Actividad #1

Se realiza refactorización de la clase propuesta como main tomando los siguientes accionables:

- Se aplica el principio de Separación de responsabilidades (Single Responsibility Principle - SRP) sobre la clase main que incluía toda la logica de negocio y toda la interacción con el usuario en un mismo método. A continuación, detallamos la implementación realizada y otras prácticas aplicadas en las mismas.

- IOHandler: Clase dedicada a toda la presentación con el usuario garantizando entonces que la capa de negocio (cálculos) este completamente separado de la interacción con usuarios.

  En esta clase podemos encontrar funciones pequeñas con una tarea especifica:

  - Visualización del menú
  - Selección de la opción deseada de acuerdo con la figura requerida
  - Obtención de valores y validación de tipos de datos.
  - Se valida entonces que los datos ingresados sean tipo numérico y adicionalmente se definen de tipo double es un tipo de datos acorde para realizar cálculos geométricos. Las áreas o perímetros podrán contener decimales.
  - Función para mostrar el resultado al usuario imprimiendo área y perímetro

- Shape: Interface para definición de firmas a ejecutar por todas las figuras para nuestro ejercicio se implementarán dos operaciones calcular el área, y calcular el perímetro. Con esto hacemos uso del principio Interface Segregation y Dependency Inversion

- ShapeFactory: Clase que opera como fábrica de figuras donde de acuerdo con la opción seleccionada por el usuario se crea el objeto correspondiente con los parámetros requeridos.

  El uso de la fábrica y el polimorfismo permite que el código sea fácilmente extensible. Si se desea agregar una nueva figura en el futuro, solo sería necesario modificar la

ShapeFactory para incluir una nueva opción en el switch, sin tocar el método handleShapeCalculations() o el main. Con esto garantizamos que estamos cerrados a modificación, pero abiertos a extensión Open/Closed (OCP).

- Main: Contiene dos métodos:
    - **main:** dedicada a realizar el control del sistema y realiza el delegado de la logica de negocio a otra función
    - **handleShapeCalculations** encargada de adminsitrar la lógica de cálculo de las figuras, mientras que la obtención de entradas (ioHandler.getValue()) y la presentación de resultados (ioHandler.showResult()) se delegan a la clase IOHandler.
    - **handleShapeCalculations** tiene la responsabilidad de **gestionar la lógica específica de los cálculos geométricos**:
        - **Crear las figuras**: Teniendo en cuenta la opción seleccionada por el usuario, recoge los valores necesarios como lados, radios, etc. Ademas usa la **fábrica de figuras** (ShapeFactory.createShape()) para crear la figura necesaria.
        - **Calcular el área y el perímetro**: Después de crear la figura, calcula su área y perímetro utilizando los métodos calculateArea() y calculatePerimeter().
        - **Muestra los resultados**: Utiliza el IOHandler para mostrar los resultados al usuario.
        - **Gestionar opciones inválidas**: Verifica que la opción seleccionada por el usuario sea válida y maneja los casos en los que no se puede crear una figura.

- La lógica de cálculo de áreas y perímetros estaba repetida en cada caso dentro del switch. Al moverla a clases específicas y métodos, eliminamos la duplicación y aplicamos el principio DRY (Don’t Repeat Yourself).

- La responsabilidad de calcular el área y el perímetro está encapsulada en las clases Shape correspondientes. Ahora cada figura maneja sus propios cálculos, en lugar de que toda la lógica esté en un solo lugar en el main. Esto sigue el principio de responsabilidad única y el de encapsulamiento.

- En general todas las clases y métodos consideraron la práctica de realizar funciones pequeñas con una pequeña cantidad de líneas y con pocos parámetros.