Expresiones lambda

Interfaz funcional

>Interfaz que proporciona un único método abstracto

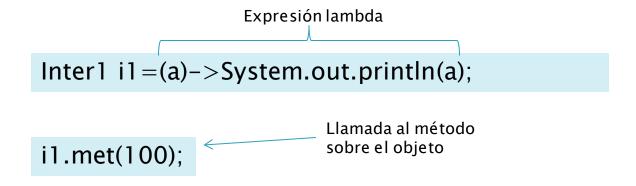
```
public interface Runnable{
  void run();
}
```

```
public interface Inter2{
  boolean process(int n, String pt);
  static void print(){}
}
```

```
public interface Inter1{
  void met(int data);
  default int res(){return 1;}
}
```

¿Qué es una expresión lambda?

- >Implementación de una interfaz funcional
- >Proporciona el código del único método abstracto de la interfaz, a la vez que genera un objeto que implementa la misma



Sintaxis

>Una expresión lambda tiene dos partes, la lista de parámetros del método y la implementación:

parametros->implementación

- >Los parámetros pueden indicar o no el tipo
- >La lista de parámetros se puede indicar o no entre paréntesis (obligatorio si hay dos o más) y también si se indica el tipo
- En caso de devolver un resultado, la implementación puede omitir la palabra *return* si consta de una sola instrucción

Ejemplos

CORRECTO

```
()->3
(int a)->System.out.println("hello")
x->x*x
(n1,n2)->{
    n1+=20;
    System.out.println(n1+n2);
}
```

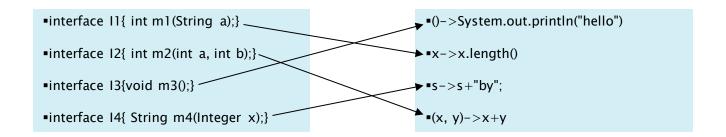
INCORRECTO

```
->3
int a->System.out.println("hello")
x->return x*x //se requieren llaves con return
n1,n2->System.out.println(n1+n2)
```

Revisión conceptos



Relaciona cada interfaz funcional con su correspondiente expresión lambda que la implementa:



Respuesta

Inferencia de tipos

Es posible inferir el tipo en los parámetros de las expresiones lambda:

```
(var a)->System.out.println(a)
```

>Aunque no se puede combinar inferencia de tipos y tipos específicos en una misma expresión:

```
(var a, int c)->a+c //error de compilación
```

>¿Qué utilidad tiene si ya es posible no indicar el tipo en los parámetros?

```
(@NotNull var c)->... //ok
(@NotNull c)->... //error de compilación
```

Comparator con lambdas

- >Interfaz utilizada para la ordenación de colecciones y arrays
- >Al ser funcional, se puede implementar con lambdas:

```
List<String> textos=new ArrayList<>();
textos.add("mi texto"); textos.add("hello"); textos.add("es el más largo");
//ordenación de la lista de textos por longitud
textos.sort((a,b)->a.length()-b.length());
//recorrido y presentación de datos
for(String s:textos){
    System.out.println(s));
}
    //hello
    //mi texto
    //es el más largo
```

Revisión conceptos



Dada la siguiente lista, mostrar los nombres de las personas ordenadas por edad

```
List<Persona> personas=new ArrayList<>(List.of(new Persona("marco",34), new Persona("ana",28), new Persona("bea",41)));
```



```
personas.sort((a,b)->a.getEdad()-b.getEdad());
//recorrido y presentación de datos
for(Persona p:personas){
    System.out.println(p.getNombre()));
}
```