Refactor

Generales

1. La clase “FormaGeometrica” original posee propiedades y comportamiento; es decir, funciona como *entidad* y *negocio* a la vez, evidenciando la falta de división por capas.
    a. Cambios:
        i. Se dejó en ella sólo el comportamiento.
        ii. Se creó una nueva clase “FormaGeometrica” de sólo propiedades, que funciona como entidad, en el namespace “Entities”.
2. Se reemplazaron las constantes de las formas geométricas y los idiomas por enumeraciones, en el namespace “Entities”.
3. El código del método “Imprimir” posee un IF con un ELSE, evidenciando una mala práctica en el uso del IF para este caso.
    a. Cambios:
        i. Se lo reemplazó por un IF solo que devuelve lo requerido, quitando el resto del código del ELSE.
4. Se quitó el resto de los IF’s y se los reemplazó con polimorfismo.
5. El código del método “Imprimir” ya no posee textos hardcodeados.
6. El contenido del reporte se realiza a través de un template.
7. Se modificaron los tests para que no verifiquen contra texto hardcodeado.

Aclaración 1:
La clase “FormaGeometrica”, con su método “Imprimir”, evidencia múltiples responsabilidades. Por un lado, se encarga de calcular áreas y perímetros y, por el otro, de generar un reporte.
Lo ideal sería que sólo se encargara de los cálculos y que, por otro lado, una clase “Reporte” se hiciera cargo del mismo. Luego, el cliente de este software sería el responsable de llamar a los métodos de estas clases para obtener el reporte final.
Esto no se hizo. Se dejó como está para evitar conflictos con los clientes actuales de este software.

Aclaración 2:
El código original está en castellano. Esto es una mala práctica.
El código que se agregó está en inglés, pero se mantuvieron algunas cosas en castellano para una mejor compresión por parte de los desarrolladores de este software.

Polimorfismo para los cálculos de las formas geométricas

El namespace “Business” posee las siguientes clases e interfaces:

- IFormaGeometrica: Interfaz que deben implementar todas las formas geométricas.
- Cuadrado, Circulo, TrianguloEquilatero: Clases que representan a cada una de las formas geométricas, encargadas de calcular sus áreas y perímetros.
- Factory: Clase encargada de manejar el polimorfismo.

Comportamiento

Cuando se recibe una colección de objetos “FormaGeometrica”, a través de su propiedad “ShapeType” se lleva a cabo este polimorfismo para obtener una instancia de una de las clases polimórficas para llevar a cabo los cálculos del área y del perímetro.

Polimorfismo para el idioma

El namespace “Texts” posee las siguientes clases e interfaces:

- De idioma:
  - ILanguage: Interfaz que deben implementar todas las clases de idioma.
  - Castellano, Ingles: Clases que representan a cada uno de los idiomas, encargadas de devolver los textos requeridos.
  - LanguageFactory: Clase encargada de manejar el polimorfismo.
- De Texto:
  - Text: Clase encargada del polimorfismo antes mencionado, que expone los textos ya traducidos al idioma requerido. De esta manera, la clase del reporte sólo utiliza una instancia de ésta para obtener sus textos, sin tener que poseer conocimiento del polimorfismo que hay detrás.
  - Shape: Dado el caso particular en el que existen textos tanto en singular como en plural en función de cierta lógica, esta clase es la encargada de traerlos ya traducidos. Es utilizada por la clase del reporte.
  - ReportTemplate: Establece el formato de texto que tendrá el contenido del reporte.

Clase agregada: Report

Es la encargada de generar el reporte, quitándole esta responsabilidad a la clase “FormaGeometrica”.

Aclaración:
Lo ideal sería tener un archivo de template, en lugar de generarlo con un StringBuilder.

Tests

Aquí también vemos malas prácticas en los tests originales:

- Sus códigos no se encuentran agrupados en “Arrange”, “Act” y “Assert”.
- Se verifican textos hardcodeados.
- Fueron desarrollados por idioma, lo que complica su escalado.
- Fueron desarrollados por figuras geométricas, lo que complica su escalado.

Refactor

Se dejaron los tests originales, sólo como para mostrar las buenas prácticas respecto de los dos primeros ítems de la lista anterior.

Luego, se agregaron sólo dos que contemplan todos los casos a testear.

El primero: EmptyList
- Prueba el caso del envío de una lista vacía en todos los idiomas.

El segundo: SeveralReports
- A través de "_getCases", prueba diferentes casos en todos los idiomas.

Casos colocados en este último método:
- 1 Cuadrado
- Muchos Cuadrados
- 1 Círculo
- Muchos Círculos
- Muchas formas

Estos cinco se repiten para ambos idiomas.
Estos casos fueron agregados sólo a modo de ejemplo de escalado. Obviamente, faltarían agregar: 1 Triángulo Equilátero, Muchos Triángulos Equiláteros, etc.

Manejo de excepciones

No se han implementado manejo de excepciones.

Se puede observar en el código original, en los métodos de cálculo, que los switch-case's contemplan la elevación de una excepción.

Considero que es una mala práctica hacerlo de esta manera. Lo ideal es crear excepciones personalizadas (clases que heredan de Exception), manejadas por una clase "Validator", que verifique más condiciones de error en función de los parámetros recibidos (valores en "0", idiomas no reconocidos, etc.).

No lo hice por falta de tiempo.

Escalado

Implementación de nuevo idioma

1. Crear una nueva clase en el namespace "Text" que implemente "ILanguage".
2. Crear un nuevo valor en la enumeración "LanguageType" (namespace "Entities") para dicho idioma.
3. Agregar un nuevo par Key-Value al diccionario de la clase "LanguageFactory".
4. En los Tests:
    a. Agregar ese mismo par Key-Value al diccionario que se encuentra en el método "OneTimeSetup".
    b. Agregar nuevo TestCase al Test "EmptyList" con dicho idioma.
    c. Agregar nuevo set de casos en "_getCases".

Ver ejemplo de la clase "Italiano".

Implementación de nueva forma geométrica

1. Crear una nueva clase en el namespace "Business" que implemente "IFormaGeometrica".
2. Codear los métodos de cálculo para dicha forma.
    a. Si fuera necesario, agregar nuevas propiedades a la clase "FormaGeometrica" del namespace "Entities" con nuevos constructores.
3. Crear un nuevo valor en la enumeración "ShapeType" (namespace "Entities") para dicha forma.
4. Agregar un nuevo par Key-Value al diccionario de la clase "Factory".
5. En el namespace "Texts":
    a. Interfaz ILanguage: Agregar nuevas propiedades para el nombre de esta forma, tanto en singular como en plural.
    b. Implementar estas propiedades con sus textos en todas las clases de idioma.
    c. Clase Text: Agregar estas mismas propiedades.
    d. Clase Shape: Agregar nuevos pares Key-Value para los diccionarios "_singulars" y "_plurals".
6. En los Tests: Agregar nuevo set de casos para esta nueva forma.

Nota:
Si bien los textos de los nombres de las formas no son utilizados por la clase "Text" (se puede observar '0 References' sobre ellos), igual conviene colocarlos por las dudas de que en el futuro se requiera utilizarlos.

Ver ejemplo de "Rectángulo" y "Trapecio".