Trabajo Practico 7 — Programación 2

Alumnos: Alliot Fermin
Carrera: Tecnicatura Universitaria en Programación
Materia: Programación 2
GitHub: https://github.com/Falliot00/UTN-TUPaD-P2/tree/main/7.%20Herencia%20y%20Polimorfismo
Fecha: 20/11/2025

Introducción

En este informe presento el desarrollo completo del Trabajo Práctico Nº 7, centrado en herencia, polimorfismo y clases abstractas en Java. A través de cuatro katas implementé los conceptos fundamentales de la Programación Orientada a Objetos, aplicados a jerarquías de clases, sobrescritura de métodos y uso de colecciones con referencias polimórficas.

Objetivos del Trabajo Práctico

Los objetivos principales del TP fueron profundizar en el uso de herencia, sobrescritura con `@Override`, uso de `super`, clases abstractas y polimorfismo en tiempo de ejecución. Cada kata aborda un escenario distinto que permite aplicar estos conceptos de manera progresiva y orientada a buenas prácticas.

Conceptos Teóricos Aplicados

• Herencia simple con `extends` para reutilización de atributos y métodos.

• Polimorfismo mediante referencias de tipo padre apuntando a objetos hijo.

• Clases abstractas y métodos abstractos como contratos obligatorios.

• Sobrescritura y late binding para seleccionar métodos en tiempo de ejecución.

• Upcasting implícito y uso de `instanceof` para verificación de tipos.

• Organización modular en paquetes, siguiendo buenas prácticas de diseño.

Kata 1 – Vehículos (Herencia Básica)

En la primera kata desarrollé una jerarquía simple: una clase base `Vehiculo` y una subclase `Auto`. Implementé atributos protegidos, sobrescritura del método `mostrarInfo()` y el uso de `super` en el constructor. Este ejercicio permitió reforzar la estructura básica de herencia y la reutilización de comportamiento en subclases.

Kata 2 – Figuras (Clases Abstractas)

En esta kata implementé una clase abstracta `Figura` con el método abstracto `calcularArea()`. Diseñé las subclases `Circulo` y `Rectangulo`, cada una con su propia fórmula de área. El uso de un array de tipo `Figura` permitió demostrar polimorfismo aplicando late binding.

Kata 3 – Empleados (Polimorfismo e instanceof)

Desarrollé un sistema de empleados aplicando polimorfismo en colecciones con `ArrayList<Empleado>`. Implementé dos tipos de empleados: `EmpleadoPlanta` y

`EmpleadoTemporal`, cada uno con su propia implementación de `calcularSueldo()`. También utilicé `instanceof` para clasificar empleados en tiempo de ejecución.

Kata 4 – Animales (Upcasting y Polimorfismo)

En la última kata implementé una jerarquía de animales donde cada subclase sobrescribe el método `hacerSonido()`. Mediante upcasting, almacené distintos animales dentro de una colección de tipo `Animal`, permitiendo invocar métodos polimórficos sin conocer el tipo real del objeto.

Conclusión

El trabajo práctico permitió afianzar el uso de herencia y polimorfismo aplicados a distintos escenarios. A través de las katas consolidé el uso correcto de clases abstractas, sobrescritura, colecciones polimórficas, upcasting e identificación de tipos. La estructura modular del proyecto contribuyó a una implementación clara, extensible y alineada con buenas prácticas de POO en Java.