PRÁCTICA 4: UNA CLASE PARA ORDENARLOS A TODOS

1. Introducción

En los años 60, Ole-Johan Dahl y Kristen Nygaard, hastiados de escribir (o troquelar tarjetas) con código muy parecido[1], decidieron solucionar dicho problema inventando la herencia y el polimorfismo de datos, creando el concepto de ***programación orientada a objetos*** y el primer lenguaje que la soportaba, denominado Simula 67. Dicho lenguaje, debido fundamentalmente a la ineficiencia introducida por la vinculación dinámica - y al igual que muchos otros lenguajes orientados a objetos, quedó principalmente restringido al ámbito académico. En la década de los años 80 las mejoras en el hardware de las computadoras y la aparición del lenguaje C++, que implementaba una forma restringida de vinculación dinámica - que suponía un compromiso intermedio entre flexibilidad y eficiencia - favorecieron el tránsito de las tecnologías orientadas a objetos de los ámbitos académicos a los industriales.

La continua mejora del hardware hizo que la barrera inicial para la adopción de los lenguajes orientados a objetos, que era su ineficiencia, fuese siendo cada vez más pequeña, hasta el punto de ser hoy en día prácticamente irrelevante en la mayoría de las aplicaciones. Por otra parte, las ventajas que ofrecen las tecnologías orientadas a objetos con respecto a capacidad de adaptación, facilidad de mantenimiento y reutilización del software, hacen que dichas tecnologías sean muy adecuadas para las necesidades de hoy en día. Actualmente las empresas no tienen demasiados inconvenientes en invertir en hardware más potente, si a cambio su software es fácilmente adaptable y modificable ante las necesidades de un mercado cada día más global y con requisitos altamente volátiles.

De forma paralela a la orientación a objetos se fue desarrollando el concepto de ***programación genérica***. En programación genérica, ciertos elementos de una aplicación software, tales como el tipo de una variable, son parametrizables. Por tanto, los programas parametrizados dejan de ser programas como tales y se convierten en *plantillas que representan familias de programas*. Una vez instanciados los parámetros de una plantilla correctamente, la plantilla se convierte en un programa concreto donde los parámetros se sustituyen por elementos específicos.

Actualmente, la mayoría de los lenguajes modernos (e.g., Java y C#) combinan programación orientada a objetos y genérica, lo que permite escribir potentes programas que favorecen tanto la reutilización y la adaptabilidad mediante la reducción del número de redundancias dentro de la descomposición modular de una aplicación software.

[1] Igual que nos ocurría a nosotros en la Práctica 1 – Computando Bajo el frío

2. Objetivos

El objetivo de esta práctica es aplicar los conceptos de programación orientada a objetos y programación genérica al enunciado de la Práctica 1 – *Computando bajo el frío* para de reducir las redundancias que en ella existían. El objetivo de la práctica es que el alumno aprenda a manejar los conceptos propios de la orientación a objetos (herencia, polimorfismo, vinculación tardía o dinámica y métodos abstractos) y de la programación genérica con el objetivo de producir software más fácilmente adaptable y reutilizable.

3. Actividades

Reimplementar la Práctica 1 haciendo ahora uso de las ventajas proporcionadas por la programación orientada a objetos y la programación genérica.

El alumno deberá crear una clase *Sorter* que proporcione un método *ordenar* que sirva para ordenar vectores de objetos de cualquier clase, siempre y cuando dicha clase cumpla con una serie de restricciones a definir por el alumno. Además, el alumno deberá implementar un método *mínimo*, que devuelva el menor objeto de un vector de objetos de la misma clase que el método *ordenar*.

4. Criterios de Evaluación y Aclaraciones

La práctica se considerará realizada correctamente si con una sola implementación de los métodos *ordenar* y *minimo* se pueden ordenar vectores y obtener el mínimo elemento de un vector para un número infinito de clases, siempre y cuando dichas clases cumplan ciertas restricciones. Además, dicha clase deberá poder usarse sin tener que realizar *castings*.

La práctica se entregará a través de la plataforma *moodle* siguiendo las instrucciones en ella proporcionadas.

Pablo Sánchez Barreiro.