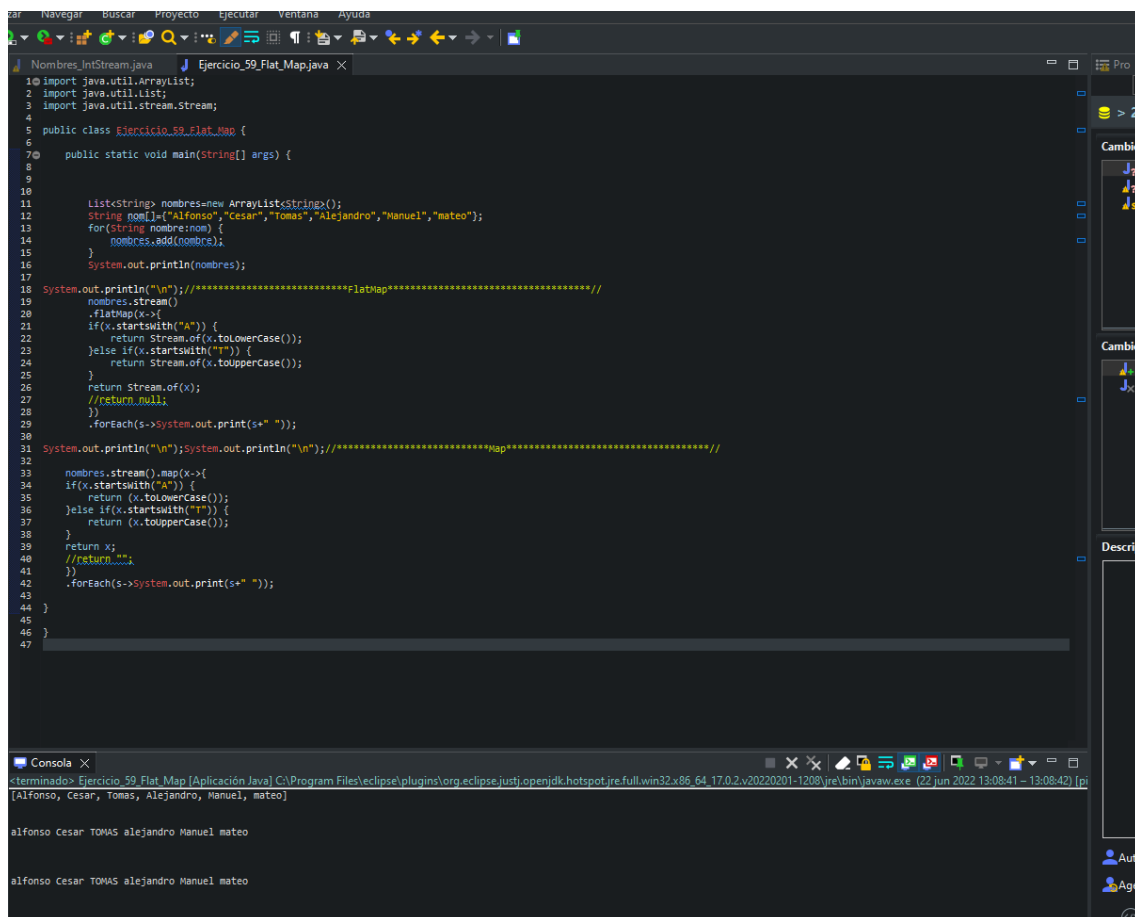
```
Consola X
<terminado> Nombres_IntStream [Aplicación Java] C:\Program Files\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v202202
[Cesar, Mateo, Natividad, Manuel]
{5,5,9,6}

IntSummaryStatistics{count=4, sum=25, min=5, average=6,250000, max=9}
```

5.59 Ejercicio 59: Con un Stream de nombres y usando la operación flatMap devuelve el nombre en mayúsculas si el nombre empieza por A y devuelve el nombre en minúsculas si empieza por T (por ejemplo, lo puedes adaptar a tu ejemplo)

La diferencia es que `map()` devuelve el mismo número de elementos que el Stream de entrada ya que es simplemente una proyección de los elementos de entrada. Es decir cada elemento de entrada se transforma en un elemento de salida.

Por otro lado `.flatMap()`, proyecta una lista de elementos de cada elemento original y los concatena en un único *stream*.



```
10 import java.util.ArrayList;
11 import java.util.List;
12 import java.util.stream.Stream;
13
14 public class Ejercicio_59_Flat_Map {
15
16     public static void main(String[] args) {
17
18         List<String> nombres = new ArrayList<String>();
19         String nom[] = {"Alfonso", "Cesar", "Tomas", "Alejandro", "Manuel", "mateo"};
20         for (String nombre : nom) {
21             nombres.add(nombre);
22         }
23         System.out.println(nombres);
24
25         System.out.println("\n"); //*****flatMap*****
26         Stream<String> stream = nombres.stream();
27         stream.flatMap(x -> {
28             if (x.startsWith("A")) {
29                 return Stream.of(x.toLowerCase());
30             } else if (x.startsWith("T")) {
31                 return Stream.of(x.toUpperCase());
32             }
33             return Stream.of(x);
34         }).forEach(s -> System.out.print(s + " "));
35
36         System.out.println("\n"); System.out.println("\n"); //*****map*****
37         nombres.stream().map(x -> {
38             if (x.startsWith("A")) {
39                 return x.toLowerCase();
40             } else if (x.startsWith("T")) {
41                 return x.toUpperCase();
42             }
43             return x;
44         }).forEach(s -> System.out.print(s + " "));
45     }
46 }
47
```

Console

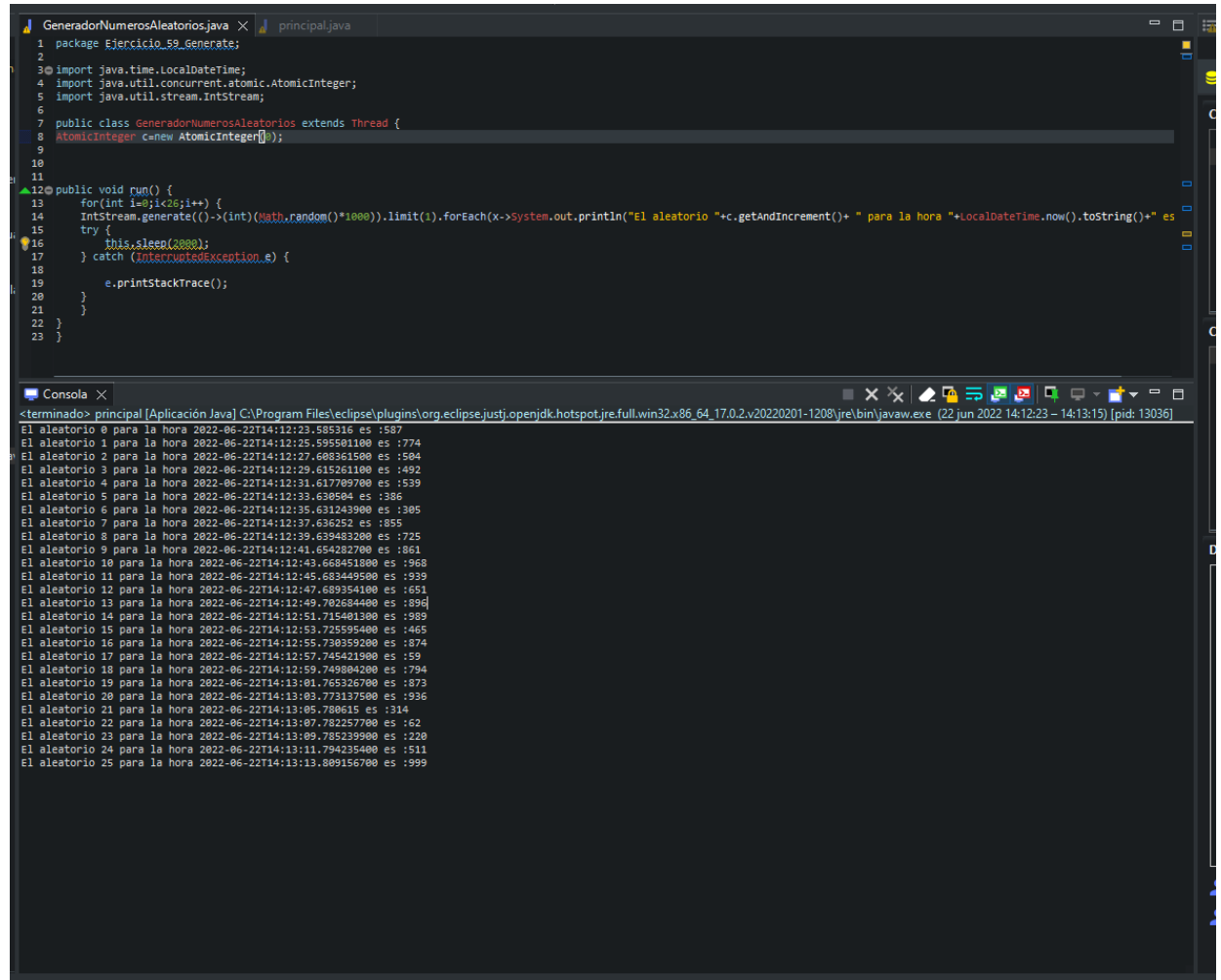
```
terminado> Ejercicio_59_Flat_Map [Aplicación Java] C:\Program Files\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.2.v20220201-1200\jre\bin\java.exe (22 jun 2022 13:08:41 - 13:08:42) [pi
[Alfonso, Cesar, Tomas, Alejandro, Manuel, mateo]

alfonso Cesar TOMAS alejandro Manuel mateo

alfonso Cesar TOMAS alejandro Manuel mateo
```

En este ejemplo resulta inútil flatMap???

5.60 Usa la función generate() para mostrar números aleatorios por pantalla cada X minisegundos (se duerme el hilo durante ese tiempo), con un límite de 25 números



The screenshot shows the Eclipse IDE with two tabs: 'GeneradorNumerosAleatorios.java' and 'principal.java'. The 'GeneradorNumerosAleatorios.java' tab is active, displaying the following code:

```
1 package Ejercicio_59_Generate;
2
3 import java.time.LocalDateTime;
4 import java.util.concurrent.atomic.AtomicInteger;
5 import java.util.stream.IntStream;
6
7 public class GeneradorNumerosAleatorios extends Thread {
8     AtomicInteger c=new AtomicInteger(0);
9
10
11
12 public void run() {
13     for(int i=0;i<25;i++) {
14         IntStream.generate(()->(int)(Math.random()*1000)).limit(1).forEach(x->System.out.println("El aleatorio "+c.getAndIncrement()+" para la hora "+LocalDateTime.now().toString()+" es "+x));
15     }
16     try {
17         this.sleep(2000);
18     } catch (InterruptedException e) {
19         e.printStackTrace();
20     }
21 }
22 }
23 }
```

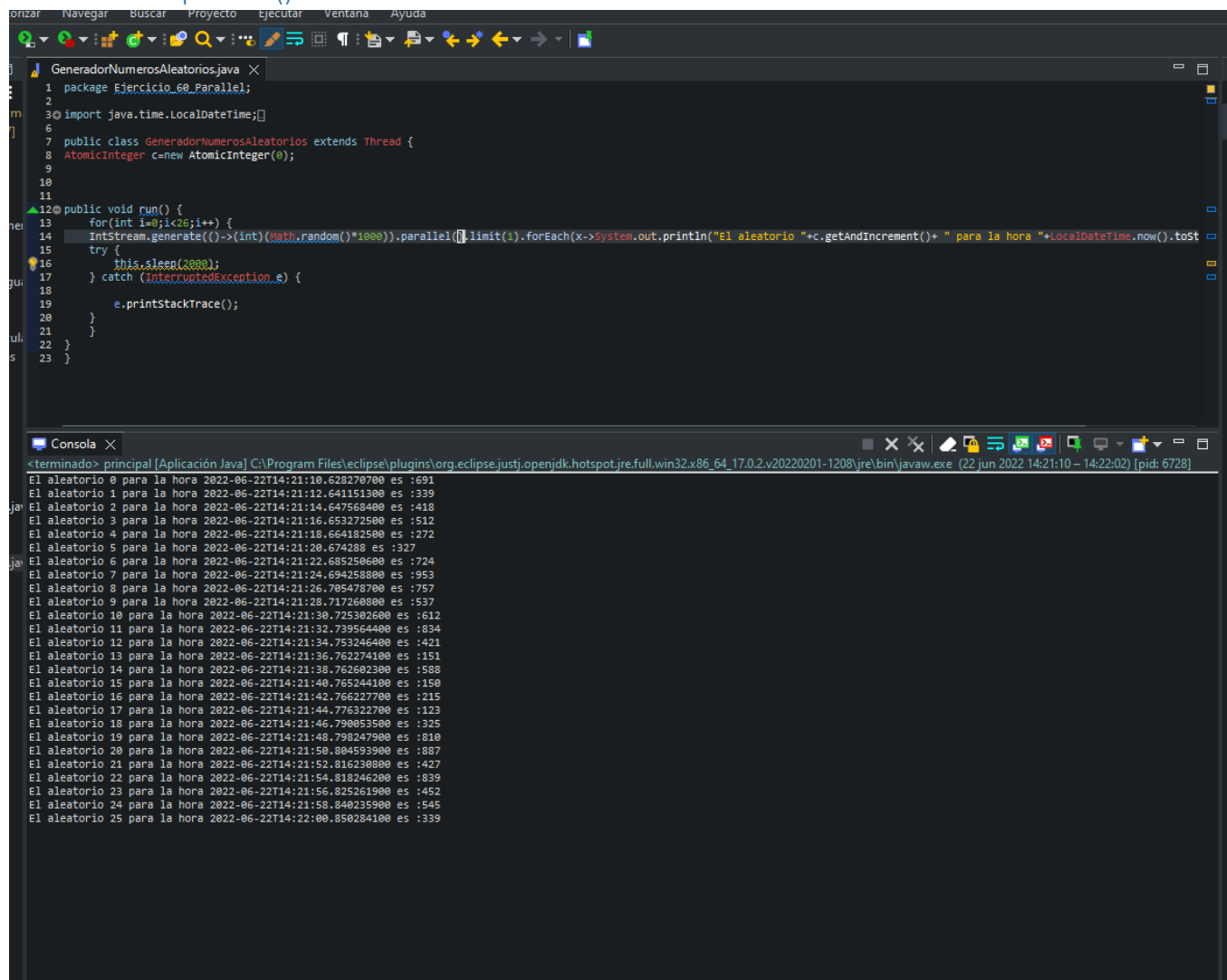
The 'principal.java' tab is also visible, showing a simple main method that calls the 'run' method of the 'GeneradorNumerosAleatorios' class.

```
1 public class principal {
2     public static void main(String[] args) {
3         GeneradorNumerosAleatorios g = new GeneradorNumerosAleatorios();
4         g.run();
5     }
6 }
```

The console output shows the execution of the program, displaying 25 random numbers, each preceded by the text 'El aleatorio' and the current time. The output is as follows:

```
<terminado> principal [Aplicación Java] C:\Program Files\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.exe (22 jun 2022 14:12:23 - 14:13:15) [pid: 13036]
El aleatorio 0 para la hora 2022-06-22T14:12:23.585316 es :587
El aleatorio 1 para la hora 2022-06-22T14:12:25.595591100 es :774
El aleatorio 2 para la hora 2022-06-22T14:12:27.608361500 es :504
El aleatorio 3 para la hora 2022-06-22T14:12:29.615261100 es :492
El aleatorio 4 para la hora 2022-06-22T14:12:31.617709700 es :539
El aleatorio 5 para la hora 2022-06-22T14:12:33.630504 es :386
El aleatorio 6 para la hora 2022-06-22T14:12:35.631243900 es :305
El aleatorio 7 para la hora 2022-06-22T14:12:37.636252 es :855
El aleatorio 8 para la hora 2022-06-22T14:12:39.639483200 es :725
El aleatorio 9 para la hora 2022-06-22T14:12:41.654282700 es :861
El aleatorio 10 para la hora 2022-06-22T14:12:43.668451800 es :968
El aleatorio 11 para la hora 2022-06-22T14:12:45.683449500 es :939
El aleatorio 12 para la hora 2022-06-22T14:12:47.689354100 es :651
El aleatorio 13 para la hora 2022-06-22T14:12:49.702694400 es :896
El aleatorio 14 para la hora 2022-06-22T14:12:51.715401300 es :989
El aleatorio 15 para la hora 2022-06-22T14:12:53.725595400 es :465
El aleatorio 16 para la hora 2022-06-22T14:12:55.730359200 es :874
El aleatorio 17 para la hora 2022-06-22T14:12:57.745421900 es :59
El aleatorio 18 para la hora 2022-06-22T14:12:59.749804200 es :794
El aleatorio 19 para la hora 2022-06-22T14:13:01.755326700 es :873
El aleatorio 20 para la hora 2022-06-22T14:13:03.773137500 es :936
El aleatorio 21 para la hora 2022-06-22T14:13:05.780615 es :314
El aleatorio 22 para la hora 2022-06-22T14:13:07.782257700 es :62
El aleatorio 23 para la hora 2022-06-22T14:13:09.785239900 es :220
El aleatorio 24 para la hora 2022-06-22T14:13:11.794235400 es :511
El aleatorio 25 para la hora 2022-06-22T14:13:13.809156700 es :999
```

5.61 Usa la función generate() para mostrar números aleatorios por pantalla cada X minisegundos (muestra este tiempo también por pantalla), con un límite de 25 números. Haz lo mismo que el punto anterior pero usando la función parallel().



```
1 package Ejercicio_60_Parallel;
2
3 import java.time.LocalDateTime;
4
5
6
7 public class GeneradorNumerosAleatorios extends Thread {
8     AtomicInteger c=new AtomicInteger(0);
9
10
11
12 public void run() {
13     for(int i=0;i<25;i++) {
14         IntStream.generate(()->(int)(Math.random()*1000)).parallel().limit(1).forEach(x->System.out.println("El aleatorio "+c.getAndIncrement()+" para la hora "+LocalDateTime.now().toString()));
15         try {
16             this.sleep(2000);
17         } catch (InterruptedException e) {
18             e.printStackTrace();
19         }
20     }
21 }
22 }
23 }
```

<terminado> principal [Aplicación Java] C:\Program Files\ eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.exe (22 jun 2022 14:21:10 – 14:22:02) [pid: 6728]

El aleatorio 0 para la hora 2022-06-22T14:21:10.628270700 es :691
El aleatorio 1 para la hora 2022-06-22T14:21:12.641151300 es :339
El aleatorio 2 para la hora 2022-06-22T14:21:14.647568400 es :418
El aleatorio 3 para la hora 2022-06-22T14:21:16.653272500 es :512
El aleatorio 4 para la hora 2022-06-22T14:21:18.664182500 es :272
El aleatorio 5 para la hora 2022-06-22T14:21:20.674288 es :327
El aleatorio 6 para la hora 2022-06-22T14:21:22.685258600 es :724
El aleatorio 7 para la hora 2022-06-22T14:21:24.694258800 es :953
El aleatorio 8 para la hora 2022-06-22T14:21:26.705478700 es :757
El aleatorio 9 para la hora 2022-06-22T14:21:28.717268800 es :537
El aleatorio 10 para la hora 2022-06-22T14:21:30.725382600 es :612
El aleatorio 11 para la hora 2022-06-22T14:21:32.739564400 es :834
El aleatorio 12 para la hora 2022-06-22T14:21:34.753246400 es :421
El aleatorio 13 para la hora 2022-06-22T14:21:36.762274100 es :151
El aleatorio 14 para la hora 2022-06-22T14:21:38.762602300 es :588
El aleatorio 15 para la hora 2022-06-22T14:21:40.765244100 es :150
El aleatorio 16 para la hora 2022-06-22T14:21:42.766227700 es :215
El aleatorio 17 para la hora 2022-06-22T14:21:44.776322700 es :123
El aleatorio 18 para la hora 2022-06-22T14:21:46.798053500 es :325
El aleatorio 19 para la hora 2022-06-22T14:21:48.798247900 es :810
El aleatorio 20 para la hora 2022-06-22T14:21:50.804593900 es :887
El aleatorio 21 para la hora 2022-06-22T14:21:52.816230800 es :427
El aleatorio 22 para la hora 2022-06-22T14:21:54.819246200 es :839
El aleatorio 23 para la hora 2022-06-22T14:21:56.825261900 es :452
El aleatorio 24 para la hora 2022-06-22T14:21:58.8480235900 es :545
El aleatorio 25 para la hora 2022-06-22T14:22:00.850284100 es :339