PATRONES SOFTWARE
LABORATORIO
CONCEPTOS BÁSICOS / REFACTORIZACIÓN

Práctica 2

En esta práctica aplicaremos los conceptos de refactorización de código y aplicaremos los patrones fundamentales de Delegation, Interface y Abstract Superclass.

*Concepto de **refactorización** de código:*

- *Básicamente una limpieza de código y su optimización.*
- *La refactorización no arregla errores ni incorpora funcionalidades.*
- *Altera la estructura interna del código sin cambiar su comportamiento externo.*

Enunciado:

Una universidad necesita una aplicación capaz de gestionar su personal. Existen dos tipos de personal: el personal docente e investigador (PDI) y el personal de administración y servicios (PAS).

Los PDI se identifican a través de un número y están caracterizados por pertenecer a un departamento y un área. Por ejemplo, el PDI 123 pertenece al Departamento de Ciencias de la Computación y está dentro del Área de Lenguajes y Sistemas Informáticos.

Los PAS se identifican a través de un número y están caracterizados por pertenecer a una sección y ocupar un cargo. Ejemplo: el PAS 456 pertenece a la sección de Biblioteca y ocupa el cargo de Adjunto a Dirección.

Dentro de la gestión de la Universidad las tareas que se llevan a cabo son las siguientes:

- Contratar PAS/PDI
- Jubilar PAS/PDI
- Modificar datos de un PAS/PDI
- Mostrar los datos de los PAS/PDI
- Generar la nómina de todo el personal

La interfaz que se ofrece en primera instancia en la aplicación será de tipo texto y a través de menús.

Partiendo del código contenido en el proyecto NB LabPat2, detectar los principales problemas que presenta desde el punto de vista de la reutilización y optimización de código y hacer su refactorización. Entre ellos se pide:

- Crear una clase abstracta Personal que sea padre de las clases PDI y PAS con el atributo identificador y que contenga el método abstracto generarNomina().
- Los ArrayList tendrán el tipo de objeto que almacenan.
- La única clase que puede presentar mensajes por pantalla es la de MenuUniversidad, el resto de clases tendrán que devolver cadenas.
- La interacción con el usuario realiza un correcto tratamiento de los errores y quede limitada a la clase MenuUniversidad.
- Aplicar el patrón fundamental de delegación e interface para realizar el cálculo de la nómina. Crearemos una interface llamada "IntNomina" con un método:
  public int calculaNomina(String tipo)
  y una clase Nomina que implementará la interface y calculará la cantidad de dinero que recibirá el personal dependiendo del tipo recibido en el parámetro del método, si se trata de un PDI recibirá una cantidad aleatoria entre 1000 y 2000 € y si es un PAS entre 1000 y 1500 €. Cuando se llame al método generarNomina() se delegará en la clase Nomina su cálculo.
  ¿Con el diseño que acabamos de aplicar como podrías incorporar una nómina especial para las pagas extra donde sumaríamos 1000 euros más a la anterior?