Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería en Informática
Paradigmas de Programación

INFORME DE LABORATORIO 3:

GIMP MEDIANTE JAVA

- **Autor:** Gabriel Ojeda.
- **Profesor:** Gonzalo Martínez.
- **Fecha:** 03 de Diciembre del 2022.

Contenido

1. INTRODUCCIÓN

En el presente informe se darán a conocer los elementos para poder adentrarse al problema que se presenta, el cual consiste en replicar las famosas aplicaciones de edición de imágenes y videos GIMP o Adobe Photoshop en lenguaje de programación JAVA, basándose en la particularidad de este, el paradigma Orientado a Objeto, de esta manera se mostrara cómo se abordó el tema, el enfoque que se le dio y como se llegó a una solución que permitió realizar un programa con algunos de los requisitos básicos completados en su totalidad, pero con una interfaz gráfica interactiva para el usuario, además se dará una conclusión donde se hará una autocrítica y como mejorar para el próximo laboratorio.

DESCRIPCIÓN DEL PROBLEMA

Se pide desarrollar una simulación de un software de edición o manipulación de imágenes digitales, el cual le permite a un usuario realizar distintas operaciones sobre éstas. Tal y como son los softwares GIMP y Adobe Photoshop. Existen distintos tipos de operaciones tales como rotar una imagen, invertirla, rotarla, retocarla, transformarla y redimensionarla, entre otras.

El proyecto se concentrará en trabajar en imágenes RGBD o RGB-D, esto es, imágenes que además de tener información en el espacio de colores (R)ed, (G)reen, (B)lue, contiene información de la profundidad (D)epth en un espacio tridimensional. Al incorporar la dimensión D (profundidad) capturada a través de una cámara especializada, sería posible saber más sobre los detalles del rostro, proyección de la nariz, sombrero, distancia del espejo en la parte posterior, etc. Incluso sería posible construir una representación tridimensional del rostro.

Otro elemento a considerar es que esta versión del software operará con representaciones sencillas de imágenes, En particular, se consideran bitmaps-d (para imágenes donde cada pixel o pixbit puede tomar el valor 0 o 1), pixmap-d (para imágenes donde cada pixel o pixrgb es una combinación de los valores para los canales R, G y B) y hexmaps-d (donde cada pixel o pixhex expresa la información del color del pixel a través de un valor único hexadecimal de 6 valores).

Se debe implementar el software en el paradigma Orientado a Objeto, específicamente en el lenguaje de programación JAVA mediante el editor IntelliJ.

DESCRIPCIÓN DEL PARADIGMA

El paradigma orientado a objetos, como su nombre lo dice, está basado en el concepto de “objetos”, que pueden tener características (atributos) y comportamientos (métodos), los cuales son una abstracción de entidades de la realidad. Los programas crean objetos en memoria y estos interactúan entre sí.

En el paradigma existen objetos de distinto tipo, que se denominan clases, y entre cada clase, pueden variar sus características y comportamientos, a las que llamamos atributos y métodos respectivamente. Con lo visto en los laboratorios anteriores, se puede decir que una clase corresponde a una implementación de un TDA.

Las clases son la definición de los atributos y métodos para un cierto tipo de objeto, por otro lado, los objetos son representaciones activas en memoria de una clase durante la ejecución de un programa. De una clase, se puede crear múltiples objetos que pueden interactuar entre sí, como también, interactuar con objetos de otras clases.

Los atributos, corresponden al conjunto de variables que caracterizan a un objeto, estos atributos pueden ser tipos de datos primitivos, TDAs u otras clases.

Los métodos representan los comportamientos que exhibe un objeto, son equivalentes a funciones y procedimientos en el paradigma imperativo-procedural. Los métodos son comportamientos que puede realizar un objeto sobre sí mismo o sobre otros objetos. En otras palabras, a través de un método se pueden leer o escribir sus atributos, o interactuar con otros objetos. Para crear un objeto en base a una clase, existe un tipo especial de método llamado “constructor”. Un constructor nos permite crear un nuevo objeto, pudiendo indicar los valores iniciales de sus atributos.

Para diseñar una solución basada en el paradigma orientado a objetos existe el lenguaje unificado de modelado, el cual permite establecer relaciones entre los objetos y componentes que forman una aplicación del paradigma. Además, permite comunicar efectivamente un diseño POO entre un grupo de desarrollo. En efectos de este laboratorio, se utiliza el diagrama de clases para diseñar la solución.

En los paradigmas anteriores, el análisis y diseño se basaba en el ¿QUÉ? y no en el ¿CÓMO?, en cambio en el paradigma orientado a objetos se basa en ambos para la solución.

OBJETIVOS

El objetivo del proyecto se basa en aprender y utilizar los conceptos del paradigma orientado a objetos, como también el uso del lenguaje de programación Java. Con esto, al terminar el proyecto, tener un buen desempeño con el paradigma, para poder ocuparlo en futuros aspectos laborales.

Otro objetivo, es el desarrollo del software similar a GIMP o Adobe Photoshop, en el lenguaje JAVA, que tiene como propósito llevar a cabo los conocimientos vistos en clases y la correcta utilización del material, para lograr el objetivo principal.

Finalmente comprender en profundidad la lógica de programación y construir un software similar a GIMP.

2. DESARROLLO

ANÁLISIS DEL PROBLEMA

Sabemos que hay que desarrollar un software que simule la aplicación de edición de imágenes GIMP y desarrollar algunas de las funciones principales de este, como recibir imágenes en diferentes formatos, los cuales están especificados en pixbit-d, pixhex-d y pixrgb-d, rotar dichas imágenes, trasladar, comprimir, expandir, y modificaciones en general.

Para esto se requiere generar funciones o estructuras que puedan realizar todas estas labores.

El objeto a estudiar en primera instancia es una imagen, la cual puede tener 3 diferentes formatos y todos deben ser soportados por el sistema. Teniendo en cuenta esto podemos decir que:

Una imagen puede ser solo de un tipo de pixels, sin embargo, los 3 tipos de pixels tienen valores comunes, como las coordenadas x e y, y la profundidad d, por lo tanto, para hacer mas optimo el programa vamos a tener selectores y modificares transversales para estos elementos y otros particulares para cada tipo de pixel.

Por otro lado, como lo mencionan en el enunciado una imagen es tridimensional, tiene un ancho, un largo y además una lista que contiene los datos del tipo de pixel, por lo que cada pixel va a ser un TDA aparte pero que en su conjunto pueden formar el TDA imagen y así abarcar todos los tipos.

Para los predicados de pertenencia y modificación para una imagen, pueden variar dependiendo del tipo de pixel, por lo que un solo predicado o regla principal, tendría que ser capaz de validar una acción por igual para los 3 tipos de imágenes, por ende, para cada regla principal de la imagen se requiere de funciones particulares que se ejecutaran dependiendo de cada condicional del pixels.

DISEÑO DE LA SOLUCIÓN

Primero se creó una Carpeta “tdas” el cual contiene las clases asociado a la imagen, con su interface, y las clases para cada uno de los pixeles.

Cada clase tiene sus correspondientes constructores, selectores, modificadores, etc.

En la clase imagen a parte de lo mencionado anteriormente, también se encuentran los métodos solicitados en los requerimientos funcionales de este laboratorio.

Por otra parte, tenemos el main, que solo realiza el llamado a la ventana principal, el cual es una clase que cumple una especie de menú con interfaz gráfica, esta tiene botones que realizan acciones y llamados a las otras clases, las cuales son diferentes ventanas.

Cada ventan emergente es una clase distinta, que tiene sus propios botones con sus respectivos atributos que realizan llamados a otras ventanas. De esta manera vamos interactuando

gráficamente con el programa, pasando de ventana en ventana, ya sea creando imagen, asignado valores, creando pixeles, imprimiendo la imagen, etc.

ASPECTOS DE IMPLEMENTACIÓN

COMPILADOR

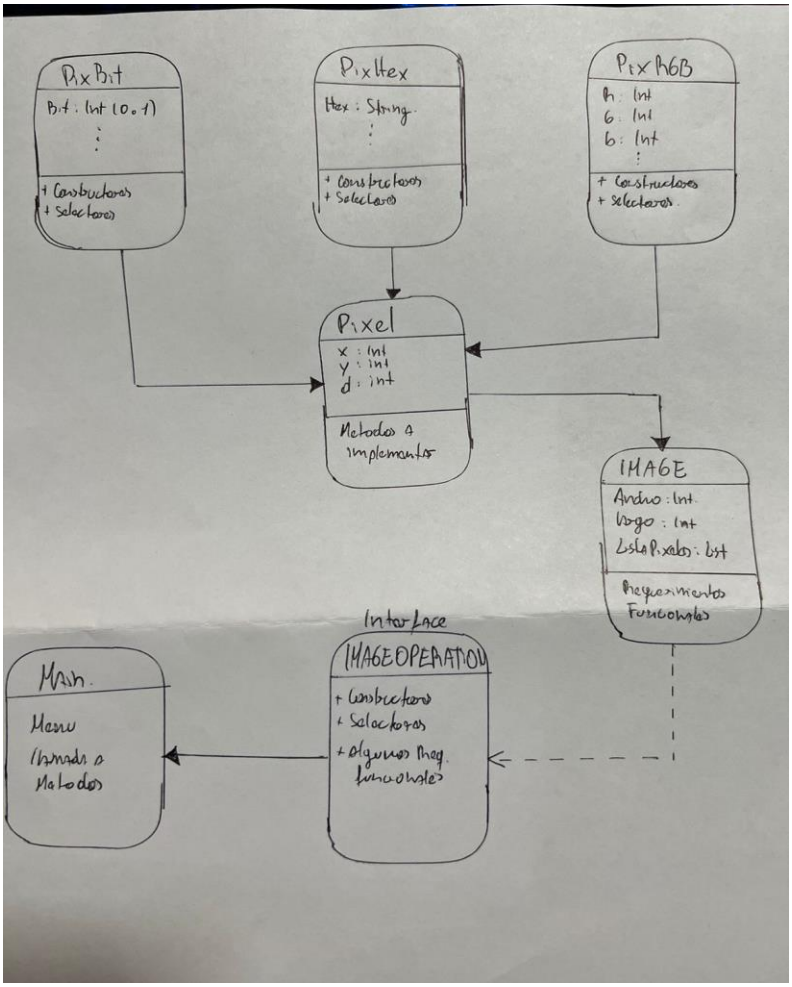
Para aspectos de este programa, se requiere usar el lenguaje Java y se decide usar el compilador JDK versión 11, en sistema operativo Windows, ya que es una herramienta bastante simple a la hora de compilar.

Se utilizo el editor IDLE IntelliJ en su versión pro, gracias a la licencia otorgado por el correo usach. Además de utilizar diferentes paquetes para cumplir con el requerimiento de la interfaz grafica del programa.

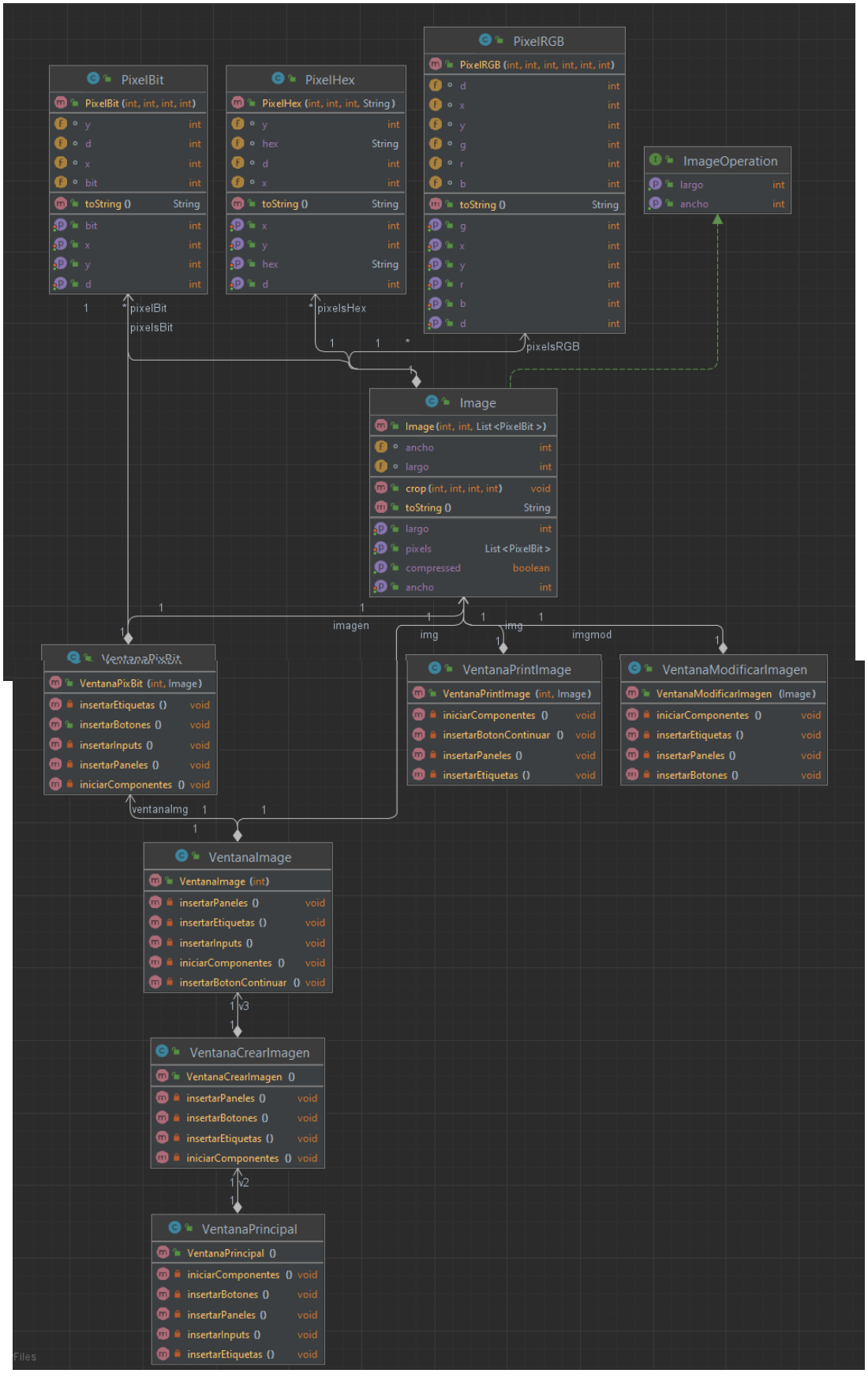
Para efectos de este laboratorio la implementación se basa en el trabajo de objetos.

DIAGRAMA DE CLASES

ANTES DE LA IMPLEMENTACIÓN



DESPUÉS DE LA IMPLEMENTACIÓN



ESTRUCTURA DE CODIGO

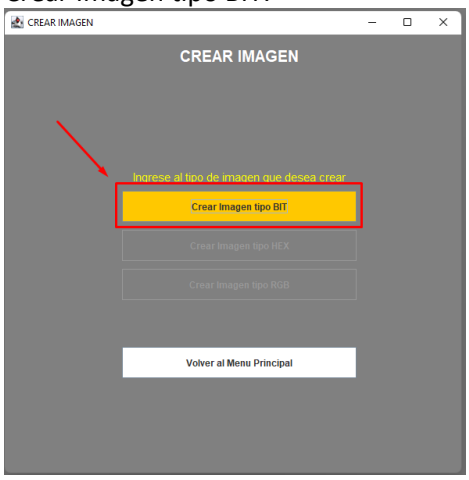
Se sabe que Java es un lenguaje que se divide por clases, por lo que el código debe estar estructurado de manera en que los atributos y métodos estén de manera funcional para un perfecto funcionamiento del software.

INSTRUCCIONES DE USO

EJEMPLO DE USO

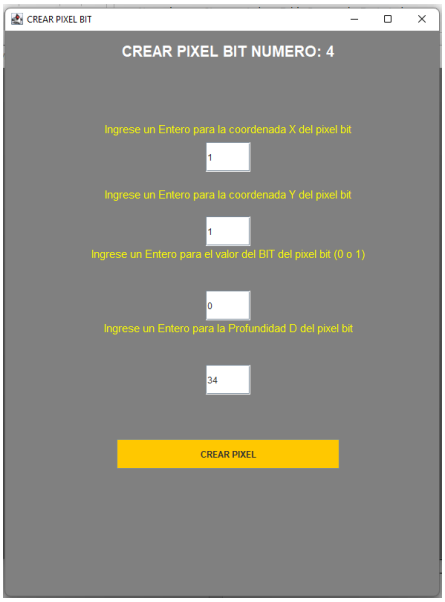
1. Ejecutar el programa y crear imagen.

2. Crear Imagen tipo BIT:



3. Ingresar Largo y Ancho de la imagen a crear.

4. Crear Los Pixeles



5. Interactuar con la imagen.

6. 

RESULTADO ESPERADO

Se espera que el programa sea 100% Orientado a Objeto para cumplir con el paradigma estudiado y siguiendo la estructura asignada, dejando una estructura acabada de los TDA que se deben implementar, con el formato visto anteriormente. Tras cumplir con las estructuras, se espera que se haga un correcto trabajo con predicados, hechos, reglas, etc.

POSIBLES ERRORES

Los posibles errores que se pueden producir en el programa son los siguientes:

- Si el usuario inicializa la imagen sin respetar el formato asignado, el programa no funciona.
- Si el usuario no respeta el formato asignado a cada parámetro de entrada en los comandos, el programa no funciona.

Al probar el programa, solamente se encuentra ese posible error, por lo demás ha funcionado correctamente con todas las funcionalidades que se implementaron.

RESULTADOS Y AUTOEVALUACIÓN

RESULTADOS OBTENIDOS

Los resultados obtenidos, fueron los esperados, ya que se implementó de buena forma la simulación, excepto por el hecho de que no se alcanzaron a crear la totalidad de las funciones solicitadas.

Al probar el programa con todos los ejemplos que se pueden hacer, sólo dejó de funcionar en lo que se menciona anteriormente, sin contar ese punto y respecto a las pruebas que se hicieron, se puede decir que el programa es completamente funcional y respeta el paradigma solicitado.

AUTOEVALUACIÓN

Se evalúan las funcionalidades, de la siguiente forma:

0: No realizado – 0.25: Funciona 25% de las veces – 0.5: Funciona 50% de las veces – 0.75: Funciona 75% de las veces – 1: Funciona 100% de las veces.

CLASES	EVALUACIÓN
TDA PIXBIT	1
TDA PIXHEX	1
TDAPIXRGB	1
TDA IMAGE	0.75
MAIN	0.5

VentanaCrearImagen	1
VentanaImage	1
VentanaModificarImagen	0.25
VentanaPixbit	1
VentanaPrincipal	1
VentanaPrintImagen	1

Todas las clases funcionan en su totalidad, constructores, selectores, modificadores, etc. Sin embargo la clase image no posee todos los requerimientos funcionales, por lo que en la clase VentanaModificarImagen tampoco esta completa ya que no posee todos los botones para ejecutar todas las acciones solicitadas.

3. CONCLUSIÓN

Se puede concluir que el resultado de este tercer laboratorio fue el esperado ya que, a pesar de no cumplir los requerimientos funcionales, si se logró aumentar considerablemente el nivel de desarrollo en el paradigma orientado a objeto a través de Java, además de aprender el lenguaje, también se logró satisfactoriamente crear una interfaz grafica utilizando el paquete JFrame de java, creando ventanas, etiquitas, inputs, botones y realizar acciones con estos, etc.

Los problemas que se presentaron fueron que en un principio se tenía una idea de cómo realizar este laboratorio y esta fue cambiando demasiado con el paso del tiempo, hasta llegar a la última idea que fue la que se dejó (Luego de reestructurar varias veces el código y los archivos) y gracias a esta se pudo realizar al 100% el laboratorio (en relación con lo funcional), pero no el 100% de las funcionalidades solicitadas.

Luego de ordenar todas las ideas y empezar a programar, las complicaciones fueron pocas, errores que se pasaban por detalles y reparaciones mínimas en implementaciones de cada Clase, cabe destacar que hubo muchos errores, pero ninguno que haya tomado un tiempo significativo a la hora de solucionarlo.

Finalmente, en comparación al laboratorio 1 y 2, en este se obtuvo un mayor aprendizaje y compresión del paradigma, por lo mismo se logró abarcar mayores cantidades de requisitos solicitados, por lo que si hubo una mejora significativa en comparación al anterior.

4. REFERENCIAS

Gonzalo Martínez, Víctor Flores y Roberto González. (2022). Programación funcional, Paradigmas de programación. Sitio web: CAMPUS VIRTUAL (usach.cl)