**PONTIFICIA UNIVERSIDAD JAVERIANA**

****

**ANÁLISIS Y DISEÑO DE SOFTWARE**

**PARCIAL 3**

**WORD EXPLICATIVO**

**Daniel Francisco Osorio Rojas**

**David Julián Cuadros Astro**

**Fabio Andrés Triana Rodríguez**

**23/11/2023**

**Explicación del Diseño**

Diagrama de Paquetes

El diseño del proyecto se organiza en paquetes que reflejan la separación de responsabilidades:

Clases:

Contiene clases relacionadas con la lógica de negocio, como empleados, asignaturas y la gestión de la nómina.

Controlador:

Contiene la clase ControladorPrincipal, que actúa como intermediario entre la vista y el modelo. Coordinando las acciones del usuario con la lógica de negocio del sistema.

Javafxvideo11:

Agrupa clases relacionadas con la interfaz de usuario y la lógica para interactuar con ella.

Contiene controladores de interfaz (InterfazController) y la configuración para la visualización de la interfaz de usuario (JavaFXVideo11).

Diagrama de Clases

Clases:

1. Asignatura: Almacena información sobre una asignatura, como su nombre y número de horas.
2. Empleado: Clase abstracta que define las propiedades básicas de un empleado, como nombre, identificación, dependencia, cargo, y número de SMLV pagados.
3. EmpleadoComun: Extiende la clase Empleado y representa a un empleado común.
4. ManejadorArchivos: Maneja operaciones de lectura y escritura de archivos para la gestión de la nómina.
5. Monitor: Extiende la clase Empleado y representa a un monitor, tiene una lista de asignaturas que puede manejar.
6. Nomina: Gestiona la información relacionada con la nómina, permite agregar diferentes tipos de empleados (comunes, profesores, monitores) y asignarles salarios. Puede calcular el salario de un empleado.
7. Parcial3ADS: Una clase con un método principal.
8. Profesor: Extiende la clase Empleado y representa a un profesor. Tiene un escalafón, valor por hora y una lista de asignaturas que puede manejar.

Controlador:

1. Interactúa con Nomina para operaciones de adición, cálculo de salario y generación de informes

javafxvideo11:

1. InterfazController: Controlador de la interfaz gráfica, maneja las interacciones del usuario con la interfaz y actúa como intermediario entre la interfaz y la clase Nomina, permite adicionar diferentes tipos de empleados (profesores, monitores, empleados comunes). Calcula el salario de un empleado, asigna una asignatura a un empleado, genera un archivo de nómina.
2. JavaFXVideo11: Clase principal que inicia la aplicación JavaFX y carga la interfaz desde el archivo FXML.

**Uso de Patrones GRASP**

Controlador:

Podría asociarse al Controlador (ControladorPrincipal), que se encarga de manejar la interacción entre la vista y el modelo.

Creador:

Se aplica este principio ya que la clase Nomina podría ser responsable de crear instancias de Empleado.

Experto: Las clases poseen la información que necesitan para realizar sus operaciones, como Empleado y sus subclases, que manejan sus propios datos.

Bajo acoplamiento:

Se aplica este patron al separar claramente la lógica de negocio (clases) de la interfaz de usuario (javaFXVideo11).

Alta cohesión:

Se aplica porque la clase Empleado se encarga solo de las operaciones relacionadas con los datos y la lógica específica de un empleado.

Alta Cohesión y Bajo Acoplamiento: Las clases tienen responsabilidades definidas y están relacionadas de manera cohesiva. Por ejemplo, Nomina tiene la responsabilidad de manejar la información de los empleados y sus salarios.

**Uso de Principios SOLID**

S - Principio de Responsabilidad Única:

La clase Empleado se encarga solo de la información y operaciones relacionadas con un empleado. Esto significa que la clase solo tiene una responsabilidad y no tiene más de un motivo para cambiar

O - Principio de Abierto/Cerrado:

Si se quiere agregar un nuevo tipo de empleado, se puede hacer sin modificar la clase Nomina. Se cumple el principio ya que está abierta para la extensión, pero cerrada para la modificación

L - Principio de Sustitución de Liskov:

Profesor y EmpleadoComun son subclases de Empleado, pueden ser utilizados donde se espera un objeto Empleado. Se cumple el principio que dice que los objetos de un programa deben ser reemplazables por instancias de sus subtipos sin afectar la corrección del programa.

I - Principio de Segregación de Interfaces:

Algunas clases necesitan ciertos métodos de una interfaz, esa interfaz se puede dividir en interfaces más pequeñas y específicas. Los clientes no deben verse obligados a depender de interfaces que no utilicen. Este principio sugiere dividir las interfaces en grupos más pequeños y específicos, en lugar de tener una interfaz grande y general.

D - Principio de Inversión de Dependencias:

Las clases del paquete javafxvideo11 dependen de interfaces o contratos (Nomina) en lugar de depender directamente de implementaciones concretas. Se cumple el principio ya que los módulos de alto nivel no deben depender de módulos de bajo nivel. Ambos deben depender de abstracciones. Además, las abstracciones no deben depender de detalles, sino que los detalles deben depender de las abstracciones. Este principio se logra mediante el uso de interfaces o clases abstractas para desacoplar componentes.