¿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, ¿QUE 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

Abstracción de un concepto que contiene tantos datos como comportamientos

Clase

Una clase tiene la intención de tener un comportamiento. Por ello es un Tipo de Dato Abstracto (TAD - Type of Abstract object). También cabe decir que las clases se organizan en directorios que son paquetes o también llamados package

¿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...

Creacion de una clase

```
public class Ventana{
  private boolean abierta;
  private double aperturaVentana;

  //Esto seria lo llamado constructor por defecto
  public Ventana(){
    abierta = false;
    aperturaVentana = 0;
  }
  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);
            }
        }
```

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 p̈rivate d̈elante 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:

[language = Java] public class Ventana private boolean abierta; private double aperturaVentana;
/** * Este es el constructor por defecto * Genera la clase Ventana con los parametros en false y a 0 **/ public Ventana() abierta = false; aperturaVentana = 0; public void abrirVentana(double grados) if (grados ¿ 0) abierta = true; aperturaVentana = grados; else abierta = false; aperturaVentana = grados; public boolean getAbierta() return abierta;

Ejercicios

1º Ejercicio

Generar una clase pistola, con los siguientes atributos:

- 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

2º Ejercicio

Hacer el javadoc del ejercicio anterior

Hasta aquí

Bueno hasta aquí hemos llegado con la presentación sobre programación orientada a objetos

¿O no? Antes de nada, ¿que pasaría si al arma le meto -5 balas? O si diparase balas en negativo?, ¿curarían? Por desgracia, no. Pero aquí seguimos con la siguientes parte