**Alumno:** Olivarez Alexis, Emanuel  **DNI**: 44542230 **Laboratorio**: 2

**Profesores:** Medina – Airaldi **Tp**: 2 **Fecha de Entrega**: 27/08/23

La programación orientada a objetos a diferencia de otros paradigmas se basa en el concepto de clases y objetos con características en común para que el código fuente se pueda reutilizar sea organizado y fácil de mantener. También puedo afirmar que adquirí conceptos como el encapsulamiento, la ocultación de información, la sobrecarga de métodos, la documentación con JavaDoc y abstracción, mediante ejercicios propuestos por el profesor.

Podemos ver encapsulamiento, así como también el doble encapsulamiento en clases, haciendo el uso de los métodos observadores y mutadores (Accessors: getters y setters). Esto está relacionado con la ocultación de información, ya que nos permiten el acceso a nuestros atributos privados de nuestra clase solo por medio de estos métodos. Al ser declarados privados, los atributos solo son accesibles mediante la misma clase. Los métodos mutadores(setters) nos permiten modificar estos atributos. Con el doble encapsulamiento, el propio objeto no accede directamente a sus atributos, sino a través de sus métodos. Ejemplo:

*public* *double* agregaSaldo(*double* *pImporte*) {

*this*.setSaldo(*this*.getSaldo() + pImporte);

*return* *this*.getSaldo();

}

En cuanto a la abstracción, se habla del hecho de mostrar solo los aspectos relevantes en ciertos contextos, y destacar los demás que no son tan importante para resolver el problema. Por ejemplo, para hacer uso de los métodos de un objeto, solo tenemos que conocer que datos pasarle, sin entender cuál es el proceso de fondo.

La sobrecarga de constructores permite definir varias formas de inicializar un objeto con un diferente ingreso de parámetros. Esto mejora la flexibilidad y la comodidad al crear instancias de una clase con diferentes configuraciones.

**JavaDoc** nos permite mostrar un resumen de los métodos y constructores que tendrá cada clase.

/\*\*

*\* El método "adicional" Calcula el monto adicional en función de los años de*

*\* servicio del empleado.*

*\* @return Devuelve la cantidad adicional basada en los años de antigüedad del*

*\*         empleado y su salario básico (sueldoBasico).*

\*/

*private* *double* adicional() {

*if* (*this*.antiguedad() < 2) {

*return* *this*.getSueldoBasico() \* 0.02;

    } *else* *if* (*this*.antiguedad() < 10) {

*return* *this*.getSueldoBasico() \* 0.04;

    }

*return* *this*.getSueldoBasico() \* 0.06;

  }

Aprendí lo básico de lo que vendría siendo la programación orientada a objetos aplicada al lenguaje de programación **Java**.