

Taller de Java Orientado a Objetos

Por el GUI y para el GUI, espera, el GUI es un objeto?

29 de noviembre de 2023



1 ¿Qué es la programación Orientada a Objetos?

- Paradigma de la programación orientada a objetos
- Objeto
- JAVADOC
- Statics

2 Ejercicios OO

- Ejercicio 1
- Ejercicio 2

3 HERENCIA

4 Ejercicios Herencia

¿Que es la programación Orientada a Objetos?

La **Programación Orientada a Objetos** es un paradigma de la programación, en el que se basa en el concepto de **clases y objetos**. Esto quiere decir que hay que hacer un diseño **Modular**

Modularidad y criterios

Modularidad

Consiste en separar y estructurar el código en fragmentos más simples y reutilizables como lo son las clases y nos permite instanciar en objetos.

Criterios

Los criterios son utilidades para que en el código se...

- facilite la **descomposición de módulos**
- facilite la **combinación de módulos**
- facilite la **comprensibilidad de los módulos** sin conocimiento sobre los otros módulos
- continuo y facilite que cambios pequeños tengan pequeñas consecuencias
- **proteja** y dificulte que los errores de un módulo afecten a otros módulos

Modularidad y reglas

Reglas

A partir de lo anterior mencionado se generan unas reglas a seguir:

- **Correspondencia directa** para que haya un único concepto de módulo
- **Pocas interfaces** para que un módulo se comunique con pocos módulos
- **Interfaces pequeñas** para que la comunicación entre módulos sea mínima
- **Interfaces explícitos** para que la comunicación sea obvia
- **Ocultación de información** puesto que solo se mostrará al público los justo

Principios de la modularidad

El resultado de juntar los criterios y las reglas es que nacen los principios, en este caso de la modularidad:

Principios

- Los módulos deben ser **unidades ligüística**
- El módulo tiene que estar **auto-documentado**
- El módulo tiene un **acceso uniforme** es decir que sus servicios no distinguen almacenamiento de calculo
- El módulo ser **cerrado** para su uso, pero **abierto** para su modificación
- Si hay un conjunto de alternativas, solo un módulo conocerá la lista completa

PERO, ¿QUÉ ES UN OBJETO?

¿Qué es un objeto?

Antes de poder descifrar esta pregunta primero tenemos que definir qué es una clase, porque el objeto es una instancia de esta

Tipo de Dato Abstracto

Un conjunto de valores y operaciones asociadas especificados de manera precisa e independiente de la implementación

Clase

Es un Tipo de Dato Abstracto (TAD - Tipo Abstracto de Datos) dotado de una implementación. También cabe decir que las clases se organizan en directorios que son paquetes o también llamados package. Y por último, una clase es un **módulo**

¿Qué es un objeto?

Finalmente podemos decir que es un objeto:

Objeto

Un objeto es una instancia de una clase, es decir, es una variable a la que apunta a la clase pero definida con unos valores.

Quizás no quede claro...

¿Qué es un objeto?



Creación de una clase

```
public class {
    private boolean abierta;
    private double aperturaVentana;
    //Esto seria el constructor de la clase
    public Ventana(boolean abierta, double apertura){
        this.abierta = abierta;
        this.aperturaVentana = apertura;
    }
    public void abrirVentana(double grados){
        if (grados > 0){
            abierta = true;
            aperturaVentana = grados;
        }
        else{
            abierta = false;
            aperturaVentana = grados;
        }}}
```

Ahora vamos a usar la clase Ventana en nuestro main:

```
public static void main(String [] args){  
    //Llamada al contructor  
    Ventana casa = new Ventana();  
  
    if(casa .getAbierta ()){  
        casa .abrirVentana (0);  
    }  
}
```

Constructor

Ventana casa = new Ventana();

Esta es la instancia, aquí es donde se genera la magia del objeto, a partir de aquí la variable casa es un objeto. Por tanto un objeto es una variable que puede usar y/o acceder a todas las características de una clase. Todas menos los Main que nos conocemos

Protecciones a tomar

protecciones

Para que no le pase nada a nuestra la clase, tenemos que declarar nuestros atributos de la clase con “private” delante y nos aseguraremos que los datos que se introduzcan sean validos, es decir, no consideraremos que los vayan a implementar bien. Aunque esto es de más adelante...

Todo esto esta muy bien, pero como vamos a saber que es lo que tiene una clase, que puede hacer que métodos tiene. Para ello usaremos javadoc, espera ¿java que?

Javadoc

Antes hemos hablado de que uno de los principios debía ser que estubieran autodocumentados, ¿no?

javadoc

Javadoc es una herramienta de java, para ayudarnos a documentar los usos que tiene cada método que forman las clases o que parámetros hacen falta para invocar a una función, esto facilita nuestro trabajo y el de nuestros compañeros.

Un ejemplo con la clase Ventana sería:

```
/**  
 * Contructor de la clase Ventana  
 *  
 * @param abierta , true si esta abierta false lo contrario  
 * @param apertura , grados de apertura si esta abierta  
 */  
public Ventana(boolean abierta , double apertura){  
    this.abierta = abierta ;  
    this.aperturaVentana = apertura ;  
}
```

```
/**  
 * Abre la ventana tantos grados y si es 0 se cierra  
 *  
 * @param grados , grados que se va a abrir la ventana  
 */  
public void abrirVentana(double grados){  
    if (grados > 0){  
        abierta = true;  
        aperturaVentana = grados;  
    }  
    else{  
        abierta = false;  
        aperturaVentana = grados;  
    } } }
```

Static

En java básico usáis public **static** void metodo(){}

Static

El static es un “comando” para hacer una declaración estática, es decir, que no va a poder variarse en las diferentes clases, si no que mantendrá un único valor, para todos los objetos y si este es modificado, se modificaría en todos.

ejemplo

El numero de banquetas que tenemos en las aulas son 6. Si lo modificamos, y dijésemos que solo haya 5 banquetas, de repente todas las banquetas de las aulas perderían 1 asiento.

Ejercicios

1º Ejercicio

Generar una clase pistola, con los siguientes atributos:

- modelo o tipo
- balas
- recamara

Y hacer los siguientes métodos:

- Dar el tipo
- Dar cuantas balas hay o quedan
- Meter bala en la recamara
- Disparar arma

Por si acaso el arma no puede disparar sin tener una bala en la recamara

Ejercicios

2º Ejercicio

Hacer que todas las armas sean el mismo **modelo o tipo** siempre y hacer el javadoc del ejercicio anterior

Hasta aquí

Hasta aquí hemos llegado con la presentación sobre programación orientada a objetos básico.

AHORA COMIENZA LO DIVERTIDO

¿Papá?

Que seríamos nosotros sin nuestros padres, literalmente nada, pues en programación tenemos algo similar a árboles genealógicos.

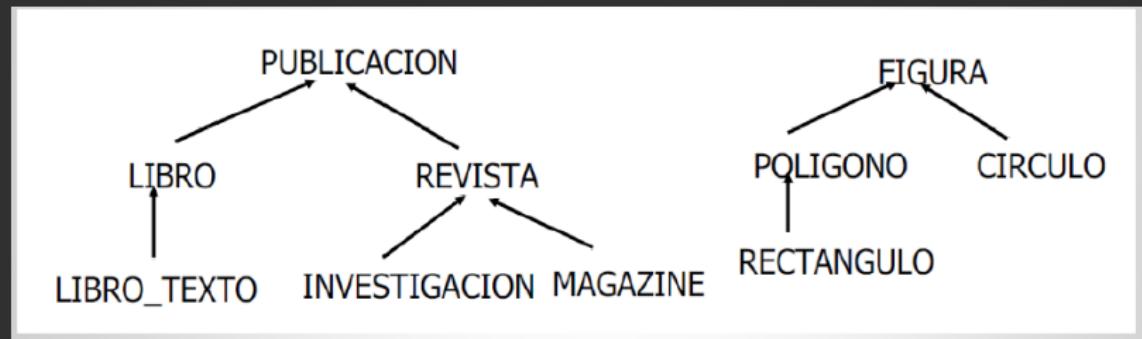
Herencia

La herencia no es otra cosa que heredar uno de otro, pero en un solo sentido. Como nosotros heredamos nuestros genes de nuestros padres, pero en vez de heredar genes, en este caso heredamos funciones.

Herencia

Es una forma de definir clases a partir de clases ya implementadas, es decir, copiar el código de otra clase en esta

Herencia



Herencia

Para aplicar Herencia en java:

relación Padre-Hijo

Para que el hijo reconozca al padre tendremos que introducir “**extend**” con esto decimos que esta clase extiende de otra (o es hija de esta otra) y por ello podemos usar sus métodos.

Super()

Usaremos la declaración super() en nuestro constructor, introduciendo entre los paréntesis los atributos que el constructor padre necesite.

Super()

La declaración “**super()**”, lo que nos permite es guardar los atributos que heredamos del padre, este los pueda guardar, como veréis a continuación:

Hijo

```
public class Pistola extend arma{  
    private int annosUsada;  
    private int disparosHechos;  
  
    public Pistola(int annos , int disparos ,  
                  String mod, int balas){  
        super(balas , mod);  
        annosUsada = annos ;  
        disparosHechos = disparos ;  
    }  
}
```

Padre

```
public class arma{  
    private int balasCargador;  
    private boolean balaRecamara;  
    private String modelo;  
  
    public arma(int balas , String mod){  
        balasCargador = balas;  
        balaRecamara = false;  
        modelo = mod;  
    }  
  
    public void disparar(){  
        balaRecamara = false;  
        balasCargador--;  
    }  
}
```

Resultado

Con esto conseguimos que cuando generamos el objeto pistola, esta pueda usar también las funciones del padre.

Todo esto de la herencia conseguimos que se sigan cumpliendo los principios de la modularidad

Ejercicios

Generar clase

- clase padre = persona
 - DNI
 - Edad
 - método cumpliraños()
- clase hija = Estudiante
 - DNI
 - Edad
 - Curso
 - nºAsignaturas
 - nºAprobados
 - método aprobarAsignatura y cumplirños()
- clase hija = Profesor
 - DNI
 - Edad
 - Asignatura
 - nº AlumnosSuspensos
 - método suspenderAsignatura(Estudiante) y cumpliraños()

AUNQUE PUEDE, nada era broma hasta aquí el taller,
DUDAS, no? Si no mi @ de telegram es @cadore33 y si
no mi hogar es el GUI esa salita (ErZuloMiAlrma) que hay
nada más subir las escaleras.