**Clase 05 – Programador Java**

**Herencia – Clase 05 Java**

Es importante el concepto de Polimorfismo, al crear herencia aplicas polimorfismo

* Muchos de los frameworks de Java usan herencia
* Objetivo: disminuir el costo de “mantenimiento” creando una buena jerarquía de clases.
* Java no cuenta con herencia múltiple debido a la sencillez.

Los diseños de jerarquía para crear son de 2 formas

* Generalización y Especializacion
* No existe una receta mágica para crear buenas jerarquías
* Cumplir con los principios de Discot sobre jerarquía de clases. Porque el principio debe estar enunciado.
* “LA JERARQUIA DE CLASES DEBE CORRESPONDER A LA JERARQUIA DE OBJETOS” Principio de Discot.

Para ver la herencia gráficamente: Navigate -> Inspect Hierarchy

Clase4 a = new Clase4();

Clase2 b = a;

Clase1 c = a;

Object d = a;

+ restar()

+ sumar() OVERRIDE

+ sumar()

Clase4 -> a

Clase2 -> b

Clase1 -> c

Object -> d

Clase4

* a, b, c ejecutan el mismo método que hereda clase 4
* Pero c y d no pueden llamar a resta.

Cuando pones un final a un método, clase o atributo este no se puede heredar a los hijos, los métodos y variables no se puedes sobrescribir, las variables se convierten en constantes.

Parcialmente abstracto es aquella clase abstracta que también tiene métodos comunes.

El objetivo de instanceof es determinar si la instancia creada es compatible con la clase del cual es juzgada, y con compatible quiero decir si una clase es padre de la clase de la instancia.

Clase2 obj = new Clase2();

System.out.println("Clase 4: "+((obj instanceof Clase4)?"SI":"NO")); //NO

System.out.println("Clase 3: "+((obj instanceof Clase3)?"SI":"NO")); //NO

System.out.println("Clase 2: "+((obj instanceof Clase2)?"SI":"NO")); //SI

System.out.println("Clase 1: "+((obj instanceof Clase1)?"SI":"NO")); //SI

System.out.println("Object: "+((obj instanceof Object)?"SI":"NO")); //SI

* Con esto queda detallado que hay casting de tipos y casting de clases. LOL

Proyecto de ejemplo: Para el problema vamos a aplicar los sig. Conceptos:

* DTO
* ARREGLO
* Simple Factory
* Excepciones
* Herencia
* Clase Abstracta

PARA LA PROXIMA CLASE: traer el proyecto anterior (tomar de lo que avanzo el profe) terminado.