Pilares de la orientación a Objetos:

TEMARIO:

1.Abstraccion

2.Encapsulamiento

3.Herencia

4.Polimorfismo

18/01/18

Lo estructural no acepta cambios, ya que tendrías que volver hacer la app y lo orientado a objetos se adapta a cambios;

**c++=** es orientado a objetos.

**c** = es estructurado.

\*Grouvy

\*Phyton(Muy Facil)

\*Kutlin(puede desvancar a java)

\*Scala(mejor pagado y poderoso)

\*Typescript

\*Todos son orientados a objetos.

***\*ABSTRACCION\****

--**Abstraccion**=> Es delimitar el dominio del problema y traucirlo a orientacion a Objetos.

--En la orientación a objetos en Java solo existen 3 tipos de clases, las cuales son:

\***Aplicacion**=> Es la que se ejecuta y solo debe de haber 1.

Es la que tiene el metodo main.

No tiene nada de logica y es por seguridad.

\***Modelo**=> Son clases logicas y hay varias.

No tienen que ver nada con el desktop(botones, hypervinculos, textbox,nada visual).

ej:Guardar usuario,modificar usuario.

\***Entidad**=>Representan objetos concretos(nomina,usuario,cliente,trabajador,estudiante,biblioteca).

***Modelo e Identidad están en la aplicación***

Ejercicio: Calcular el imc basandonos en la sig formula:

imc=peso(kg)/(altura(m))^2

imc= indice de masa corporal

peso ideal 20<imc<=25

sobrepeso imc>25

flaco imc<=20

Usuario tiene(altura,imc,peso,resultado)

**Atributo**= necesita solo un valor, más de dos es una "**CLASE**".

**Clase**= Es de dos o más valores.

|  |
| --- |
| PooEiam |
| main |

Aplicacion

|  |
| --- |
| Usuario |
| peso : float  altura: float |

Los atributos son primitivos. Es la Entidad

|  |
| --- |
| Imc |
| usuario:Usuario  calcular: String |

El de Usuario es referenciado. Es el Modelo

**Alto Nivel**= Es diseño en general, es hacerlo en UML.

**Bajo Nivel**= hacer las plantillas

22/01/18

**Abstracción:** 3 tipos de clases: Aplicación (main), Módulos (IMC), Entidades (Usuario)

**Encapsulamiento:** Todas las clases entidad se encapsulan. Es el ocultamiento de la información.

Se aplican estas 3 reglas para el encapsulamiento:

1. Los atributos deben tener modificador **private.**
2. Por cada atributo generar un método **set**, para asignar valores de dicho atributo *desde fuera de esta clase* deben tener modificador **public.**

*Se ocupa el método* ***set*** *por cada atributo que tengas.*

*// "this" es para asignar cual es el atributo y no se confunda con la variable local*

1. *Por cada atributo generamos un método* **get** para acceder al valor del atributo en el momento que necesitemos con su valor. (Métodos Accesorios)

**API:** Es la documentación y jamás es para el cliente, es para el desarrollador.

Lógicas de programación:

* Transaccional🡪No cambia la base de datos.(Servidor Web)
* Persistencia🡪Si cambia la base de datos.(Base de Datos)

Proyecto: c# + unity

Vamos a hacer un video juego **A I** (Artificial Intelligence) + **VR**(Virtual Reallity)

25/01/18

TIPOS DE DATOS

Primitivos Referenciados

Todos los de un tipo de clase existente

|  |  |  |  |
| --- | --- | --- | --- |
| Char | Decimales | Enteros | Booleanos |
| Carácter  ‘ ‘ | Float 2^32  Doublé 2^64  (defecto) | Byte 2^8  Short 2^16  Int 2^32(def)  Long 2^64 | True/false |

**Primitivos**: porque son los primeros del lenguaje

**Referenciados:** porque están referenciados a una clase.