Resumen

Paquetes:

El proyecto se ha organizado en varios paquetes para mejorar la estructura y la mantenibilidad del código. A continuación, se detallan algunos de los paquetes utilizados:

com.alvarobajo.banco.controller: Contiene clases que actúan como controladores para gestionar las operaciones y acciones específicas del banco.

com.alvarobajo.banco.models: Alberga las clases de modelo, como CuentaBancaria y Persona, proporcionando una representación estructurada de los datos.

com.alvarobajo.banco.view: Incluye la clase principal Menu y otras vistas responsables de la interfaz de usuario, permitiendo una separación clara entre la lógica de control y la presentación.

Funcionalidades en Paquetes:

La organización por paquetes sigue una lógica funcional, agrupando clases relacionadas según sus propósitos específicos. Ejemplos notables de paquetes y sus funcionalidades incluyen:

Controladores de Cuentas: En el paquete com.alvarobajo.banco.controller.cuentas, se encuentran clases que manejan operaciones y consultas relacionadas con cuentas bancarias.

Operaciones Bancarias: El paquete com.alvarobajo.banco.controller.cuentas también incluye la clase OperarCuenta, que proporciona métodos para realizar operaciones bancarias como retiros, ingresos y consultas de saldo.

Vista Principal: El paquete com.alvarobajo.banco.view contiene la clase Menu, que actúa como la interfaz principal del usuario, presentando menús interactivos para diversas operaciones bancarias.

Separación de Responsabilidades:

Se ha buscado una clara separación de responsabilidades entre las clases y paquetes. Este enfoque se destaca en las siguientes áreas:

La clase OperarCuenta se encarga de las operaciones bancarias, abstrayendo la lógica específica de cada operación.

LecturaTeclado se utiliza para gestionar la entrada de datos del usuario, manteniendo la lógica de interacción de usuario en un lugar separado.

Menu actúa como la interfaz principal pero delega la lógica específica de las operaciones bancarias a otras clases, siguiendo principios de diseño como la modularidad y la reutilización del código.

Entrada/Salida:

La clase LecturaTeclado se utiliza para manejar la entrada de datos del usuario y mantener la separación de preocupaciones relacionadas con la interacción del usuario. Esta práctica sigue el principio de diseño de única responsabilidad.

Main Class:

La clase Menu actúa como el punto de entrada principal para la aplicación. En el método main, se inicializa el banco, se cargan cuentas de prueba y se presenta un menú interactivo al usuario. Este diseño facilita el control del flujo principal del programa.

MostrarDatos:

La clase MostrarCuenta proporciona métodos para mostrar información relacionada con las cuentas bancarias, incluyendo la capacidad de buscar cuentas y mostrar listados. Esto sigue el principio de encapsulamiento y abstracción al separar la lógica de presentación.

Representación Gráfica del Proyecto:

(No proporcionado en la descripción. Se puede incluir un diagrama de clases o de paquetes para ofrecer una representación visual del diseño del proyecto).

Observaciones Específicas:

En la clase CrearPersona, se han proporcionado métodos para crear instancias de la clase Persona con información validada, contribuyendo a la integridad de los datos.

La clase CrearCuenta ofrece métodos estáticos para crear instancias de cuentas de diferentes tipos, como ahorro, corriente personal y corriente de empresa. Estos métodos facilitan la creación de cuentas y siguen el principio de simplicidad y reutilización de código.

En OperarCuenta, se han implementado métodos para realizar operaciones bancarias específicas como retiros, consultas de saldo e ingresos. Cada método incluye validaciones para garantizar la consistencia y seguridad de las operaciones.

La clase Menu se encarga de la interacción del usuario y utiliza otras clases para realizar acciones específicas según la elección del usuario, siguiendo el principio de delegación.

Conclusiones:

En conclusión, el código del proyecto demuestra una organización y estructuración efectivas para mejorar la legibilidad, mantenimiento y extensibilidad del programa. La adopción de buenas prácticas de diseño, como la separación de responsabilidades y la cohesión de paquetes, ha contribuido a un diseño robusto y escalable.