

Trabajo Práctico 2.1 Repaso sobre Expresiones Lambda

1. Dadas las siguientes interfaces:

```
interface A {
          void metodo();
}
interface B {
          void metodo(String b);
}
interface C {
          boolean metodo(String c);
}
interface D<T, R> {
          R metodo(T c);
}
```

la siguiente clase:

```
public class AprendiendoLambdas {
    public void unMetodo(A a) {
        a.metodo();
    }

    public void unMetodo(B b) {
        b.metodo("unString");
    }

    public void unMetodo(C c) {
        System.out.println(c.metodo("otroString")?"true": "false");
    }

    public void unMetodo(D<Long, Long> d) {
        d.metodo(10L);
    }
}
```

Y la clase Main:

```
public class Main {
    public static void main(String[] args) {
```



```
AprendiendoLambdas a = new AprendiendoLambdas();
a.unMetodo((b) → { System.out.println("abcd" + b);});
a.unMetodo(() → System.out.println("abcd"));
a.unMetodo((variable) → { System.out.println("abcd");});
a.unMetodo((String variable) → { System.out.println("abcd"); return true;});
a.unMetodo((Long variable) -> {

System.out.println("abcd");
return 10L;
});
}
```

Indique qué metodos de la clase AprendiendoLambdas se invocan en cada caso. Explique claramente porque.

2. Dada la interfaz:

```
interface C {
    boolean metodo(String c);
}
```

La clase:

```
public class AprendiendoLambdas {
        public void unMetodo(C c) {
            System.out.println(c.metodo("abcd"));
        }
}
```

Escriba una clase Main para poder llamar al metodo AprendiendoLambdas#unMetodo(C c) de la siguiente forma:

- a. Utilizando un lambda dado que imprima true si el largo del String es par, false en caso contrario.
 b. Utilizando un lambda dado que imprima true si el String empieza con a minúscula, false en caso contrario.
- 3. Dadas la clase Persona:

```
public class Persona {
    private String nombre;
    private String apellido;

public Persona(String nombre, String apellido) {
        this.nombre = nombre;
}
```



```
this.apellido = apellido;
}

public String nombre() {
    return nombre;
}

public String apellido() {
    return apellido;
}

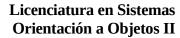
@Override
public String toString() {
    return "Persona [nombre=" + nombre + ", apellido=" + apellido + "]";
}
```

Observe los siguientes métodos:

```
//filtra la lista de personas devolviendo otra lista con
//solo aquellas cuyo nombre comienza con E
public List<Persona> nombresQueEmpiezanConE(List<Persona> p) {
        List<Persona> resultado = new ArrayList<>();
        for (Persona persona : p) {
            if (persona.nombre().startsWith("E")) {
                resultado.add(persona);
            }
        }
        return resultado;
}
```

```
public List<Persona> nombresCuyaCantidadDeLetrasEsPar(List<Persona> p) {
    List<Persona> resultado = new ArrayList<>();
    for (Persona persona : p) {
        if (persona.nombre().length() % 2 == 0) {
            resultado.add(persona);
        }
    }
    return resultado;
}
```

¿No son practicamente iguales ambos métodos? ¿Hay código duplicado? ¿Cómo implementaría un único metodo que permita resolver ambos requerimientos (nombres que comiencen con E y nombres cuya cantidad de letras es Par), sin tener código duplicado?.





Ayuda:

Observe que los métodos son iguales, salvo la condición de la sentencia *if*. Utilice lambdas para pasar dicha condición por parámetro al método.

```
AprendiendoLambdas a = new AprendiendoLambdas();

egthinspace{1mm} a.unMetodo((b) \rightarrow { System.out.println("abcd" + b);});
        \Rightarrowa.unMetodo(() \rightarrow System.out.println("abcd"));
        \overline{\phantom{a}} a.unMetodo((variable) \rightarrow {System.out.println("abcd");});
        \neg a.unMetodo((String variable) \rightarrow {System.out.println("abcd"); return true;});
  5 ----- a.unMetodo((Long variable) -> {
                  System.out.println("abcd");
                  return 10L;
         });
     }
1_ EN este CASO SE TOMA UN PARAMETRO "b" Y NO
 De BUEUR NINGUN VALOR, SOLO REALIZA UNA OPERACIÓN que
 EN ESTE CASO ES IMPRIMIR UNA CADENA CONCATENADA CONEC
VALOR "B" YEN ESTE CASO ES UN LAMBOA CONSUMIDOR
Y coinside con la FIRMA De B
2 - este No tiene Ningun PARAMETRO PERO NO DEBUEVE Ningun
VALOR SIMPLEMENTE IMPRIME UNA CADENA, esta coinside A
LA FIRMA A gre NO TIENE NI PARAMETROS NI RETORNO, TAMBIEN
 ES UN CONSUMIDOR
3 en este caso, el LAMBIDA TOMA UN DARAMETRO POR
LO que implementa, la inteRFACE funcional 3° y es
UN CONSIMIDER de DATOS.
Y EN 05 TE CASO YOMAUN PARAMETRO Y GEBUELVE IN BULEARD
OSTE UTILIZA LA INTERFACE FUNCIONAL CÉ POR LO TANTO
es UN PREDICADO
S. este TOMA UN PARAMETRO Y dEVNEVUE UN PARAMETRO
este celuside con la FIRMA de interface Funcional
D' QUE TOMA UN PARAMETRO DY RETORNA UN R POR LO
TANTO ET UN PROVEEDOR de dATOS
```

DA- EN EL MAIN ESTARIA APREN diendo LAMBDAS a PRENdiento LAMBDAS NEW APREN duendo LAMBOLAS (); ADRENDIENDOLAMBDAS. UN METODO ((STRINGC) -> C. LENGTH() APRENDIENDOLAMBDAS.UN METODO ((STRINGC)-7 C. STARTS (VITH ("á));