**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Iterator**;**

**import** java**.**util**.**List**;**

public class ColeccionApp **{**

private List**<**String**>** lista**;**

public void llenarLista**()** **{**

lista **=** **new** ArrayList**<>();**

lista**.**add**(**"Munoz"**);**

lista**.**add**(**"Robledo"**);**

lista**.**add**(**"Daniel"**);**

**}**

public void usarForEach**()** **{**

/\*for(String elemento : lista){

System.out.println(elemento);

}\*/

// lista.forEach(x -> System.out.println(x));

lista**.**forEach**(**System**.**out**::**println**);**

**}**

public void usarRemoveIf**()** **{**

/\*for(String elemento : lista){ ---- java.util.ConcurrentModificationException

if(elemento.equalsIgnoreCase("daniel")){

lista.remove(elemento);

}

}\*/

Iterator**<**String**>** it **=** lista**.**iterator**();**

**while(**it**.**hasNext**()){**

**if(**it**.**next**().**equalsIgnoreCase**(**"daniel"**)){**

it**.**remove**();**

**}**

**}**

// lista.removeIf(x -> x.equalsIgnoreCase("Code"));

**}**

public void usarSort**()** **{**

lista**.**sort**((**x**,**y**)->** x**.**compareTo**(**y**));**

**}**

public static void main**(**String**...** args**)** **{**

ColeccionApp app **=** **new** ColeccionApp**();**

app**.**llenarLista**();**

app**.**usarRemoveIf**();**

app**.**usarSort**();**

app**.**usarForEach**();**

**}**

**}**

**...**

Munoz

Robledo

**==============================**

@FunctionalInterface

public interface iApp **{**

double metodo**(**double a**,** double b**);** // es metodo de interfaz funcional!!!

// double metodo2(double a, double b); NO ES YA VALIDO PORQUE NO VALDRIAN LAS LAMBDA!!!

**default** public void metodoDesdeDefaul**(){**

System**.**out**.**println**(**"Soy un metodo default implementado!"**);**

**}**

static void motodoDesdeStatic**()** **{**

System**.**out**.**println**(**"Soy un metodo estatic!"**);**

**}**

**}**

**...**

public class Clase **{**

public static void main**(**String**[]** args**)** **{**

final double**[]** doble **=** **{**3**};**

iApp i\_1 **=** **new** iApp**()** **{**

@Override

public double metodo**(**double a**,** double b**)** **{**

**return** doble**[**0**]** **=** a **+** b**;**

**}**

**};**

iApp i\_2 **=** **(**a**,** b**)** **->** a **+** b **+** doble**[**0**];**

iApp i\_3 **=** **(**a**,** b**)** **->** **{return** **(**a **+** b **+** doble**[**0**]);};**

System**.**out**.**println**(**i\_1**.**metodo**(**3**,** 3**));**

System**.**out**.**println**(**i\_2**.**metodo**(**3**,** 3**));**

System**.**out**.**println**(**i\_3**.**metodo**(**3**,** 3**));**

iApp**.**motodoDesdeStatic**();**

**}**

**}**

**...**

6.0

12.0

12.0

Soy un metodo estatic**!**

**...**

public class ClaseParaDefault **implements** iApp**{**

public static void main**(**String**[]** args**)** **{**

ClaseParaDefault i **=** **new** ClaseParaDefault**();**

i**.**metodoDesdeDefaul**();**

**}**

@Override

public double metodo**(**double a**,** double b**)** **{**

**return** 0**;**

**}**

**}**

**...**

Soy un metodo **default** implementado**!**

**=========================================================**

@FunctionalInterface

public interface IPersona **{**

Persona crear**(**int id**,** String nombre**);**

**}**

**...**

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**Comparator**;**

public class MeRefApp **{** // HAY 4 MANERAS DE LLAMAR POR REFERENCIA!!!

public static void referenciarMetodoStatic**()** **{**

System**.**out**.**println**(**"Método Referido static"**);**

**}**

public void referenciarMetodoInstanciaObjetoArbitrario**()** **{**

String**[]** nombres **=** **{** "Mito"**,** "Code"**,** "Jaime" **};**

/\*

\* Arrays.sort(nombres, new Comparator<String>() {

\*

\* @Override public int compare(String o1, String o2) { return

\* o1.compareToIgnoreCase(o2); } });

\*/

// Arrays.sort(nombres, (s1,s2) -> s1.compareToIgnoreCase(s2));

// System.out.println(Arrays.toString(nombres));

Arrays**.**sort**(**nombres**,** String**::**compareToIgnoreCase**);**

System**.**out**.**println**(**Arrays**.**toString**(**nombres**));**

**}**

public void referenciarMetodoInstanciaObjetoParticular**()** **{**

System**.**out**.**println**(**"Método Referido Instancia de Objeto en particular"**);**

**}**

public void referenciarConstructor**()** **{**

/\*

\* IPersona iper = new IPersona(){

\*

\* @Override public Persona crear(int id, String nombre) { return new

\* Persona(id, nombre); }

\*

\* }; iper.crear(1, "MitoCode");

\*/

/\*

\* IPersona iper2 = (x,y)->(new Persona(x,y));

\* Persona per = iper2.crear(1, "MitoCode");

\* System.out.println(per.getId()+" - "+per.getNombre());

\*/

IPersona iper3 **=** Persona**::new;**

Persona per **=** iper3**.**crear**(**1**,** "MitoCode"**);**

System**.**out**.**println**(**per**.**getId**()** **+** " - " **+** per**.**getNombre**());**

**}**

public void operar**()** **{**

// Operacion op = () -> MeRefApp.referenciarMetodoStatic();

// op.saludar();

Operacion op2 **=** MeRefApp**::**referenciarMetodoStatic**;**

op2**.**saludar**();**

**}**

public static void main**(**String**[]** args**)** **{**

MeRefApp app **=** **new** MeRefApp**();**

// app.operar();

// app.referenciarMetodoInstanciaObjetoArbitrario();

Operacion op **=** app**::**referenciarMetodoInstanciaObjetoParticular**;** // porque es FUNCTIONAL, tendra el tregger desde este metodo!!!

op**.**saludar**();**

app**.**referenciarConstructor**();**

**}**

**}**

**...**

@FunctionalInterface

public interface Operacion **{**

void saludar**();**

**}**

**...**

public class Persona **{**

private int id**;**

private String nombre**;**

public Persona**()** **{** **}**

public Persona**(**int id**,** String nombre**){**

**this.**id **=** id**;**

**this.**nombre **=** nombre**;**

**}**

**...**

Método Referido Instancia de Objeto en particular

1 **-** MitoCode

**=======================================================**

PROGRAMACION IMPERATIVA

PROGRAMACION DECLARATIVA **---** STREAM API

public class StreamsApp **{**

private List**<**String**>** lista**;**

private List**<**String**>** numeros**;**

public StreamsApp**(){**

lista **=** **new** ArrayList**<>();**

lista**.**add**(**"Munoz"**);**

lista**.**add**(**"Robledo"**);**

lista**.**add**(**"Daniel"**);**

numeros **=** **new** ArrayList**<>();**

numeros**.**add**(**"1"**);**

numeros**.**add**(**"2"**);**

**}**

public void filtrar**(){**

lista**.**stream**().**filter**(**x **->** **!**x**.**startsWith**(**"R"**)).**forEach**(**System**.**out**::**println**);**

**}**

public void ordenar**(){**

// lista.stream().sorted().forEach(System.out::println); // ORDENA ASC

lista**.**stream**().**sorted**((**x**,**y**)->** y**.**compareTo**(**x**)).**forEach**(**System**.**out**::**println**);** // DESC

**}**

public void transformar**(){**

//lista.stream().map(String::toUpperCase).forEach(System.out::println); ME IMPRIME TODOS EN MAYUSCULAS

/\*for(String x : numeros){ --- FORMA IMPERATIVA

int num = Integer.parseInt(x) + 1;

System.out.println(num);

}\*/

numeros**.**stream**().**map**(**x **->** Integer**.**parseInt**(**x**)** **+** 1**).**forEach**(**System**.**out**::**println**);** // FORMA DECLARATIVA

**}**

public void limitar**(){**

lista**.**stream**().**limit**(**2**).**forEach**(**System**.**out**::**println**);**

**}**

public void contar**(){**

long x **=** lista**.**stream**().**count**();**

System**.**out**.**printf**(**"%s"**,** **(**double**)** x**);**

**}**

public static void main**(**String**[]** args**)** **{**

StreamsApp app **=** **new** StreamsApp**();**

app**.**filtrar**();**

app**.**ordenar**();**

app**.**transformar**();**

app**.**limitar**();**

app**.**contar**();**

**}**

**}**

**...**

Munoz

Daniel

Robledo

Munoz

Daniel

2

3

Munoz

Robledo

3.0

**======================================**

public class OptionalApp **{**

public void probar**(**String valor**){**

System**.**out**.**println**(**valor**.**contains**(**"Daniel"**));**

**try{**

Optional op **=** Optional**.**empty**();**

op**.**get**();**

**}catch(**Exception e**){**

System**.**out**.**println**(**"No hay elemento"**);**

**}**

**}**

public void orElse**(**String valor**){**

Optional**<**String**>** op **=** Optional**.**ofNullable**(**valor**);**

String x **=** op**.**orElse**(**"predeterminado"**);**

System**.**out**.**println**(**x**);**

**}**

public void orElseThrow**(**String valor**){**

Optional**<**String**>** op **=** Optional**.**ofNullable**(**valor**);**

op**.**orElseThrow**(**NumberFormatException**::new);**

**}**

public void isPresent**(**String valor**){**

Optional**<**String**>** op **=** Optional**.**ofNullable**(**valor**);**

System**.**out**.**println**(**op**.**isPresent**());**

**}**

public static void main**(**String**[]** args**)** **{**

OptionalApp app **=** **new** OptionalApp**();**

app**.**probar**(**"Daniel"**);**

app**.**orElse**(null);**

**try** **{**

app**.**orElseThrow**(null);**

**}** **catch** **(**Exception e**)** **{**

e**.**printStackTrace**();**

System**.**out**.**println**(**e**.**fillInStackTrace**());**

**}**

app**.**isPresent**(**"Daniel"**);**

**}**

**}**

**...**

**true**

No hay elemento

predeterminado

java**.**lang**.**NumberFormatException

at java**.**base**/**java**.**util**.**Optional**.**orElseThrow**(**Optional**.**java**:**408**)**

at optional**.**OptionalApp**.**orElseThrow**(**OptionalApp**.**java**:**26**)**

at optional**.**OptionalApp**.**main**(**OptionalApp**.**java**:**40**)**

java**.**lang**.**NumberFormatException

**true**

**====================================================** MAP

public class App **{**

private Map**<**Integer**,** String**>** map**;**

public void poblar**()** **{**

map **=** **new** HashMap**<>();**

map**.**put**(**1**,** "Daniel"**);**

map**.**put**(**2**,** "Robledo"**);**

map**.**put**(**3**,** "Munoz"**);**

**}**

public void imprimir\_v7**()** **{**

**for** **(**Entry**<**Integer**,** String**>** e **:** map**.**entrySet**())** **{**

System**.**out**.**println**(**"Llave: " **+** e**.**getKey**()** **+** " Valor: " **+** e**.**getValue**());**

**}**

**}**

**...**

Llave**:** 1 Valor**:** Daniel

Llave**:** 2 Valor**:** Robledo

Llave**:** 3 Valor**:** Munoz

public void insertarSiAusente**()** **{**

map**.**putIfAbsent**(**5**,** "Jaime"**);**

**}**

public void imprimir\_v8**()** **{**

//map.forEach((k,v) -> System.out.println("Llave: " + k + " Valor: " + v));

map**.**entrySet**().**stream**().**forEach**(**System**.**out**::**println**);**

**}**

**...**

1**=**Daniel

2**=**Robledo

3**=**Munoz

5**=**Jaime

public void operarSiPresente**()** **{**

map**.**computeIfPresent**(**3**,** **(**k**,**v**)->** k **+** v**);**

System**.**out**.**println**(**map**.**get**(**3**));**

**}**

**...**

3Munoz

public void obtenerOrPredeterminado**()** **{**

String valor **=** map**.**getOrDefault**(**5**,** "valor por defecto"**);**

System**.**out**.**println**(**valor**);**

**}**

**...**

valor por defecto

public void recolectar**()** **{**

Map**<**Integer**,** String**>** mapaRecolectado **=** map**.**entrySet**().**stream**()**

**.**filter**(**e**->**e**.**getValue**().**contains**(**"d"**))**

**.**collect**(**Collectors**.**toMap**(**p**->** p**.**getKey**(),** p**->** p**.**getValue**()));**

mapaRecolectado**.**forEach**((**k**,**v**)->** System**.**out**.**println**(**"Llave: " **+** k **+** " Valor: " **+** v**));**

**}**

**...**

Llave**:** 2 Valor**:** Robledo

public static void main**(**String**[]** args**)** **{**

App app **=** **new** App**();**

app**.**poblar**();**

app**.**imprimir\_v7**();**

app**.**insertarSiAusente**();**

app**.**imprimir\_v8**();**

app**.**operarSiPresente**();**

app**.**obtenerOrPredeterminado**();**

app**.**recolectar**();**

**}**

**}**

**==============================================** DATETIME

**import** java**.**text**.**DateFormat**;**

**import** java**.**text**.**ParseException**;**

**import** java**.**text**.**SimpleDateFormat**;**

**import** java**.**time**.**Duration**;**

**import** java**.**time**.**Instant**;**

**import** java**.**time**.**LocalDate**;**

**import** java**.**time**.**LocalDateTime**;**

**import** java**.**time**.**LocalTime**;**

**import** java**.**time**.**Period**;**

**import** java**.**time**.**format**.**DateTimeFormatter**;**

**import** java**.**util**.**Calendar**;**

**import** java**.**util**.**Date**;**

public class App **{**

public void verificar**(**int version**)** **{**

**if** **(**version **==** 7**)** **{**

Calendar fecha1 **=** Calendar**.**getInstance**();**

Calendar fecha2 **=** Calendar**.**getInstance**();**

fecha1**.**set**(**1991**,** 0**,** 21**);**

System**.**out**.**println**(**fecha1**.**after**(**fecha2**));**

**}** **else** **if** **(**version **==** 8**)** **{**

LocalDate fecha1 **=** LocalDate**.**of**(**1991**,** 01**,** 21**);**

LocalDate fecha2 **=** LocalDate**.**now**();**

System**.**out**.**println**(**fecha1**.**isAfter**(**fecha2**));**

System**.**out**.**println**(**fecha1**.**isBefore**(**fecha2**));**

LocalTime tiempo1 **=** LocalTime**.**of**(**22**,** 30**,** 50**);**

LocalTime tiempo2 **=** LocalTime**.**now**();**

System**.**out**.**println**(**tiempo1**.**isAfter**(**tiempo2**));**

System**.**out**.**println**(**tiempo1**.**isBefore**(**tiempo2**));**

LocalDateTime fechaTiempo1 **=** LocalDateTime**.**of**(**1991**,** 1**,** 21**,** 22**,** 30**,** 50**);**

LocalDateTime fechaTiempo2 **=** LocalDateTime**.**now**();**

System**.**out**.**println**(**fechaTiempo1**.**isAfter**(**fechaTiempo2**));**

System**.**out**.**println**(**fechaTiempo1**.**isBefore**(**fechaTiempo2**));**

**}**

**}**

**...**

**false**

**true**

**true**

**false**

**false**

**true**

// currentTimeMillis

public void medirTiempo**(**int version**)** **throws** InterruptedException **{**

**if** **(**version **==** 7**)** **{**

long ini **=** System**.**currentTimeMillis**();**

Thread**.**sleep**(**1000**);**

long fin **=** System**.**currentTimeMillis**();**

System**.**out**.**println**(**fin **-** ini**);**

**}** **else** **if** **(**version **==** 8**)** **{**

Instant ini **=** Instant**.**now**();**

Thread**.**sleep**(**1000**);**

Instant fin **=** Instant**.**now**();**

System**.**out**.**println**(**Duration**.**between**(**ini**,** fin**).**toMillis**());**

**}**

**}**

**...**

1000

// Calendar

public void periodoEntreFechas**(**int version**)** **{**

**if** **(**version **==** 7**)** **{**

Calendar nacimiento **=** Calendar**.**getInstance**();**

Calendar actual **=** Calendar**.**getInstance**();**

nacimiento**.**set**(**1991**,** 0**,** 21**);**

actual**.**set**(**2017**,** 2**,** 04**);**

int anios **=** 0**;**

**while** **(**nacimiento**.**before**(**actual**))** **{**

nacimiento**.**add**(**Calendar**.**YEAR**,** 1**);**

**if** **(**nacimiento**.**before**(**actual**))** **{**

anios**++;**

**}**

**}**

System**.**out**.**println**(**anios**);**

**}** **else** **if** **(**version **==** 8**)** **{**

LocalDate nacimiento **=** LocalDate**.**of**(**1991**,** 1**,** 21**);**

LocalDate actual **=** LocalDate**.**now**();**

Period periodo **=** Period**.**between**(**nacimiento**,** actual**);**

System**.**out**.**println**(**"Han transcurrido " **+** periodo**.**getYears**()** **+** " años y " **+** periodo**.**getMonths**()** **+** " meses y "

**+** periodo**.**getDays**()** **+** " dias, desde " **+** nacimiento **+** " hasta " **+** actual**);**

**}**

**}**

**...**

Han transcurrido 28 años y 1 meses y 24 dias**,** desde 1991**-**01**-**21 hasta 2019**-**03**-**17

// Conversion

public void convertir**(**int version**)** **throws** ParseException **{**

**if** **(**version **==** 7**)** **{**

String fecha **=** "21/01/1991"**;**

DateFormat formateador **=** **new** SimpleDateFormat**(**"dd/MM/yyyy"**);**

Date fechaConvertida **=** formateador**.**parse**(**fecha**);**

System**.**out**.**println**(**fechaConvertida**);**

Date fechaActual **=** Calendar**.**getInstance**().**getTime**();**

formateador **=** **new** SimpleDateFormat**(**"dd/MM/yyyy HH:mm:ss a"**);**

String fechaCadena **=** formateador**.**format**(**fechaActual**);**

System**.**out**.**println**(**fechaCadena**);**

**}** **else** **if** **(**version **==** 8**)** **{**

String fecha **=** "21/01/1991"**;**

DateTimeFormatter formateador **=** DateTimeFormatter**.**ofPattern**(**"dd/MM/yyyy"**);** // ES NECESARIO!!! Si no exception!!!

LocalDate fechaLocal **=** LocalDate**.**parse**(**fecha**,** formateador**);**

System**.**out**.**println**(**fechaLocal**);**

formateador **=** DateTimeFormatter**.**ofPattern**(**"ddMMyyyy"**);**

System**.**out**.**println**(**formateador**.**format**(**fechaLocal**));**

**}**

**}**

**...** con la 7

Mon Jan 21 00**:**00**:**00 GMT 1991

17**/**03**/**2019 15**:**47**:**58 pm

**...**

1991**-**01**-**21

21011991

public static void main**(**String**[]** args**)** **{**

App app **=** **new** App**();**

**try** **{**

app**.**verificar**(**8**);**

app**.**medirTiempo**(**8**);**

app**.**periodoEntreFechas**(**8**);**

app**.**convertir**(**8**);**

**}** **catch** **(**Exception e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

**=====================================** FUNCIONES DE ALTO ORDEN

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**List**;**

**import** java**.**util**.**function**.**Consumer**;**

**import** java**.**util**.**function**.**Function**;**

**import** java**.**util**.**function**.**Predicate**;**

public class HighApp **{**

private Function**<**String**,** String**>** convertirMayusculas **=** x **->** x**.**toUpperCase**();**

private Function**<**String**,** String**>** convertirMinusculas **=** x **->** x**.**toLowerCase**();**

public void imprimir**(**Function**<**String**,** String**>** funcion**,** String valor**)** **{**

System**.**out**.**println**(**funcion**.**apply**(**valor**));**

**}**

public Function**<**String**,** String**>** mostrar**(**String mensaje**)** **{**

**return** **(**String x**)** **->** mensaje **+** x**;**

**}**

public void filtrar**(**List**<**String**>** lista**,** Consumer**<**String**>** consumidor**,** int longitud**,** String cadena**){**

//lista.stream().filter(this.establecerLogicaFiltro(longitud)).forEach(consumidor);

lista**.**stream**().**filter**(this.**establecerLogicaFiltro**(**cadena**)).**forEach**(**consumidor**);**

**}**

public Predicate**<**String**>** establecerLogicaFiltro**(**int longitud**){**

**return** texto **->** texto**.**length**()** **<** longitud**;**

**}**

public Predicate**<**String**>** establecerLogicaFiltro**(**String cadena**){**

**return** texto **->** texto**.**contains**(**cadena**);**

**}**

public static void main**(**String**[]** args**)** **{**

HighApp app **=** **new** HighApp**();**

app**.**imprimir**(**app**.**convertirMayusculas**,** "daniel"**);** // DANIEL

app**.**imprimir**(**app**.**convertirMinusculas**,** "Daniel"**);** // daniel

String rpta **=** app**.**mostrar**(**"Hola "**).**apply**(**"Daniel-Robledo"**);** // Hola Daniel-Robledo

System**.**out**.**println**(**rpta**);**

List**<**String**>** lista **=** **new** ArrayList**<>();**

lista**.**add**(**"Munoz"**);**

lista**.**add**(**"Robledo"**);**

lista**.**add**(**"Daniel"**);**

//app.filtrar(lista, System.out::println, 5, null);

app**.**filtrar**(**lista**,** System**.**out**::**println**,** 0**,** "o"**);**

**}**

**}**

Munoz

Robledo

==================================== GENERICS

**Type Parameter Naming Conventions**

By convention, type parameter names are single, uppercase letters. This stands in sharp contrast to the variable [naming](https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html#naming) conventions that you already know about, and with good reason: Without this convention, it would be difficult to tell the difference between a type variable and an ordinary class or interface name.

The most commonly used type parameter names are:

* E - Element (used extensively by the Java Collections Framework)
* K - Key
* N - Number
* T - Type
* V - Value
* S,U,V etc. - 2nd, 3rd, 4th types

You'll see these names used throughout the Java SE API and the rest of this lesson.

public class Clase**<**K**,**T**,**V**,**E**>** **{**

private K objetoK**;**

private T objetoT**;**

private V objetoV**;**

private E objetoE**;**

public Clase**(**K objetoK**,** T objetoT**,** V objetoV**,** E objetoE**)** **{**

**this.**objetoK **=** objetoK**;**

**this.**objetoT **=** objetoT**;**

**this.**objetoV **=** objetoV**;**

**this.**objetoE **=** objetoE**;**

**}**

public void mostrarTipo**(){**

System**.**out**.**println**(**"K es un: " **+** objetoK**.**getClass**().**getName**());**

System**.**out**.**println**(**"T es un: " **+** objetoT**.**getClass**().**getName**());**

System**.**out**.**println**(**"V es un: " **+** objetoV**.**getClass**().**getName**());**

System**.**out**.**println**(**"E es un: " **+** objetoE**.**getClass**().**getName**());**

**}**

**}**

**...**

public class App **{**

public static void main**(**String**[]** args**)** **{**

/\*List<String> lista = new ArrayList<>();

lista.add("MitoCode");

//lista.add(25);

String texto = lista.get(0);

System.out.println(texto);

/\*String[] array = new String[5];

array[0] = "MitoCode";

array[1] = 1;\*/

Clase**<**String**,**Integer**,**String**,**Double**>** c **=** **new** Clase**<>(**"MitoCode"**,**25**,**"Jaime"**,**25.0**);**

c**.**mostrarTipo**();**

List**<**Clase**<**String**,**Integer**,**String**,**String**>>** lista **=** **new** ArrayList**<>();**

lista**.**add**(new** Clase**<**String**,**Integer**,**String**,**String**>(**"MitoCode"**,**25**,**"Jaime"**,**"Suscribete"**));**

**for(**Clase**<**String**,**Integer**,**String**,**String**>** cl **:** lista**){**

cl**.**mostrarTipo**();**

**}**

**}**

**}**

**...**

K es un**:** java**.**lang**.**String

T es un**:** java**.**lang**.**Integer

V es un**:** java**.**lang**.**String

E es un**:** java**.**lang**.**Double

K es un**:** java**.**lang**.**String

T es un**:** java**.**lang**.**Integer

V es un**:** java**.**lang**.**String

E es un**:** java**.**lang**.**String

**============================================** WILDCARDS

ublic class AppWild **{**

public void listarUpperBounded**(**List**<?** **extends** Persona**>** lista**)** **{**

**for** **(**Persona per **:** lista**)** **{**

//if(a instanceof Alumno){

System**.**out**.**println**(**per**.**getNombre**());**

//}else if(a instanceof Profesor){

//}

**}**

**}**

public void listarLowerBounded**(**List**<?** **super** Alumno**>** lista**)** **{**

**for** **(**Object al **:** lista**)** **{**

//if(a instanceof Alumno){

System**.**out**.**println**(((**Persona**)**al**).**getNombre**());**

//}else if(a instanceof Profesor){

//}

**}**

**}**

public void listarUnBounded**(**List**<?>** lista**)** **{**

**for** **(**Object al **:** lista**)** **{**

//if(a instanceof Alumno){

System**.**out**.**println**(((**Persona**)**al**).**getNombre**());**

//}else if(a instanceof Profesor){

//}

**}**

**}**

public static void main**(**String**[]** args**)** **{**

AppWild aw **=** **new** AppWild**();**

Persona al1 **=** **new** Profesor**(**"MITOCODE"**);**

Persona al2 **=** **new** Profesor**(**"JAIME"**);**

Persona al3 **=** **new** Profesor**(**"MITO"**);**

List**<**Persona**>** lista **=** **new** ArrayList**<>();**

lista**.**add**(**al1**);**

lista**.**add**(**al2**);**

lista**.**add**(**al3**);**

aw**.**listarUpperBounded**(**lista**);**

System**.**out**.**println**();**

aw**.**listarLowerBounded**(**lista**);**

System**.**out**.**println**();**

aw**.**listarUnBounded**(**lista**);**

**}**

**}**

**...**

MITOCODE

JAIME

MITO

MITOCODE

JAIME

MITO

MITOCODE

JAIME

MITO