**Alumno:** Olivarez Alexis Emanuel  **DNI**: 44.542.230 **Laboratorio**: 2

**Profesores:** Medina – Airaldi **TP**: 5 **Fecha de Entrega**: 22/10/23

En la guía de ejercicios del TP5, implementamos la reutilización de clases predefinidas, conceptos de herencia, polimorfismo y volvimos a utilizar los contenedores, específicamente ArrayList.

A través de la herencia pude implementar uno de los principios de la programación orientada a objetos: la reutilización. Sucede lo mismo con las clases abstractas que nos permiten implementar la herencia y en algunos casos sin necesidad de tener un método escrito ya que al crear una clase abstracta podemos utilizar métodos abstractos solamente declarando el nombre, si es público o protegido y los parámetros que serán pasados (firma del método). Los métodos privados no pueden ser redefinidos, si se lo intenta en realidad se lo estará definiendo un nuevo método.

Si tenemos una clase abstracta, básicamente tenemos una clase “genérica”. A partir de ella crearemos las demás clases que respetarán y reutilizarán los métodos de la clase abstracta de la cual son hijos (subclase). Ésta reutilización se lleva a cabo mediante el uso de dos palabras clave: ***super*** que nos permite hacer referencia al método o a la clase padre (superclase). Ejemplo:

***public* Premium(*double* *p\_costo*, *int* *p\_colores*) {**

***super*(p\_costo);**

***this*.setColores(p\_colores);**

**}**

La otra palabra clave ***extends*** que se declara junto con el identificador de la clase para señalar que es la clase padre (superclase). Ejemplo:

***public* *class* Premium *extends* Etiqueta {}**

Clases y métodos abstractos:

Las clases abstractas no pueden ser instanciadas, una clase es abstracta cuanto contiene uno o más métodos abstractos, pero las clases abstractas también se pueden declarar sin métodos abstractos, una clase abstracta se declara con la palabra clave ***abstract***:

***public* *abstract* *class* Etiqueta {}**

Un método abstracto es aquel que no tiene implementación, y que por lo tanto nunca será ejecutado. Cada clase hija que hereda este método deberá implementarla independientemente. Ejemplo:

***protected* *abstract* *String* tipo();**

**// *Clase Premium***

**@*Override***

***protected* *String* tipo() {**

***return* "*Premium*";**

**}**

**// *Clase Comun***

**@*Override***

***protected* *String* tipo() {**

***return* "*Común*";**

**}**