UNIVERSIDAD DE LIMA

HERENCIA – ATRIBUTOS - MÉTODOS

UNIDAD 3: HERENCIA Y POLIMORFISMO

SEMANA 8

Temario

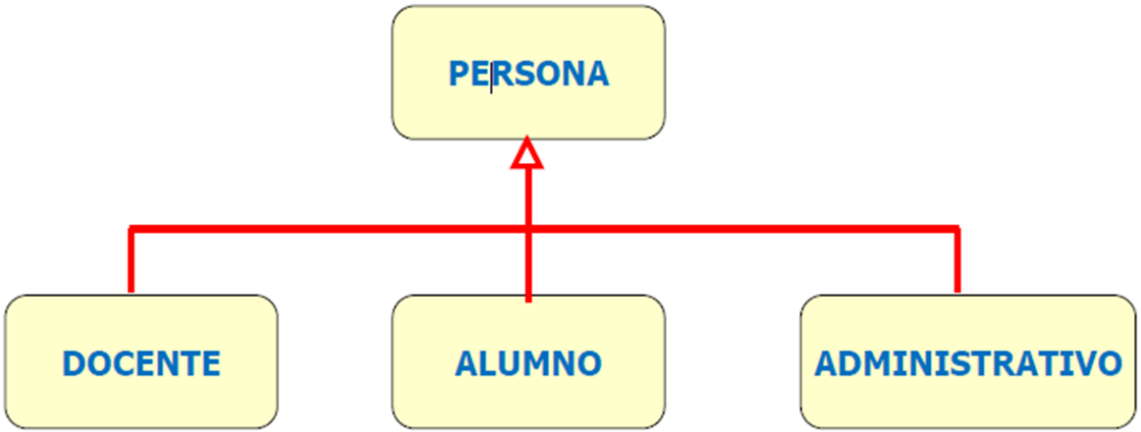
- Definición de Herencia Simple. Representación de la herencia con UML.
- Superclases y Subclases.
- Constructores de subclases.
- Miembros protegidos.
- Métodos redefinidos.

¿Es útil la herencia en programación?

https://youtu.be/KJ3wLaLQtF4

1. Definición de Herencia Simple

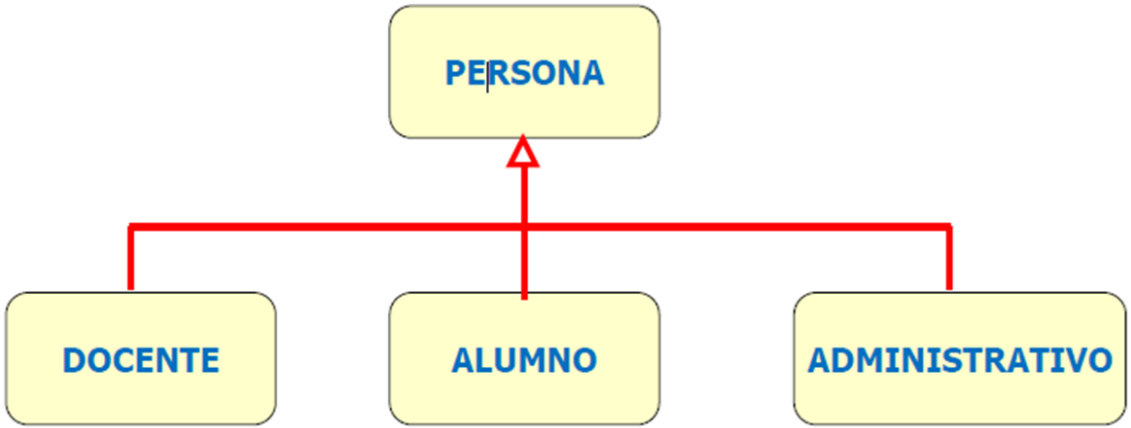
- Permite crear subclases a partir de otras clases, con el propósito de: Extender, modificar y/o implementar funcionalidad.
  - Por ejemplo:En el contexto académico, Se puede identificar las clases: CPersona, CDocente, CAlumno y CAdministrativo. Notar que la clase CDocente debe tener algunos de los atributos y métodos de la clase CPersona, entonces, su implementación es una extensión a la clase CPersona, agregando lo necesario para que se comporte como un Docente.



UNIVERSIDAD DE LIMA

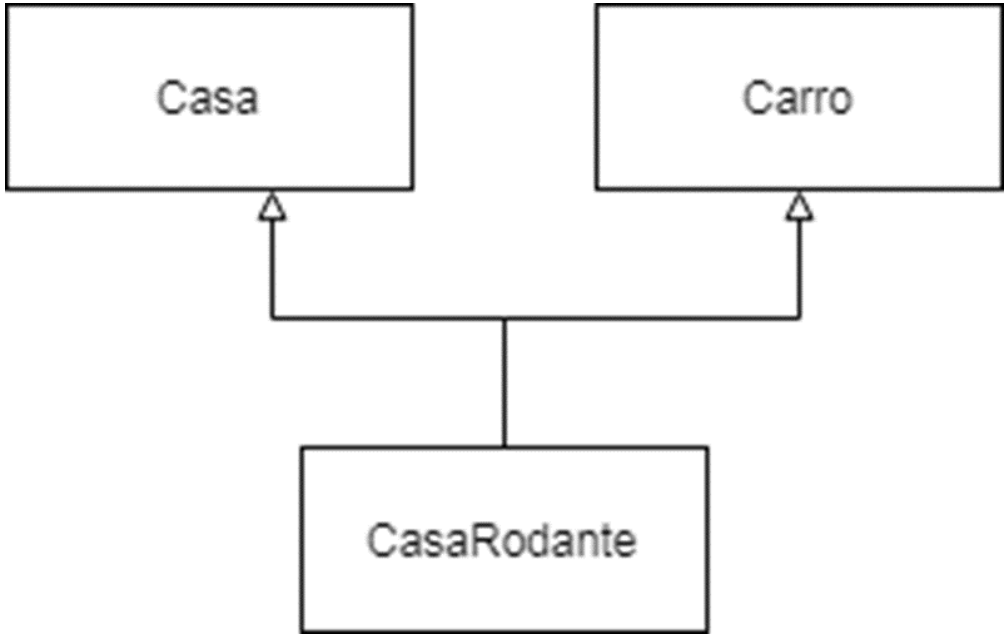
1. Definición de Herencia Simple (Continua...)

- La herencia es el principio básico para reutilizar código entre clases relacionadas.
- La herencia modela la relación ES–UN(A)
  - En el diagrama: Docente ES UNA Persona, Alumno ES UNA Persona y Administrativo ES UNA Persona



1. Definición de Herencia Simple (Continua...)

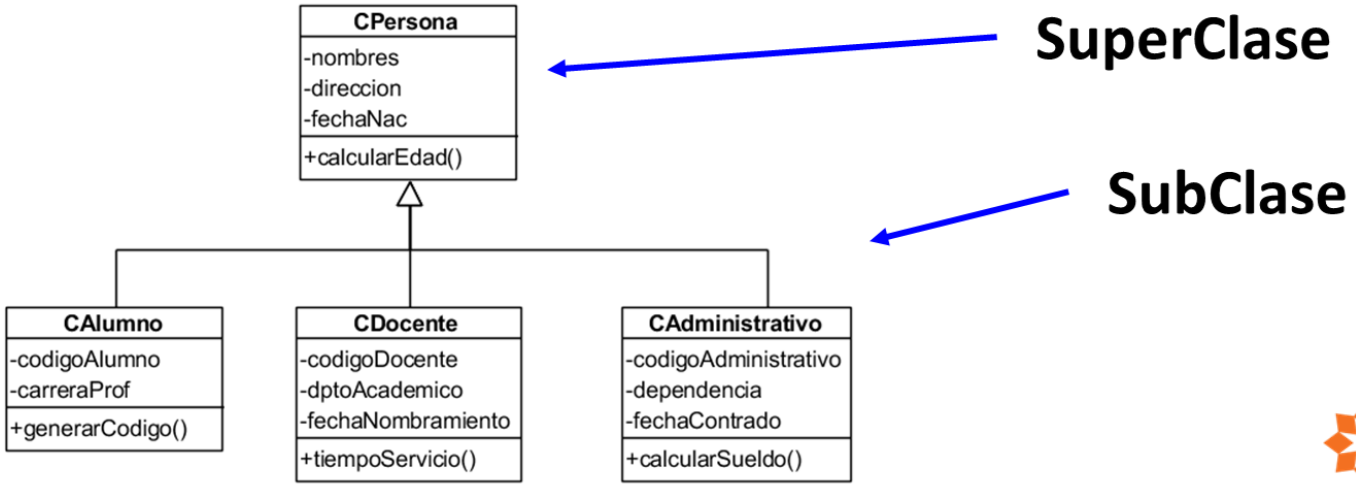
- Una característica de la Herencia simple es que las clases hijas heredan de una sola clase Padre. Sin embargo, existe la herencia padre donde una clase hija hereda de varias clases padre.



UNIVERSIDAD DE LIMA

2. Superclases y Subclases.

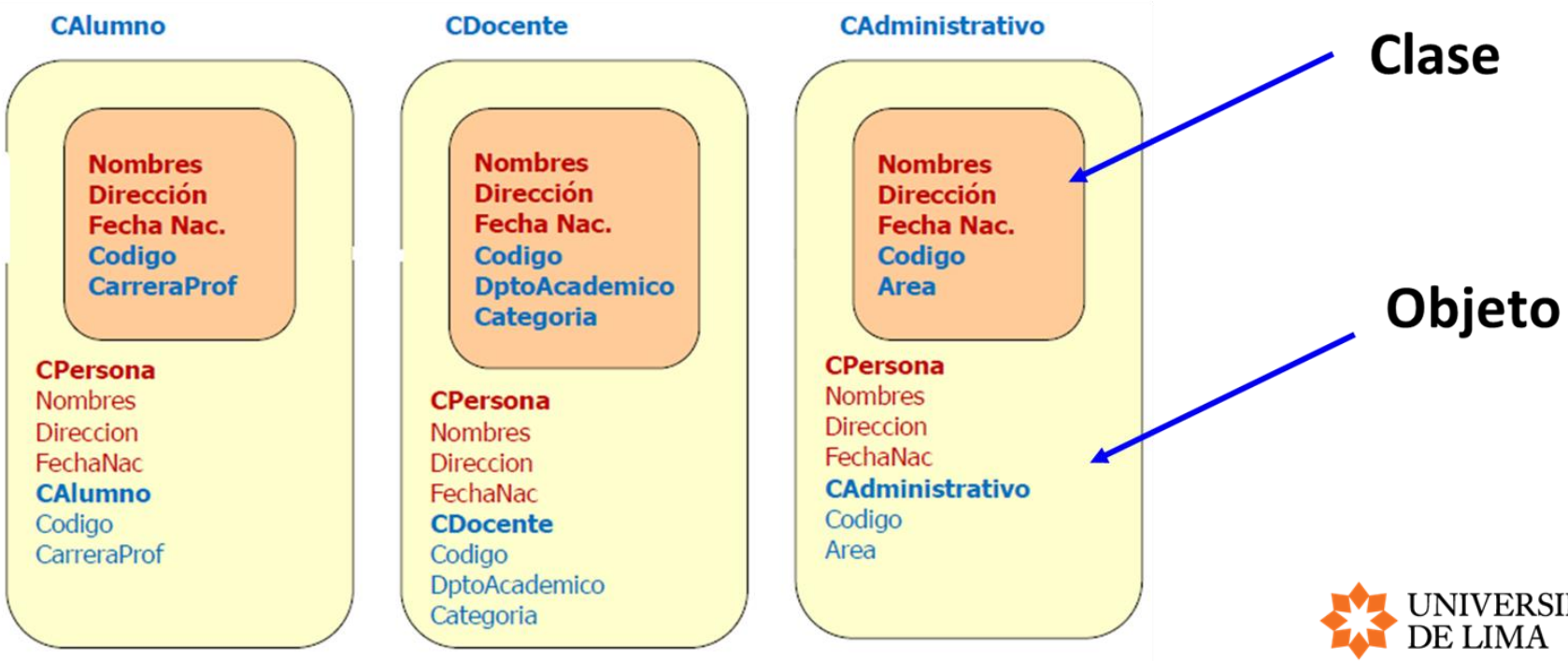
- La clase que contiene atributos y métodos comunes de otras clases es llamada **SuperClase o Clase Padre**. La clase que hereda o extiende de las características y métodos de una SuperClase se llama **SubClase** o **Clase Hija**.
  - Ejemplo: Primero se implementa la clase **CPersona (Superclase)** con las características comunes a CDocente, CAlumno y CAdministrativo. Luego se implementan las clases: CDocente, CAlumno y CAdministrativo (**Subclases**) como herederos de CPersona, agregando los atributos y métodos específicos a cada caso.



UNIVERSIDAD DE LIMA

2. SubClases y SuperClases (Continua...)

- En tiempo de ejecución, al crear objetos de la clase CAlumno, éste tendrá los atributos y métodos heredados de CPersona y los implementados en CAlumno.

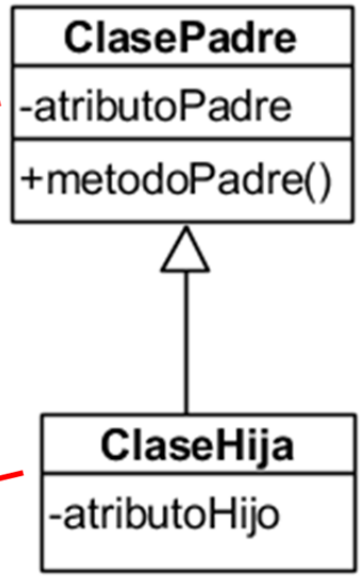


UNIVERSIDAD DE LIMA

2. SubClases y SuperClases (Continua...)

```java
public class ClasePadre {
    // ATRIBUTOS
    private int atributoSuperClase;
    // MÉTODOS
    public ClasePadre() { // Constructor de ClasePadre
        this.atributoSuperClase = 0;
    }
    public void metodoClasePadre(){
        this.atributoSuperClase++;
    }
}
```

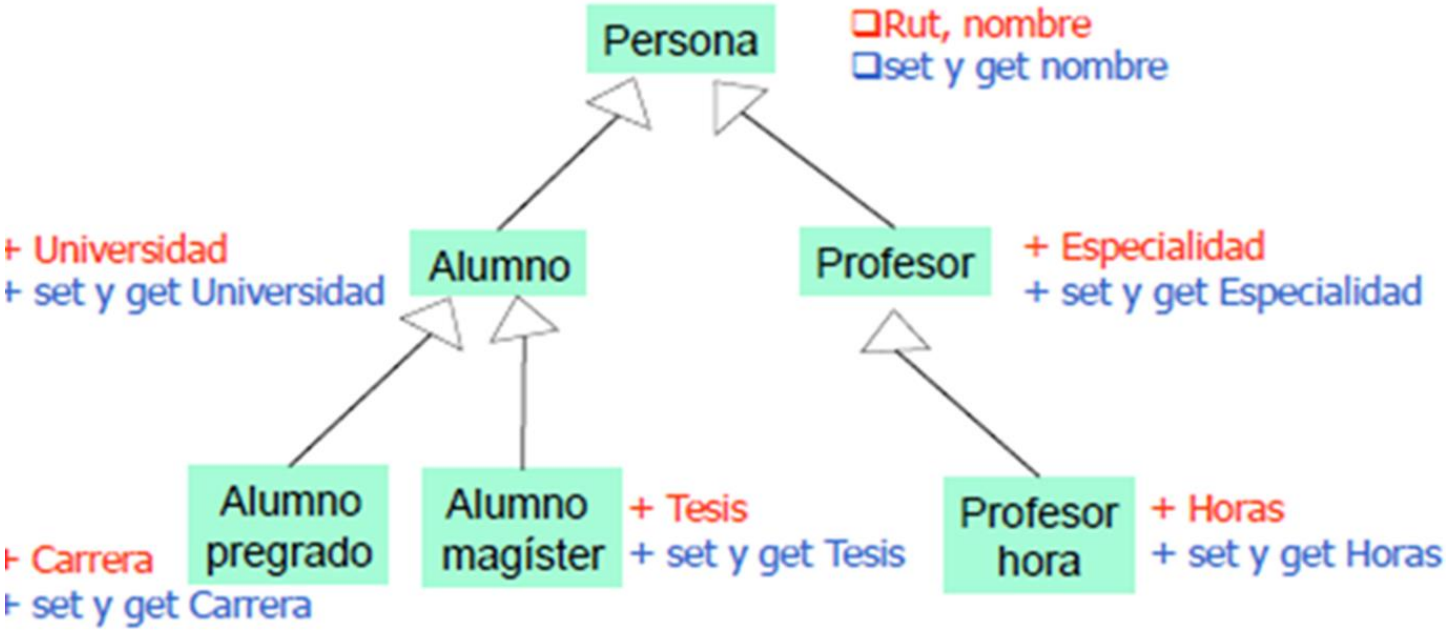
```java
public class ClaseHija extends ClasePadre{
    // ATRIBUTOS
    private int atributoClaseHija;
    // MÉTODOS
    public ClaseHija() { // Constructor de ClaseHija
        super(); // invocar el constructor del Padre
    }
}
```



UNIVERSIDAD DE LIMA

3. Construcción de una Jerarquía de clases

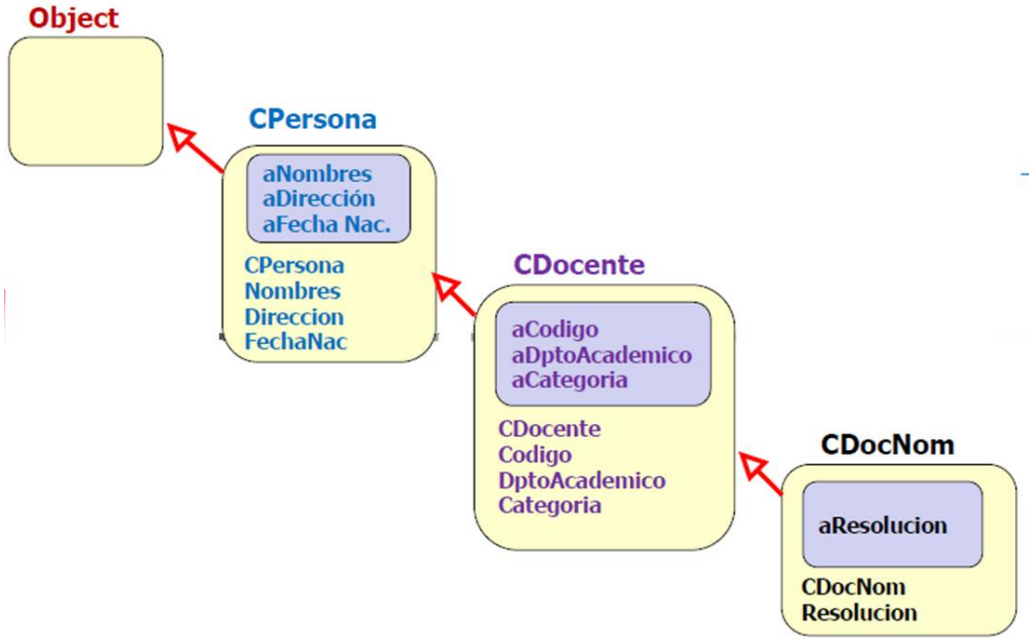
- Se pueden crear jerarquías de Clases o Tipos
  - Ejemplo de la clase persona y sus derivados



UNIVERSIDAD DE LIMA

3. Construcción de una Jerarquía de clases (Continua...)

- En el lenguaje Java la Clase Object es el padre de todas las clases y las nuevas que se crearan.



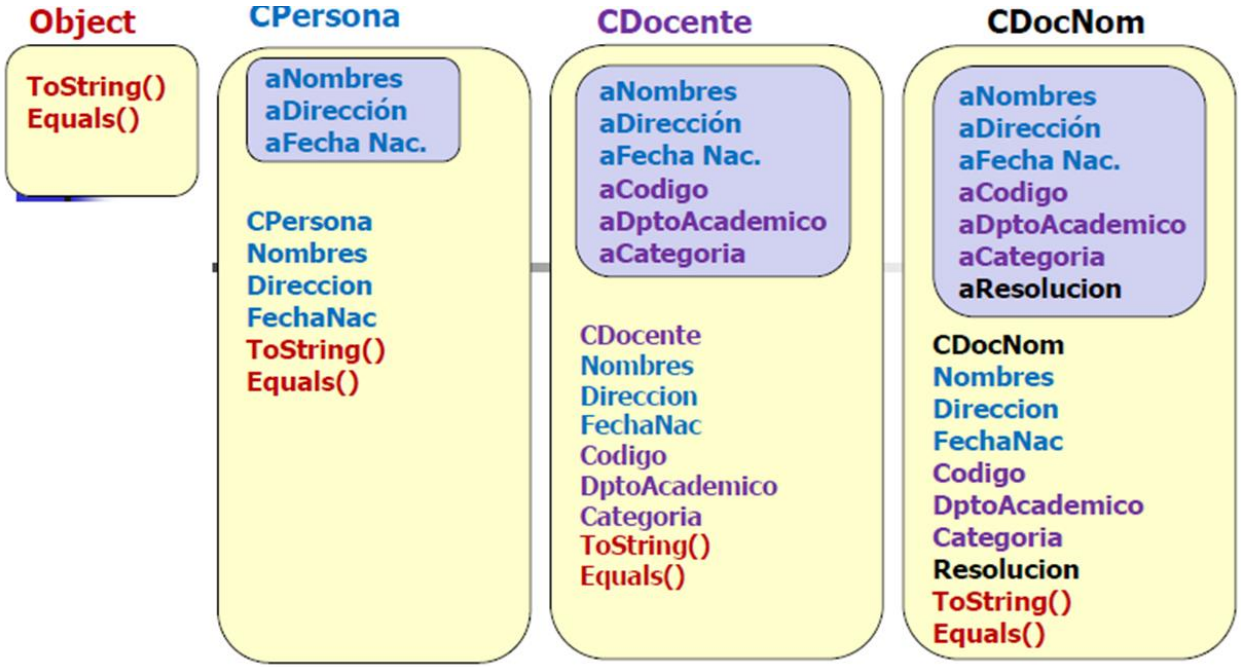
Se puede implementar varios niveles de jerarquía.

- Object es la superclase de todas las clases
- CPersona es la superclase de CDocente.
- CDocente es la superclase de CDocNom
- CDocNom es la subclase de CDocente
- CDocente es la subclase de CPersona

UNIVERSIDAD DE LIMA

3. Construcción de una Jerarquía de clases (Continua...)

- Considerando el ejemplo anterior:
  - Cada subclase hereda las características de su respectiva superclase. En consecuencia, la clase CDocNom hereda los atributos y métodos de todos sus ancestros.



UNIVERSIDAD DE LIMA

3. Construcción de una Jerarquía de clases (Continua...)

- El acceso protected permite que los atributos o métodos calificados de esa manera en la clase padre, sean accesibles en todas las clases hijas.

```
public class Profesor
{
    protected String nombre;
    protected String apellido;
    protected int edad;
    protected int id;
```

Criterios de accesibilidad

Modificador	La misma clase	El mismo package	Subclase	Universo
private	Yes			
default	Yes	Yes		
protected	Yes	Yes	Yes	
public	Yes	Yes	Yes	Yes

UNIVERSIDAD DE LIMA

4. Constructores en la Herencia

- **Los constructores no se heredan**, las subclases deben definir su propio constructor.
- Normalmente será necesario inicializar los atributos de la superclase
- Para ello se llama a su constructor desde la subclase

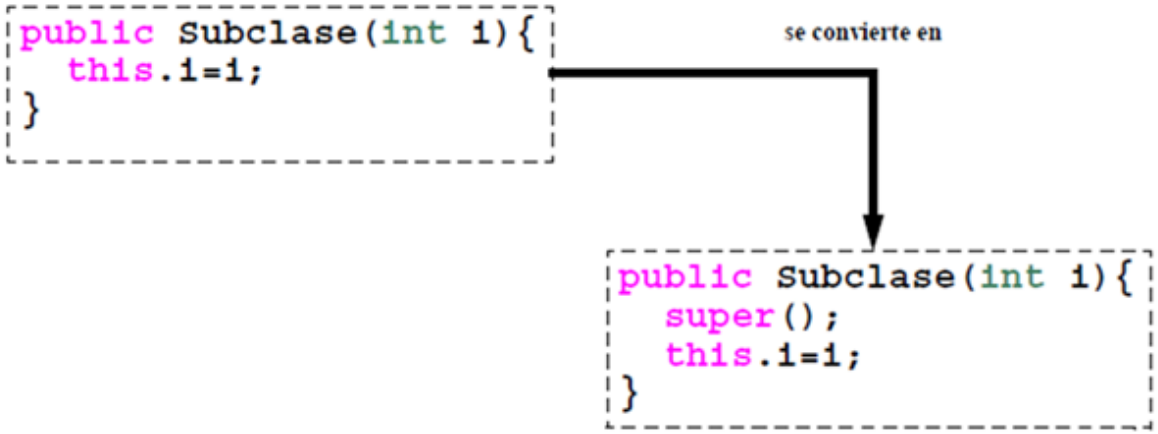
```
/** constructor de una subclase */
public Subclase(parámetros...) {
  // invoca el constructor de la superclase
  super(parámetros para la superclase);
  // inicializa sus atributos
  ...
```

- La llamada a super debe ser la primera instrucción del constructor de la subclase

UNIVERSIDAD DE LIMA

4. Constructores en la Herencia (Continua ....)

- En caso desde un constructor de una subclase **no** se llama expresamente al constructor de la superclase, el compilador añade la llamada al constructor sin parámetros o default



- En el caso de que la superclase no tenga un constructor sin parámetros se produciría un error de compilación.

UNIVERSIDAD DE LIMA

4. Constructores en la Herencia (Continua ....)

- La palabra reservada **super** sirve para indicar que una variable o método es de la superclase. Así mismo, **super** se usa para llamar al constructor de la clase padre.

```
class Persona {
    private String nombre;
    private int edad;

    public Persona (String nombre, int edad){
            this.nombre = nombre;
            this.edad = edad;
    }

     // getter y setter

}
```

```
class Alumno extends Persona{
   private int ciclo;

  public  Alumno (String n, int e, int c){

            super (n, e);

            this.ciclo =c;
   }
   //getter y setter

}
```

UNIVERSIDAD DE LIMA

5. Métodos redefinidos

- Una subclase puede modificar el comportamiento heredado de una clase padre.
- Una subclase puede redefinir el método de la clase padre; siempre y cuando en la SubClase y SuperClase el método tiene el mismo Nombre, Tipo de retorno y Lista de argumentos.

```
class Persona {
    private String nombre;
    private int edad;
    //.....................

    public String mostrarDetalle() {
            return nombre + "   " + edad;
    }

}
class Alumno  extends Persona{
     private int ciclo;
     //.....................
     public String mostrarDetalle() {
             return super.mostrartDetalle()  + " " + ciclo ;
     }

}
```

La clase hija redefine el método mostrartDetalle()

5. Métodos redefinidos

- Una subclase puede modificar el comportamiento heredado de una clase padre.
- Una subclase puede redefinir el método de la clase padre; siempre y cuando en la SubClase y SuperClase el método tiene el mismo Nombre, Tipo de retorno y Lista de argumentos.

```
class Persona {
    private String nombre;
    private int edad;
    //.....................

    public String mostrarDetalle() {
            return nombre + "   " + edad;
    }

}
class Alumno  extends Persona{
     private int ciclo;
     //.....................
     public String mostrarDetalle() {
             return super.mostrartDetalle()  + " " + ciclo ;
     }

}
```

La clase hija redefine el método mostrartDetalle()

Referencia

- Deitel, H. M. (2016). Java: como programar.
- Arturo Rozas Huacho, Algoritmos y Estructuras de Datos, 2002, Grupo Liebre, Cusco, Perú.

UNIVERSIDAD DE LIMA