Trabajo Práctico Especial Programación Orientada a Objetos (Recursada) 2019

04/11/2019

Integrantes:
Valerioti, Mauro Lorenzo
D'Ambrosio, David Leandro

Ayudante designado:
Rodriguez, Guillermo

Visión general

Este trabajo corresponde a la materia de Programación Orientada a Objetos, correspondiente al 3er año de la carrera de Ingeniería de Sistemas de la Facultad de Cs.Exactas, en Unicen. La materia hace enfoque al paradigma de orientado a objetos. Se explican conceptos fundamentales como Clases, Objetos, Herencia, Polimorfismo, entre otros más. El lenguaje propuesto a la hora programar problemas pensados a nivel de objetos es JAVA.

Objetivos

1. Se pretende que el alumno frente a un problema sepa identificar los objetos que intervienen, su estado y comportamiento, y la relación que existen entre ellos.
2. Es importante tener en cuenta la delegación de responsabilidades, es decir, no caer en extremos de que una clase provea toda la funcionalidad del problema, o para cada método crear un clase, si no es necesario.
3. También se busca que la solución orientada a objetos sea legible, extensible y flexible, ya que lo más importante en el ciclo de vida de un algoritmo es la mantención del mismo en el tiempo.
4. No caer en problemas del tipo Clase vs. Instancia, romper el encapsulamiento, duplicación de código, etc.

Especificaciones

Servidor de páginas Web

Un servidor de páginas web, además de permitir el almacenamiento y recuperación de páginas, brinda servicios de transformación y adaptación de las mismas para adecuarlas a los estilos que requieren sus clientes. Para esto el servidor permite definir estilos de transformación que luego serán aplicados a las páginas cuando éstas sean solicitadas. Un cliente solicita una página diciendo además qué estilo prefiere.

Una página se compone de texto (múltiples párrafos), imágenes y vínculos a otras páginas, estructurados en cualquier orden según cada página en particular.

Las transformaciones que se pueden aplicar a las páginas son las siguientes:

- ***Agregar al final de la página el tamaño de la misma en bytes.(cantidad de caracteres párrafo + bytes imágenes + tamaño del documento referenciado por el vínculo)***
- ***Agregar al final de la página la cantidad de componentes. (cantidad de párrafos + cantidad de imágenes + cantidad de vínculos)***
- ***Agregar fecha actual.***
- ***Agregar un encabezado al comienzo de la página.***
- ***Agregar un pie de página.***
- ***Reducir el tamaño del documento (reducir imágenes y tamaño de la fuente en cierta escala - cada párrafo puede definir su propio tamaño de fuente).***
- ***Aumentar el tamaño del documento (aumentar imágenes y tamaño de la fuente en cierta escala - cada párrafo puede definir su propio tamaño de fuente).***

Un estilo define qué transformaciones aplicar y en qué orden, por ejemplo:

- estilo ¨defecto ¨aplica transformación 1,3 y 6.
- estilo ¨grande ¨ aplica 3 y 7, etc...

Adicionalmente debe poder agregarse nuevas transformaciones de manera sencilla.

También se debe tener en cuenta que no se puede modificar la página original sino que se debe devolver una copia de la página transformada.

Nota: el agregado de elementos (pie de página, fecha, etc.) se realiza mediante el agregado de párrafos a la página en la posición correspondiente.

Abstracción del problema

Desglosando el problema, se tendrá un conjunto de **Pagina**(s), donde cada una tiene un conjunto de **Parrafo**(s) , **Imagen**(es) y **Vinculo**(s). Cada uno de estos **Elemento**(s) debe tener la capacidad de resolver ciertos *comportamientos* usando sus *estados* para que la página luego se transforme.

En el siguiente cuadro se muestra el estado y comportamiento de los Objetos que intervienen, y su relación entre ellos:

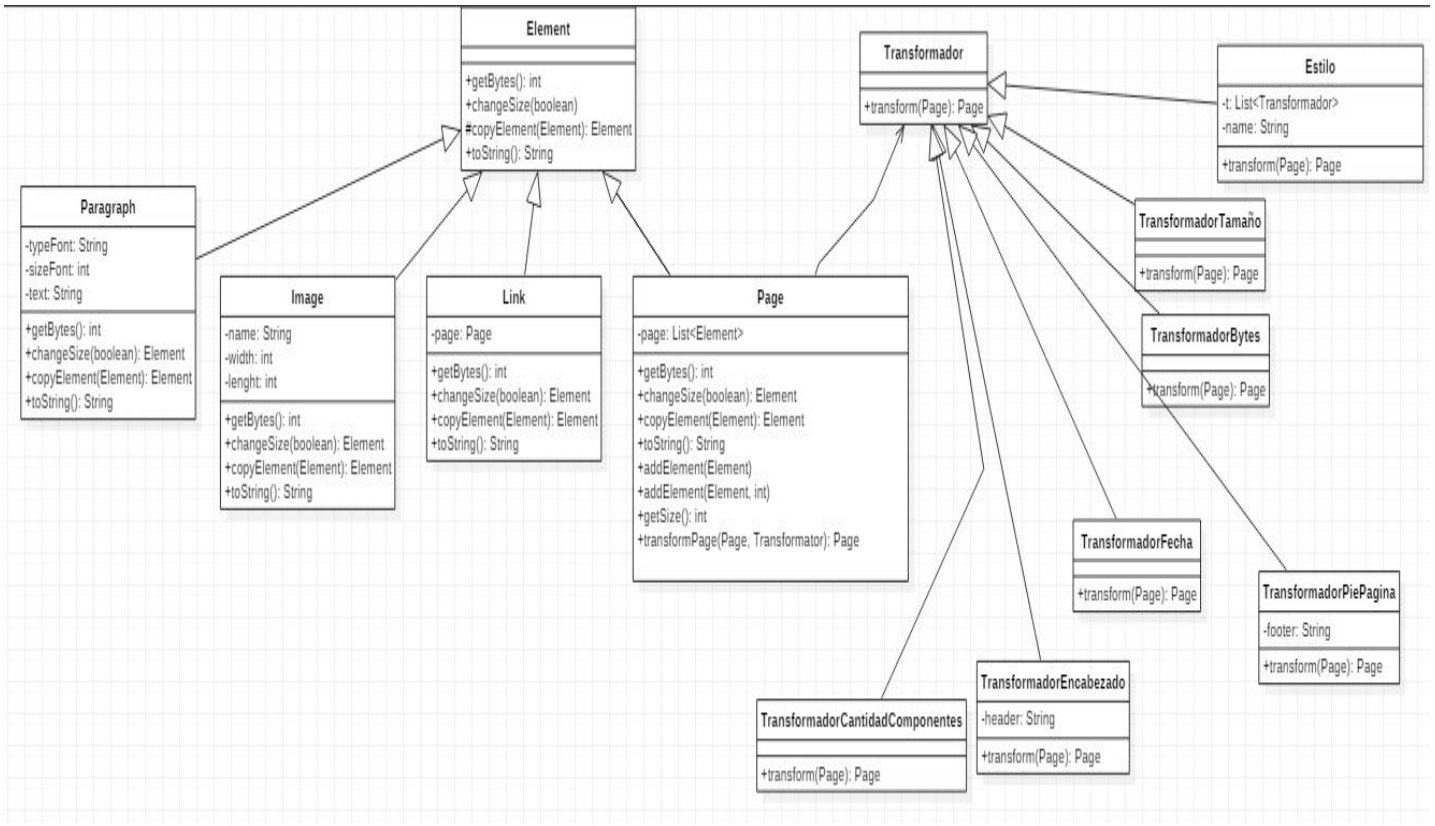
Objeto	Estado	Comportamiento	Relación
Parrafo	• Tamaño fuente • Tipo de fuente • Contenido	• Cantidad de Bytes • Aumentar/reducir tamaño	Ninguna
Imagen	• Nombre • Largo • Alto	• Cantidad de Bytes • Aumentar/reducir tamaño.	Ninguna
Vinculo	• Pagina asociada	Comportamiento de la página asociada	Tiene una página
Página	-	• Añadir un Párrafo/Imagen/Vinculo • Transformar Página • Cantidad de Bytes • Aumentar/reducir tamaño	Conjunto de párrafos, imágenes y vínculos

Viendo el cuadro, existe comportamiento en común entre los objetos, el de obtener la cantidad de bytes y modificar el tamaño. Entonces, puede abstraerse ese comportamiento, y crear un Objeto llamado **Elemento,** y que **Parrafo**, **Imagen**, **Vinculo** y **Pagina** *extiendan* de este nuevo Objeto. Tanto el ***getBytes()*** como ***changeSize()*** serán métodos que el elemento no sabrá cómo realizarlo, por lo que debe delegarlo a sus hijos, y estos métodos serán abstractos, por lo tanto Elemento es una Clase Abstracta.

Como el vínculo tiene asociada una página, y tal tiene un conjunto de Elementos, una página puede tener contenida otra , y esta incluso otras. Esto es un Patrón de Diseño llamado Composite. Particularmente se tendrá la misma idea con el **Transformador**, que será una clase Abstracta, con un método abstracto de transformar, y cada hijo de la clase padre, implementará tal método(transformación) de acuerdo a la especificación. Aquí también se hace uso de del Composite, teniendo una clase **Estilo** que hereda de Transformador, y esta tendrá una lista de Transformadores, que se aplicarán sucesivamente a una página.

Diagrama de Clases UML

Así quedaron conformadas las clases:



Implementación:

Elemento:

Para llevar a cabo la implementación del problema, se tuvo en cuenta lo siguiente:

En **Parrafo:**

- getBytes() : retorna el largo del String paragraph.
- changeSize(boolean) : incrementa/decrementa en 1 el tamaño de la fuente dependiendo el valor booleano.

En **Imagen**:

- getBytes(): retorna el producto entre el largo y el alto de la imagen.
- changeSize(boolean): incrementa/decrementa en 100 el largo y el alto de la imagen

En **Vinculo**:

- getBytes(): retorna el getBytes() asociado a la página
- changeSize(boolean): retorna el changeSize(boolean) asociado a la página

En **Pagina**:

- getBytes(): retorna el getBytes() para cada elemento de la página
- changeSize(boolean): retorna el changeSize(boolean) asociado a la página

Además existen otros métodos abstractos como el *toString()*, para mostrar el elemento y el copyElement(Element) para hacer una copia en profundidad de un elemento, necesario para cuando se aplica la transformación a la página.

Transformador

- **TransformadorBytes:** transform(Page) agrega al final de la página la cantidad de bytes de la misma
- **TransformadorCantComponentes** : transform(Page) agrega al final de la página la cantidad de componentes de la misma (solo a nivel superficial).
- **TransformadorFecha** : transform(Page) agrega al principio de la página la fecha actual.
- **TransformadorEncabezado**: transform(Page) agrega al principio de la página un encabezado provisto por la clase.
- **TransformadorPiePagina**: transform(Page) agrega al final de la página un pié de página provisto por la clase.
- **TransformadorCambiarTamaño**: transform(Page) modifica el tamaño de cada elemento de la página dependiendo de la variable booleana provisto por la clase.
- **Estilo**: transform(Page) aplica todas las transformaciones provistas por la clase a la página.

Aclaración: para agregar datos como int,String y Date se creó un Párrafo donde la variable paragraph tomaba el valor de estos, ya que una Página solamente soporta Objetos de tipo Elemento.

Enlace entre Pagina y Transformador

La clase Pagina tiene un método llamado transformPage(Page,Transformator) donde al objeto instanciado de tipo página se le aplique la transformación. Por el **binding dinámico** y **polimorfismo** no importa que instancia de la transformación le llega, simplemente la aplica y devuelve una copia de la página a la que se transformó.