UNIVERSIDAD DE LIMA

DISEÑO DE INTERFACES

UNIDAD 3: CLASES ABSTRACTAS E INTERFACES

SEMANA 10

UNIVERSIDAD DE LIMA

MOTIVACIÓN

https://youtu.be/03Cyw5dlKa4

Temario

- Definición, reglas y uso de interfaces.
- Diferencias entre clases abstractas y clases interfaces.
- Herencia múltiple.
- Desarrollo de ejercicios usando interfaces.

Interfaces

- Colección de métodos abstractos, los cuales solo tienen declaración y no implementación.
- Los métodos serán implementados por una o mas clases.
- Para crear una interface, se utiliza la palabra reservada «*interface*» en lugar de class. La interface puede definirse public o sin modificador de acceso
- Una interface puede también contener atributos, pero estos son siempre ***static*** y ***final***.
- Todos los métodos que declara una interface son siempre ***public***.

Reglas y Uso

- Proveer implementación concreta (no abstracta) para todos los métodos declarados en una interface.
- Seguir las reglas de la sobre escritura de métodos.
- Declarar excepciones no verificadas en la implementación de los métodos distintos de aquellos declarados por los métodos de la interface, o subclases de aquellos declarados en los métodos de la interface.
- Mantener la firma de los métodos de la interface, y mantener el mismo tipo de retorno (o un subtipo del tipo de retorno).La clase que usa una interfaz debe implementar todos los métodos.

Diferencia entre clase Abstracta e Interface

- Como Java no permite herencia múltiple – una clase sólo puede extender una superclase – esto dificulta que una clase se adecue a más de un comportamiento. Una interfaz, por el contrario, permite que una clase implemente una o más interfaces para resolver el problema de mezclar diversos comportamientos en un mismo tipo de objeto
- Una clase abstracta no puede implementar los métodos declarados como abstractos, una interface no puede implementar ningún método (ya que todos son abstractos).

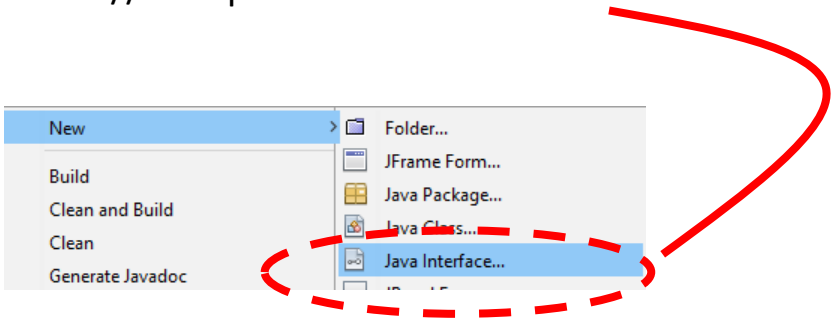
Herencia Múltiple

- Una clase puede implementar varias interfaces, pero sólo puede tener una clase ascendiente directa.
- Contaminación de una interfaz. Cuando se añade un método a una clase base solo porque una de sus derivadas lo necesita.

Declaración de una Interface

Sintaxis:

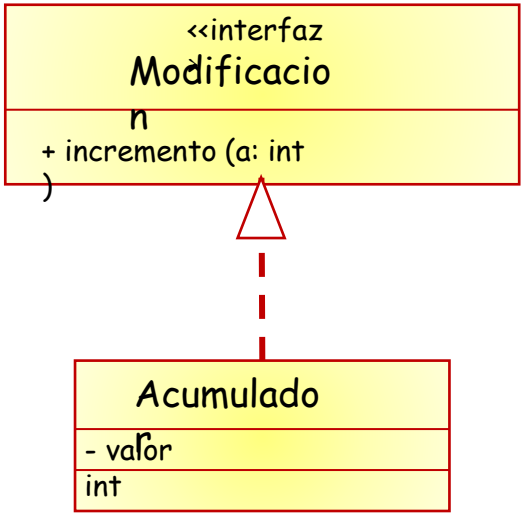
```
public interface IdentificadorInterfaz {
        // Cuerpo de la interface . . .
}
```



NOTA: no incluye ni la declaración de variables de instancia ni la implementación de los métodos (sólo las cabeceras).

UNIVERSIDAD DE LIMA

Implementación de una Interface en una clase



Implementación de una Interface en una clase

Para declarar una clase que implemente una interface es necesario utilizar la palabra reservada implements en la cabecera de declaración de la clase.

Ejemplo:
Definamos la interface Modificacion

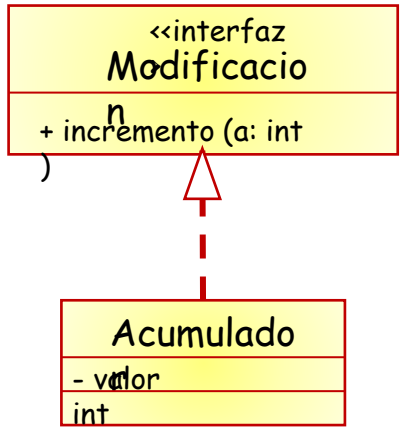

```
public interface Modificacion {
    void incremento (int a);
}
```

UNIVERSIDAD DE LIMA

Implementación de una Interface en una clase

Ejemplo:

Definamos ahora la clase Acumulador.




```
2
3   public class Acumulador  implements Modificacion{
4       public int valor;
5
6       public Acumulador(int valor) {
7           this.valor = valor;
8       }
9
10      public int daValor(){
11          return valor;
12      }
13
15      public void incremento(int a){
15          valor+=a;
16      }
17  }
```

Obliga a la clase Acumulador a definir el método incremento declarado en la interface Modificacion.

UNIVERSIDAD DE LIMA

Implementación de una Interface en una clase

Ejecución:

```
3   public class Prueba {
4
5       public static void main(String[] args) {
6
7           Acumulador p = new Acumulador(25);
8           p.incremento(12);
9           System.out.println(p.daValor());
10      }
11  }
```

```
run:
37
BUILD SUCCESSFUL (total time: 0 seconds)
```

La clase **Acumulador** tendría también la posibilidad de utilizar directamente las constantes declaradas en la interface si las hubiera.

UNIVERSIDAD DE LIMA

Jerarquía entre Interface

La jerarquía entre interfaces permite la herencia simple y múltiple. Es decir, tanto la declaración de una clase, como la de una interfaz pueden incluir la implementación de otras interfaces.

Ejemplo:

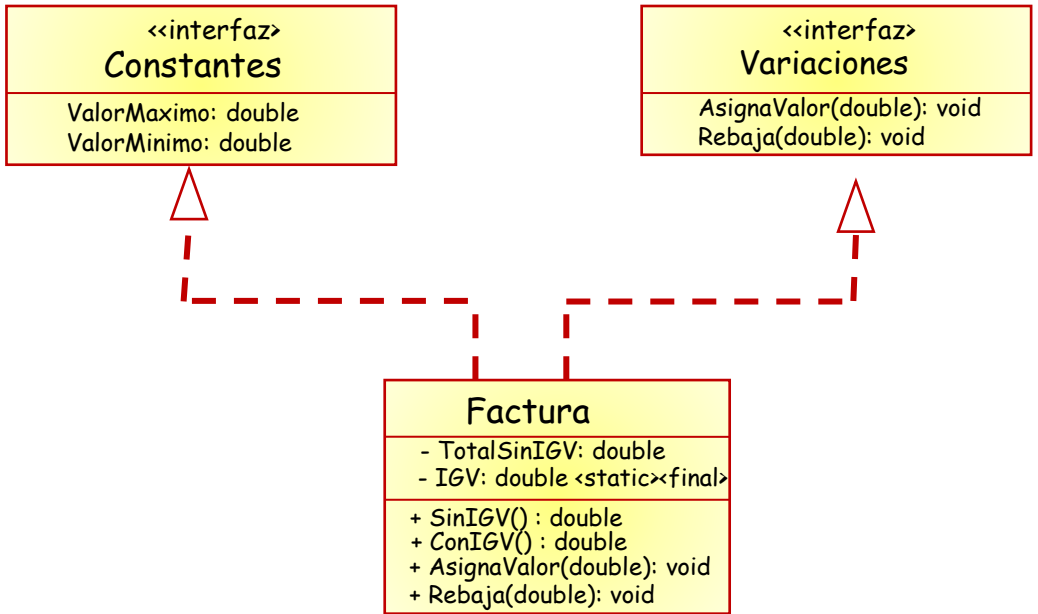
```
public class Una implements Dos, Tres {
        // Cuerpo de la interface . . .
}
```

Las clases que implementan la interface **Una** también lo hacen con **Dos** y **Tres**

Jerarquía entre Interface

Otro ejemplo de aplicación:

Construyamos dos interfaces, **Constantes** y **Variaciones**, y una clase, **Factura**, que las implementa.



UNIVERSIDAD DE LIMA

Jerarquía entre Interface

Otro ejemplo de aplicación:

Construyamos dos interfaces, **Constantes** y **Variaciones**, y una clase, **Factura**, que las implementa.

```
3   public interface Constantes {
4       double valorMaximo=1000000.0;
5       double valorMinimo=-0.01;
6   }
```

```
3   public interface Variaciones {
4       void asignaValor(double x);
5       void rebaja(double t);
6   }
```

UNIVERSIDAD DE LIMA

Jerarquía entre Interface

Otro ejemplo de aplicación:

Declaración de la clase **Factura**, que las implementa.

```
3   public class Factura implements Constantes, Variaciones{
4       private double totalSinIGV;
5       public final static double IGV = 0.19;
6
7       public double sinIGV() {
8           return totalSinIGV;
9       }
10      public double conIGV() {
11          return totalSinIGV * (1+IGV);
12      }
13      public void asignaValor(double x) {
14          if (valorMinimo<x){
15              totalSinIGV=x;
16          }else{
17              totalSinIGV=0;
18          }
19      }
20      public void rebaja(double t) {
21          totalSinIGV *= (1-t/100);
22      }
23  }
```

UNIVERSIDAD DE LIMA

Jerarquía entre Interface

Otro ejemplo de aplicación: Clase **Prueba**.

```
3   public class Prueba {
4
5       public static void main(String[] args) {
6           Factura refF = new Factura();
7           refF.asignaValor(250.0);
8           System.out.println("El precio sin IGV es: " + refF.sinIGV());
9           System.out.println("El precio con IGV es: " + refF.conIGV());
10          System.out.println("Rebajado durante el mes de mayo un 20%");
11          refF.rebaja(20);
12          System.out.println("Rebajado sin IGV es: " + refF.sinIGV());
13          System.out.println("Rebajado con IGV es: " + refF.conIGV());
14      }
15  }
```

```
Salida - Proyecto Interface_02 (run)
run:
El precio sin IGV es: 250.0
El precio con IGV es: 297.5
Rebajado durante el mes de mayo un 20%
Rebajado sin IGV es: 200.0
Rebajado con IGV es: 238.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

UNIVERSIDAD DE LIMA

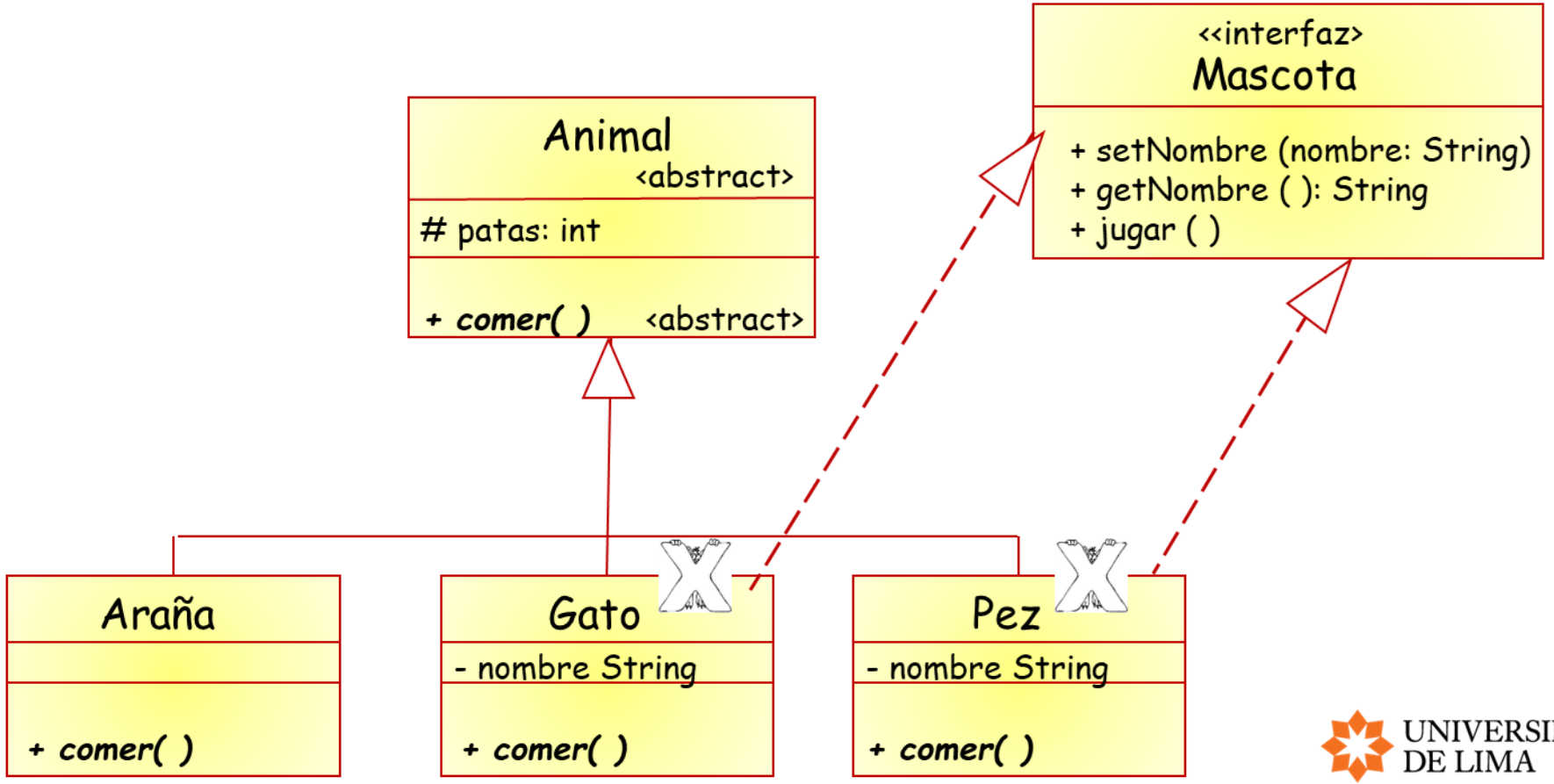
Jerarquía entre Interface

Una clase puede simultáneamente descender de otra clase e implementar una o varias interfaces. En este caso la sección **implements** se coloca a continuación de **extends** en la cabecera de declaración de la clase.

Ejemplo :

```
public class ClaseDescendiente extends ClaseAscendiente
implements Interfaz {
        ...
}
```

Trabajando con Interfaces y Clases Abstractas



UNIVERSIDAD DE LIMA

Trabajando con Interfaces y Clases Abstractas

Clase abstracta Animal

```
3   public abstract class Animal {
4       //atributo
5       protected int patas;
6
7       //constructor
8       public Animal(int patas) {
9           this.patas = patas;
10      }
11
12      //método abstracto a implementar por
13      //todas las clases hijas
14      public abstract void comer();
15
16      //método que puede ser redefinido
17      //por las clases hijas
18      public void caminar(){
19          System.out.println("caminar () de la clase padre " +
20                  " este animal camina sobre " +  patas +  " patas ");
21      }
22  }
```

UNIVERSIDAD DE LIMA

Trabajando con Interfaces y Clases Abstractas

Clase hija Araña

```
3   public class Araña extends Animal{
4
5       //constructor
6       public Araña() {
7           super(8);
8       }
9
10      //implementación del método
11      //abstracto
12      public void comer(){
13          System.out.println(" comer () de la clase hija Araña " +
14                  " las arañas atrapan moscas para comer ");
15      }
16  }
```

UNIVERSIDAD DE LIMA

Trabajando con Interfaces y Clases Abstractas

Interface Mascota

```
1  public interface Mascota {
2      public void setNombre(String nombre);
3      public String getNombre();
4      public void jugar();
7  }
```

Clase hija Gato

```
3  public class Gato extends Animal implements Mascota{
4      //atributo
5      private String nombre;
6
7      //constructor
8      public Gato() {
9          super(4);
10     }
11
12 //implementación del método abstracto
14     public void comer() {
15         System.out.println(" comer() de la clase hija Gato.  " +
16                 " Este  gato come vegetales.");
17     }
```

UNIVERSIDAD DE LIMA

Trabajando con Interfaces y Clases Abstractas

Clase hija Gato ……. continuación

```
18      //método redefinido
19      public void caminar() {
20          super.caminar();
21          System.out.println("caminar() de la clase hija Gato.");
22      }
23      //implementación de los métodos de la interface
24      public void setNombre(String nombre) {
25          this.nombre = nombre;
26      }
27      public String getNombre() {
28          return nombre;
29      }
30      public void jugar() {
31          System.out.println("Los  gatos juegan y arañan.");
32      }
33  }
```

UNIVERSIDAD DE LIMA

Trabajando con Interfaces y Clases Abstractas

Clase hija Pez

```
3   public class Pez extends Animal implements Mascota{
4      //atributo
5      private String nombre;
6
7      //constructor
8       public Pez() {
9            super(0);
10     }
11
12      //método redefinido
        public void caminar() {
14          super.caminar();
15          System.out.println("caminar() de la clase hija Pez. " +
16                                              " El pez no camina, nada!!.");
17      }
18
19      //implementación del método abstracto
        public void comer() {
21          System.out.println("comer() de la clase hija Pez." +
22                    " Este pez come plantas acuáticas.");
23     }
```

UNIVERSIDAD DE LIMA

Trabajando con Interfaces y Clases Abstractas

Clase hija Pez ……. Continuación

```
25      //métodos de la interface
        public void setNombre(String nombre) {
27              this.nombre = nombre;
28      }
        public String getNombre() {
30            return nombre;
31      }
32
        public void jugar() {
34        System.out.println("Los peces nadan todo el día.");
35      }
36  }
```

UNIVERSIDAD DE LIMA

Trabajando con Interfaces y Clases Abstractas

Clase Prueba

```
7   public class Prueba {
8       public static void main(String[] args) {
9
10          Araña a = new Araña();
11          Pez p = new Pez();
12          Gato g = new Gato();
13
14          // demostración método abstracto
15          a.comer();
16          p.comer();
17          g.comer();
18          // demostración método redefinido
19          p.caminar();
20          g.caminar();
21          a.caminar();
22          //Demostración de diferentes
23          //implementaciones de una interface
24           p.jugar();
25           g.jugar();
26           g.setNombre("Benito");
27           p.setNombre("carlota");
28           System.out.println(g.getNombre());
29           System.out.println(p.getNombre());
30      }
31  }
```

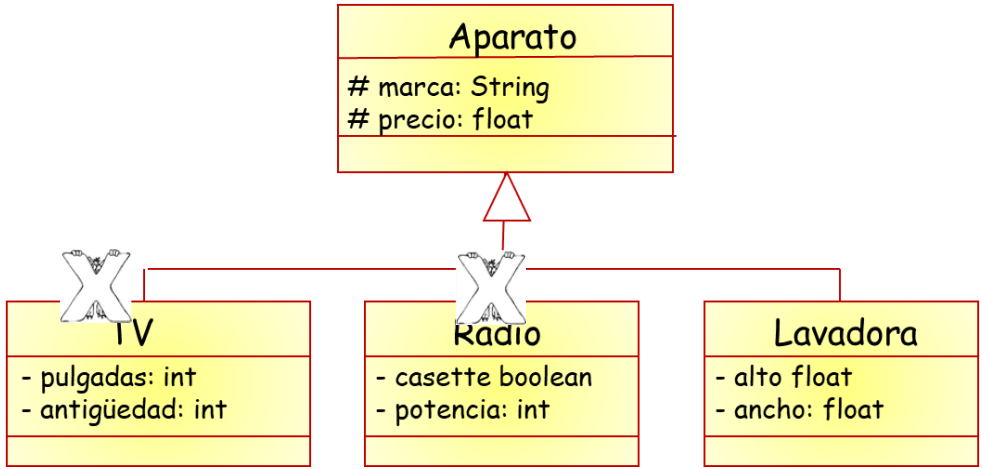
Ejecución

```
run:
comer () de la clase hija Araña  las arañas atrapan moscas para comer
comer() de la clase hija Pez. Este pez come plantas acuáticas.
comer() de la clase hija Gato.   Este  gato come vegetales.
caminar () de la clase padre  este animal camina sobre 0 patas
caminar() de la clase hija Pez.  El pez no camina, nada!!.
caminar () de la clase padre  este animal camina sobre 4 patas
caminar() de la clase hija Gato.
caminar () de la clase padre  este animal camina sobre 8 patas
Los peces nadan todo el día.
Los  gatos juegan y arañan.
Benito
carlota
BUILD SUCCESSFUL (total time: 0 seconds)
```

UNIVERSIDAD DE LIMA

Ejercicio

Considere la siguiente jerarquía de clases:



Se pide:

1. Construir un interface llamado "Hablador" que posea un único método "hablar()" (sin parámetros y sin valor de retorno).

2. Hacer que todas las clases que represente a entidades con la capacidad de hablar implementen esta interface (clases que aparecen marcadas con x).

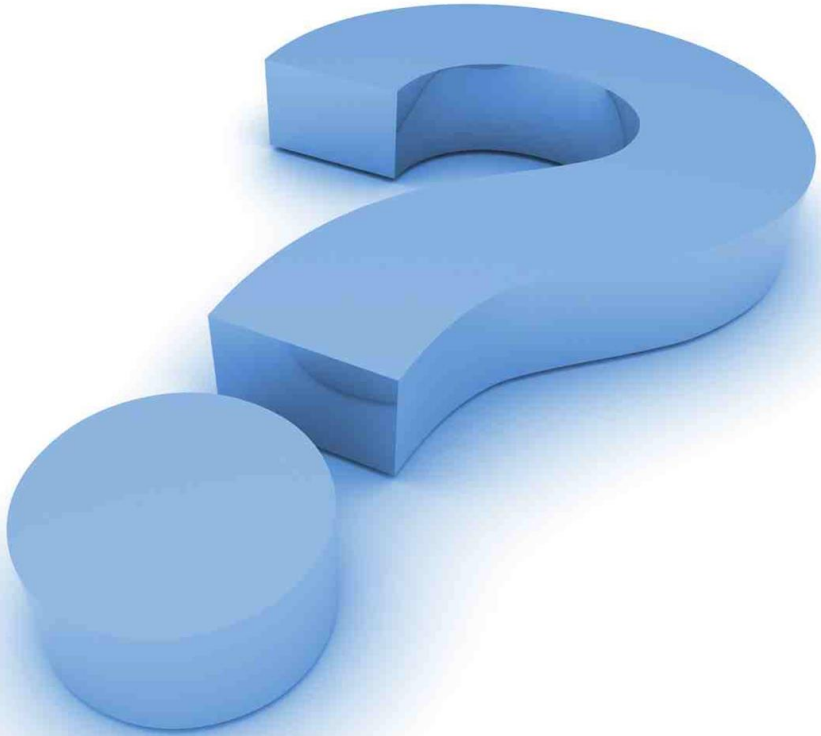
UNIVERSIDAD DE LIMA

Ejercicio

Cada una de estas clases debe implementar este interfaz de manera que el método “hablar()” visualice por pantalla el mensaje “Hola, soy un <CLASE> y sé hablar”, junto con los valores de los atributos del objeto

Por ejemplo:

Hola, soy una TV y sé hablar.
Marca: “Sony”
Precio: 1560.00 soles
Pulgadas: 20 pulg
Antigüedad : 3 años



UNIVERSIDAD DE LIMA

Referencias

- Deitel, H. M. (2016). Java: como programar.
- Paul S. Wang, Java con programación orientada a objetos y aplicaciones en la WWW, México, 2000.
- Sun microsystem, Fundamentals of the Java™ Programming Language SL-110-SE6
- A.M. Vozmediano, Java para novatos, 2017.