

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:

- Utilizando un lambda dado que imprima true si el largo del String es par, false en caso contrario.
- 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 prácticamente 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();
```

```
1 → a.unMetodo(b) → { System.out.println("abcd" + b);};
```

```
2 → a.unMetodo() → System.out.println("abcd");
```

```
3 → a.unMetodo(variable) → { System.out.println("abcd");};
```

```
4 → a.unMetodo(String variable) → { 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 devuelve ningun valor, solo realiza una operacion que en este caso es imprimir una cadena concatenada con el valor "b" y en este caso es un lambda consumidor y coincide con la firma de "B"

2- este no tiene ningun parametro pero no devuelve ningun valor simplemente imprime una cadena, esta coincide a la firma "A" que no tiene ni parametros ni retorno, tambien es un consumidor

3 en este caso, el lambda toma un parametro por lo que implementa la interfaz funcional "B" y es un consumidor de datos.

4 en este caso toma un parametro y devuelve un booleano este utiliza la interfaz funcional "C" por lo tanto es un predicado

5- este toma un parametro y devuelve un parametro este coincide con la firma de interfaz funcional "D" que toma un parametro D y retorna un R por lo tanto es un proveedor de datos

2) A - en el MAIN ESTARIA

APRENDIENDO LAMBDA aprendiendo LAMBDA =
new Aprendiendo LAMBDA();

APRENDIENDO LAMBDA.unMetodo((String c) -> c.length()
% 2 == 0);

APRENDIENDO LAMBDA.unMetodo((String c) ->
c.startsWith("a"));