Laboratorio 1

Clase desde 18:45hs

Objetivos:

* Paradigma orientado a objetos
* UML (diagrama de clases y secuencia)
* Patrones
* Programación en java

Evaluación:

Parcial (conceptos teóricos): 08/05

Trabajo Practico (MVP): 12/06

Recuperatorio 1 parcial: 12/06

Recuperatorio TP: 19/06

Recuperatorio fuera cursada: 26/06

Final: Defensa del trabajo practico (trabajo completo)

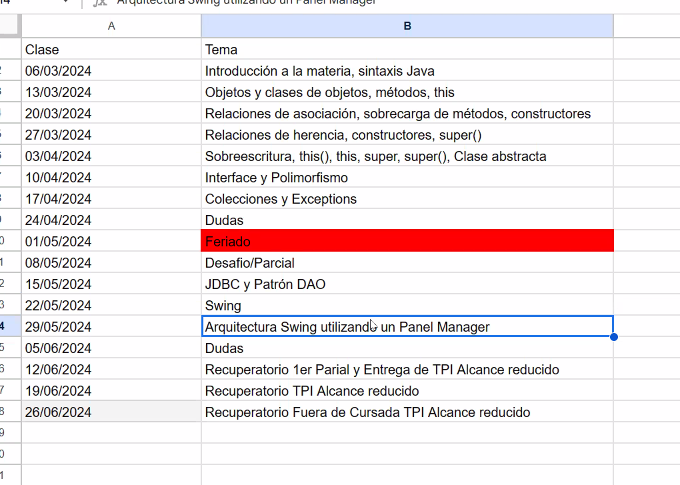
Temas a saber:

* Polimorfismo

Herramientas:

* Intellij IDEA

Cronograma:



Clase 1:

Todas las clases deben estar dentro de src.

Java es tipado, hay que indicar el dato a utilizar.

Tiene datos primitivos y clases. Una clase se puede usar como un tipo de dato

Clase 2:

Objetos y clases de objetos.

Objeto: Es la instancia de una clase.

Clase: Molde de la instancia.

Tiene:

Propiedades: (NO son variables) Valor que se puede guardar en un objeto

* Atributos
* Características
* Propiedades

Comportamiento:

* Responsabilidades/Operaciones que podemos hacer sobre los objetos

Métodos: Es una acción de un objeto. No es lo mismo que una función.

Los métodos y propiedades tienen que corresponder al objeto que queremos crear.

Abstracción: Describir un objeto con sus atributos y responsabilidades ignorando otras.

Que puedo hacer, como se orienta la clase pero no definir el COMO se puede hacer.

Diferencia entre el QUE y el COMO se puede hacer. Se puede agachar, pero no se sabe COMO se puede agachar.

Diferentes contextos pueden tener mismas clases pero distintas propiedades.

Una acción es un método y a su vez un verbo es una acción.

Herencia: Son clases que suman propiedades y métodos que provienen de una clase padre.

Clase 3:

Clases, métodos, propiedades, getters y setters es hasta esta clase enfocado en herencia.

Bloque: todo el código fuente dentro de llaves {}

Constructor: Sirve para construir el objeto de la clase. New se encarga de llamar al constructor.

Siempre que se crea una clase extiende de la clase Object.

Herencia: Permite heredar a la clase que hago referencia..

Multi-herencia: Java no permite que una clase extienda de 2 superclases.

Cosas a tener en cuenta en JAVA:

Hay que acceder a las propiedades mediante métodos.

Getters: GET retorna el valor de una propiedad

Setters: SET le otorga un valor a una propiedad

Las propiedades se ponen privadas para no acceder a una propiedad directamente desde el objeto.

No tiene sentido definir una superclase teniendo una subclase.

Constructor: Tiene las propiedades de la clase, pero no las extendidas.

Super: Si se construye un objeto subclase tiene que ser antes un objeto superclase, es la sobreescritura. Super() permite setear las propiedades de una superclase en una subclase.

Se puede tener varios constructores, pero no conviene tener los que no se usen.

NO es buena practica enviar null en un constructor.

Las clases es recomendable tener al menos un constructor.

Diferencia entre constructor y setter: Un constructor inicializa el objeto y se puede llamar una sola vez, el setter se puede llamar varias veces a lo largo de la vida del objeto.

Parámetros de tipo clase: Se le da como parámetro al método un objeto.

Una vez que el modelo ya está armado puede salir muy caro.

Clase 4:

Herencia: relación “es un”

“En un” entre clases. Si tienen y hace todo lo que hace un perro entonces es un perro.

Un caniche es un perro.

La herencia es un ordenamiento entre clases que define una relación “es un”.

Ventajas de la herencia:

* Evitar duplicidad: Cuando una clase deriva de otra, adquiere automáticamente sus métodos y atributos, no es necesario volver a programarlo.
* Reutilización de código
* Facilitar el mantenimiento: Si la definición de la clase PADRE cambia, solo se modifica el código de esa clase, los cambios automáticamente impactan en todas las clases derivadas
* Soluciones escalables: Que tan difícil será crear distintos tipos de clases hijas a futuro

Herencia multiple: En java no esta permitida

Se establece cuando una clase hereda de varias otras clases.

Super(): Entrega atributos a la clase que tenga encima.

UML: Atributos con – y métodos con +.

Polimorfismo:

Clases abstractas: No se puede instanciar.

Métodos abstractos: Pertenecen a la clase abstracta, al heredarlos hay que volverlos a redefinir.

Desventaja: Por cada método abstracto hay que definirlo si o si en la clase.