UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Curso: Programación Orientada a Objetos

Sesión 2:

Encapsulación. Modificadores de acceso

UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Curso: Programación Orientada a Objetos

Indicador de Logro

Al finalizar la unidad, el estudiante aplica los conceptos básicos de la programación orientada a objetos, el concepto de herencia, relaciones entre clases, en la solución de problemas usando Java.

UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Curso: Programación Orientada a Objetos

Importancia

La **utilidad del encapsulamiento** va por la **facilidad para manejar la complejidad,** ya que tendremos a las **Clases como cajas negras** donde **sólo se conoce el comportamiento pero no los detalles** internos, y esto es conveniente porque **únicamente deberíamos de conocer qué hace la Clase** pero **no** será necesario saber **cómo lo hace**.

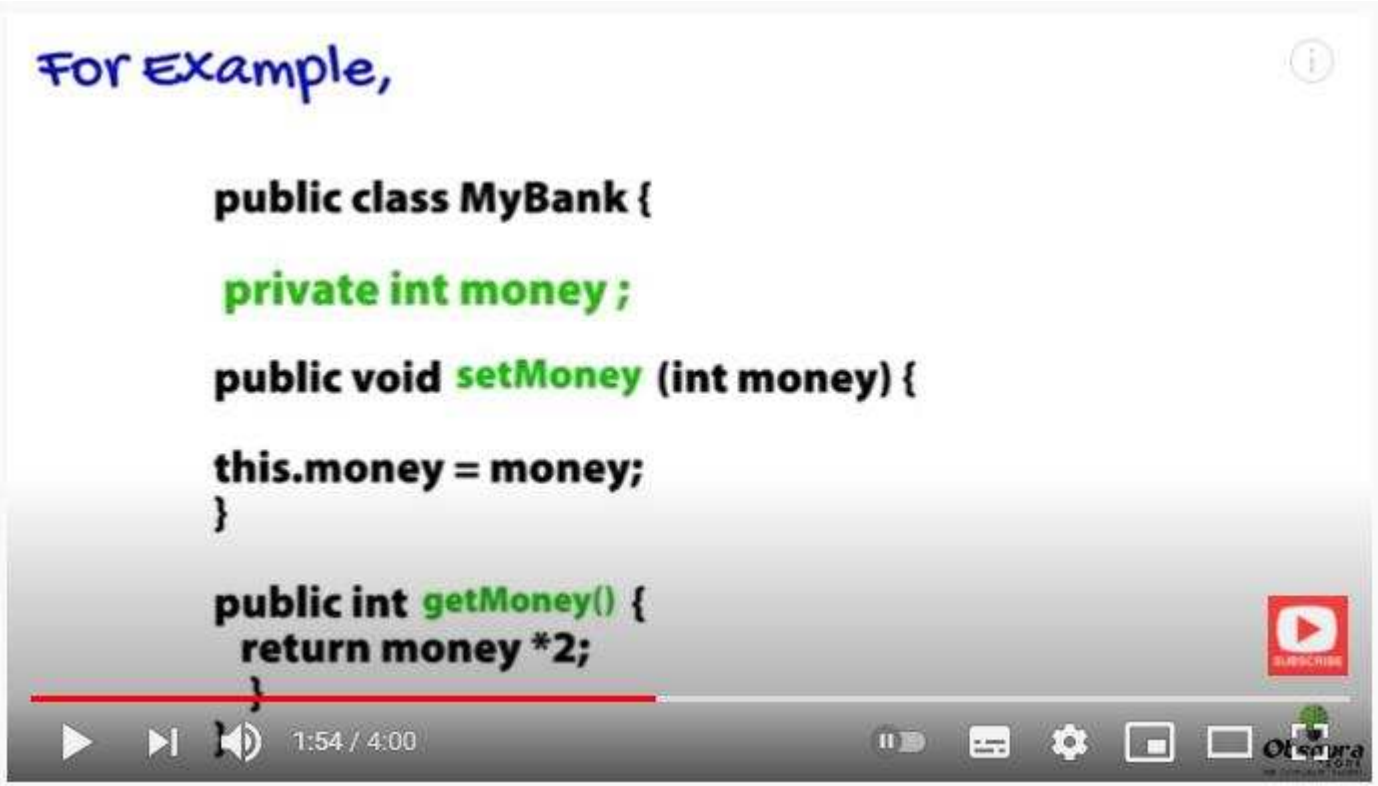
UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Curso: Programación Orientada a Objetos

Sesion 02: Contenido General

- Modificadores de acceso.
- Constructores
- Encapsulamiento.
- Sobre escritura de métodos.



Fuente: https://www.youtube.com/watch?v=QxRYnPm-HOA

Modificadores de acceso en Java

Los modificadores de acceso, **determinan desde qué clases se puede acceder a un determinado elemento.** En Java tenemos 4 tipos: **public**, ***private***, **protected** y el tipo *por defecto* (que no tiene ninguna palabra clave asociada)

	La misma clase	Otra clase del mismo paquete	Subclase de otro paquete	Otra clase de otro paquete
public	X	X	X	X
protected	X	X	X	
default	X	X		
private	X			

Modificadores de acceso en Java

	La misma clase	Otra clase del mismo paquete	Subclase de otro paquete	Otra clase de otro paquete
public	X	X	X	X
protected	X	X	X	
default	X	X		
private	X			

```java
package p1;
public class A {
    public void mostrar()
    {
        System.out.println("Hola Mundo!");
    }
}
```

```java
package p2;
import p1.A;

public class B extends A
{
    public static void main(String args[]){
        B obj = new B();
        obj.mostrar();
    }
}
```

Clase compilada - no hay errores de sintaxis

Modificadores de acceso en Java

	La misma clase	Otra clase del mismo paquete	Subclase de otro paquete	Otra clase de otro paquete
public	X	X	X	X
protected	X	X	X	
default	X	X		
private	X			

```java
package p1;
public class A {
    protected void mostrar()
    {
        System.out.println("Hola Mundo!");
    }
}
```

```java
package p2;
import p1.A;

public class B extends A
{
    public static void main(String args[]){
        B obj = new B();
        obj.mostrar();
    }
}
```

Clase compilada - no hay errores de sintaxis

Modificadores de acceso en Java

	La misma clase	Otra clase del mismo paquete	Subclase de otro paquete	Otra clase de otro paquete
public	X	X	X	X
protected	X	X	X	
default	X	X		
private	X			

```java
package p1;
class A {
    void mostrar()
    {
        System.out.println("Hola Mundo!");
    }
}
```

```java
package p2;

import p1.*;
public class B {
    public static void main(String args[])
    {
        A obj = new A();
        obj.mostrar()
    }
}
```

Unknown type: A
- Fix: Correct to: B
- Fix: Correct to: Map (java.util package)
- Fix: Correct to: Tab (javafx.scene.control package)

Modificadores de acceso en Java

	La misma clase	Otra clase del mismo paquete	Subclase de otro paquete	Otra clase de otro paquete
public	X	X	X	X
protected	X	X	X	
default	X	X		
private	X			

```java
package p1;
class A {
    private void mostrar()
    {
        System.out.println("Hola Mundo!");
    }
}
```

```java
package p1;

class B{
    public static void main(String[] args) {
        A obj= new A();
        obj.mostrar();
    }

}
```

mostrar() has private access in p1.A

Constructores en Java

Un ***constructor*** es un método perteneciente a la clase que posee unas características especiales:

- Se llama igual que la clase.
- No devuelve nada, ni siquiera void.
- Pueden existir varios, pero siguiendo las reglas de la sobrecarga de funciones.
- De entre los que existan, **tan sólo uno se ejecutará** al crear un objeto de la clase

Constructores en Java

```
public class Alumno{

    String nombre;
    int edad; //edad que tiene un alumno en el momento que finaliza su carrera.

    public Alumno(){
        edad = 0;
        nombre = "anónimo";
    }

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

    public void imprimir(){
        System.out.print(nombre+" a sus "+edad+" años ha concluido la carrera de Ing. Sistemas" );
    }
}
```

Deberían ser **private**

La única forma de manipular los atributos debería ser mediante métodos públicos get() and set()

Destructores en Java

El destructor

Un destructor es un método opuesto a un constructor, éste método en lugar de crear un objeto lo destruye liberando la memoria de nuestra computadora para que pueda ser utilizada por alguna otra variable u objeto.

El destructor se utiliza para destruir una instancia de una clase y liberar memoria. En Java no hay destructores, ya que **la liberación de memoria** es llevada acabo por el Garbage Collector cuando las instancias de los objetos quedan desreferenciadas.

ACTIVIDAD

Investigar en grupos: ¿Cómo funciona el Garbage Collector en Java? y explicar con ejemplos.

CUESTIONARIO

Resolver la actividad (Cuestionario):

G1: https://forms.gle/XZCQh5k1fpPpun5PA

G2: https://forms.gle/yR6dUa4EpBtxC8fe8

G3: https://forms.gle/HTEi61qFUUoHuife8

CIERRE

¿Qué hemos aprendido hoy?