Informe Laboratorio 3

Paradigma Orientado a Objetos – Lenguaje Java

Fabián Alejandro Lizama González

Departamento de Ingeniería Informática

Facultad de Ingeniería, Universidad de Santiago de Chile

Paradigmas de Programación

Profesor Roberto González Ibañez

3 de diciembre de 2022

Introducción

En el informe se presenta el concepto de Paradigma Orientado a Objetos (concepto detallado en el cuerpo del documento) aplicado al desarrollo de un software de manipulación de imágenes bajo un enfoque simplificado, bajo el lenguaje de programación “Java”.

Se desarrollará este software presentando el problema, analizándolo y comentando aspectos del proceso de solución e implementación de éste junto a algunas instrucciones y ejemplos de uso para terminar con el apartado de resultados y conclusiones.

Descripción del problema

Se solicita diseñar un software de manipulación de imágenes del tipo RGBD, las cuales tienen la particularidad de adicionalmente a los canales de colores (red, green, blue) se tiene un canal de profundidad para representar imágenes en tres dimensiones, con el objetivo de poder manipular a gusto este tipo de archivos mediante distintas funciones. Para esto se requiere implementar la solución bajo el paradigma funcional e implementar una representación basada en “Tipos de Datos Abstractos” (TDAs).

Descripción del Paradigma

El Paradigma Orientado a Objetos mencionado en la introducción pertenece a la familia de los paradigmas declarativos y consiste en trabajar principalmente con objetos que interactúan entre sí, los cuales poseen atributos y métodos.

Algunos aspectos de este paradigma a tratar en la solución son los siguientes:

Atributos: Características o cualidades de un objeto.

Métodos: Comportamientos o acciones que realiza un objeto.

Clase: Plantilla de un objeto que norma los atributos y métodos que esté tendrá.

Relaciones entre objetos, como se mencionó anteriormente, los objetos se relacionan entre sí de distintas maneras:

Composición: Relación entre dos clases donde el tiempo de vida de un objeto depende del objeto en el que está incluido.

Agregación: Cuando la vida de un objeto incluido en otro no depende de este.

Asociación: Relación débil que sucede cuando un objeto colabora con otro y sus tiempos de vida son independientes entre sí.

Dependencia: Una clase depende de la otra, pero la que depende no conoce la existencia de la otra.

Herencia: Cuando una clase hereda los atributos y métodos de otra, además puede tener propios.

Clase abstracta: Una clase que no puede ser instanciada, sirve como plantilla para definir herencias.

Interfaces: Clases que tampoco pueden ser instanciadas, pero solo marcan pauta de los métodos que tendrán las clases que hereden de esta, además en una interfaz, los métodos solo poseen su encabezado.

Análisis del Problema

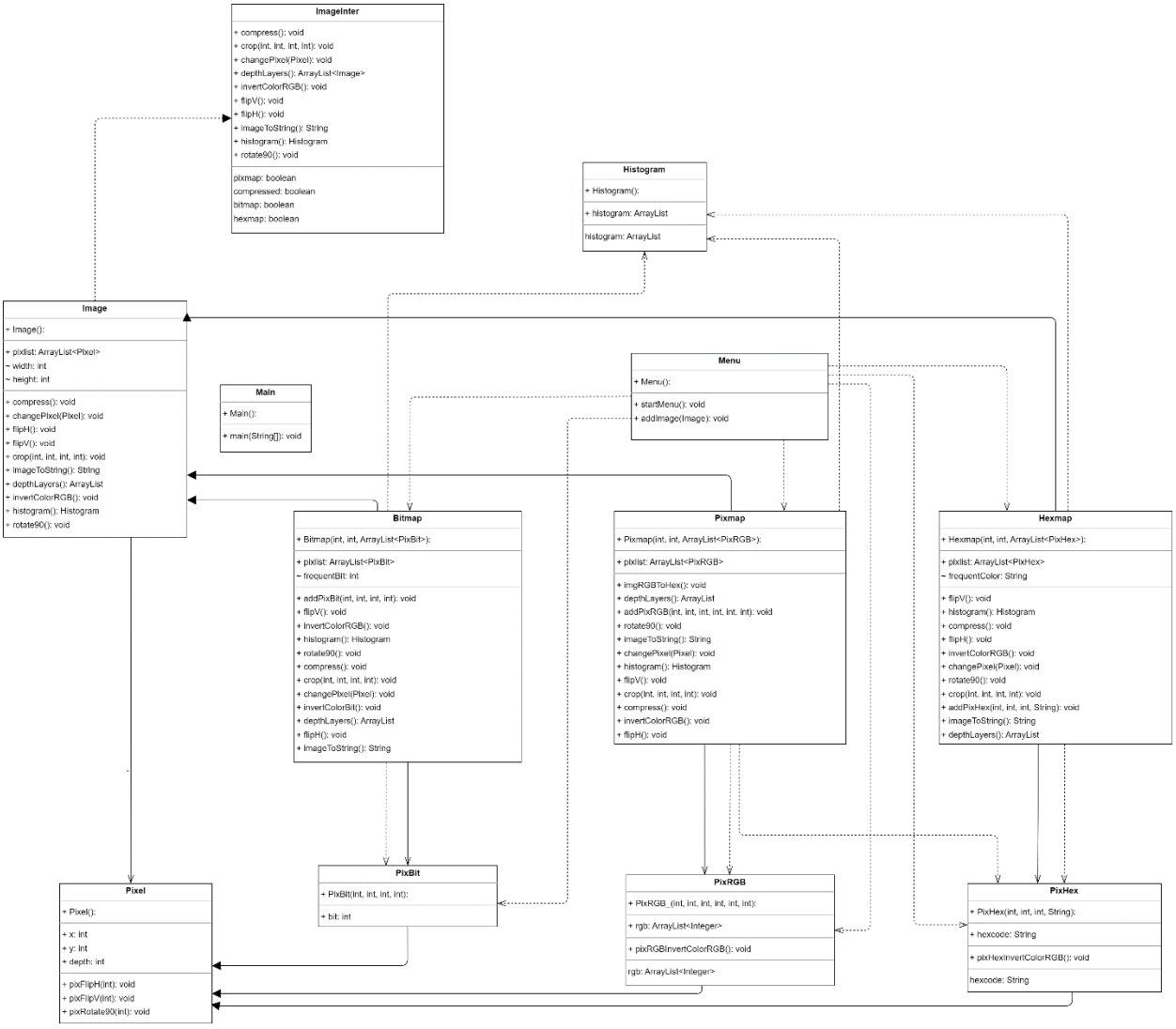
La principal complejidad del problema radica en el realizar el software solo bajo el paradigma orientado a objetos, nuevamente evitando la programación imperativa y abordando un “cambio de paradigma” al haber desarrollado esta solución anteriormente bajo el paradigma funcional y bajo el paradigma lógico. Se implementarán abstracciones apropiadas para el problema con los objetos apropiados, junto a sus métodos asociados de construcción, selección, modificación y otros. Se busca que se logren manipular imágenes digitales bajo un enfoque simplificado, con el formato RGBD, es decir, canales de colores y además un cuarto canal de profundidad para representar imágenes en 3D, además de poder realizar las siguientes operaciones:

- Recortar imagen
- Invertir una imagen horizontalmente
- Invertir una imagen verticalmente
- Comprimir imagen en base a eliminación del color con mayor frecuencia.
- Convertir a hexadecimal
- Visualizar la imagen
- Rotar imagen en 90° a la derecha
- Rotar imagen en 90° a la izquierda
- Histograma
- Descomprimir imagen en base a restitución del color con mayor frecuencia
- Aplicar operaciones como las anteriores a un área seleccionada dentro de la imagen
- Convertir imagen a blanco y negro
- Convertir imagen a escala de grises
- Editar una imagen a partir de la aplicación de funciones especiales sobre los pixeles
- Separar capas de una imagen 3D en base a la profundidad. De esta forma desde una imagen 3D se puede devolver una lista de imágenes 2D
- Redimensionar imagen

Diseño de la solución

Para el desarrollo de esta se crearon 13 TDA’s, los cuales se detallarán a continuación, en conjunto a dos diagramas de clases UML, uno realizado manualmente antes de realizar el código del proyecto y uno realizado posteriormente de manera automática (se encuentran en detalle en los anexos):

- Diagrama pre-solución:



- Clase Abstracta Píxel:

  Clase abstracta que representa un píxel, esta posee coordenadas “x” e “y “representadas con enteros, y una lista de métodos que se pueden apreciar en el diagrama UML.

- Clase Pixbit:

  Herencia de Clase Píxel, posee un bit que se representa con un entero (1 o 0) y algunos métodos más que su padre.

- Clase PixRGB:

  Herencia de Clase Píxel, posee un código rgb que se representa con una lista de enteros donde cada elemento representa la intensidad de cada color (r, g, b), posee algunos métodos más que su padre.

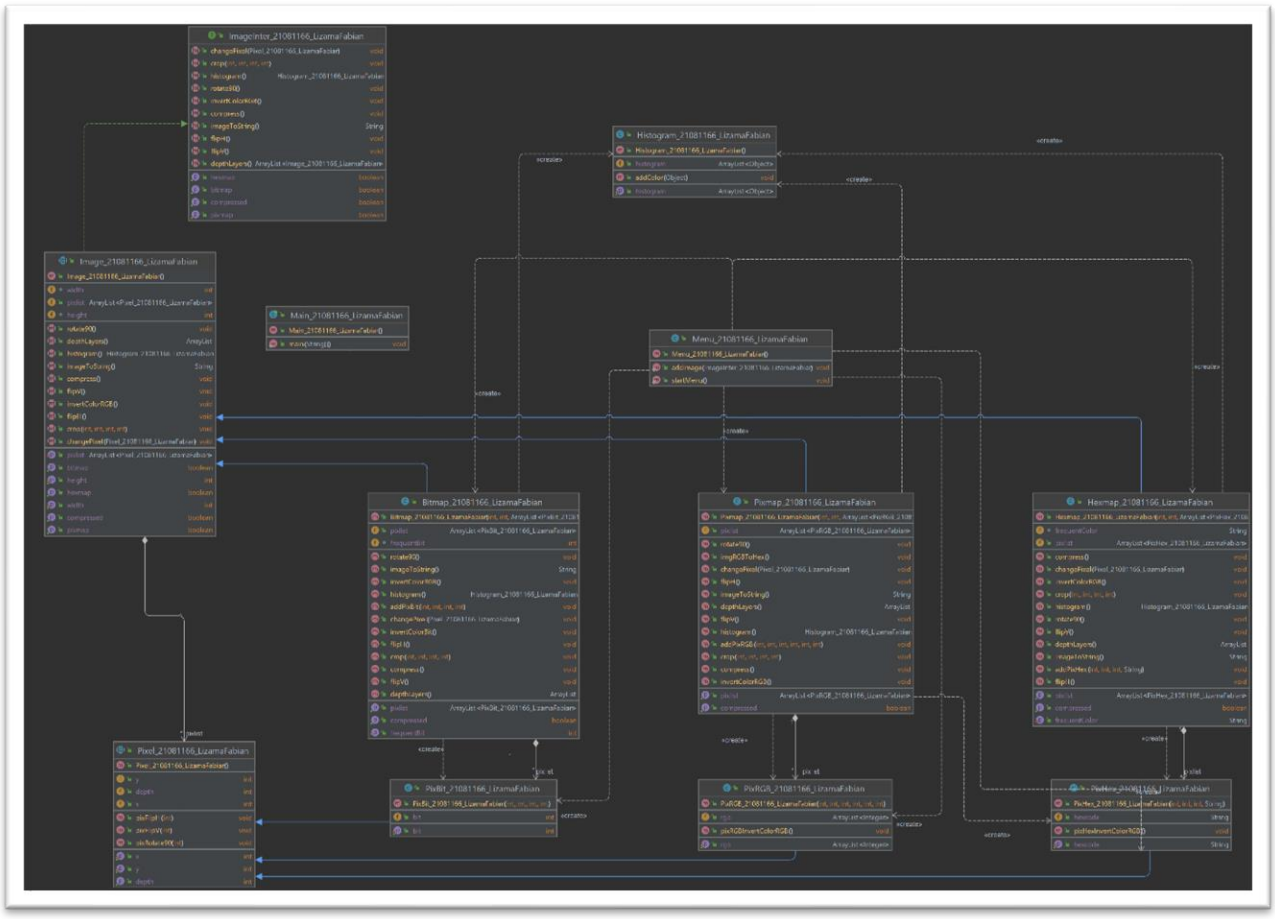
- Clase PixHex:

  Herencia de Clase Píxel, posee un código hexadecimal que se representa con un string en minúsculas, de la forma “ffffff” y algunos métodos más que su padre.

- Interfaz ImageInter:
  Interfaz que representa los métodos que tendrán las Imágenes.
- Clase Abstracta Image:
  Clase abstracta que implementa ImageInter, que representa una imagen, posee un largo, un ancho y una lista homogénea de los pixeles que la componen.
- Clase Bitmap:
  Herencia de Image, representa una imagen formada solo por Pixbit’s.
- Clase Pixmap:
  Herencia de Image, representa una imagen formada solo por PixRGB’s.
- Clase Hexmap:
  Herencia de Image, representa una imagen formada solo por PixHex’s.
- Clase Histogram:
  Clase que representa un histograma, el cual es una estructura que alberga la cantidad de veces que aparece un color en una imagen, posee una lista de pares, los pares son de la forma (X, color) donde X representa el número de ocurrencias de ese color en específico, por su parte “color” puede ser un bit, un código hexadecimal o una lista rgb.

- Clase Menu:
  Representa el menú con el cuál el usuario interactúa con el programa (mediante consola), tiene relaciones con la mayoría de las clases nombradas anteriormente.
- Clase Main:
  Clase necesaria para el funcionamiento de cualquier programa en Java, desde ahí comienza la ejecución del proyecto.

- Diagrama post-solución:



Aspectos de Implementación

Para el proyecto se utiliza el lenguaje “Java” con OpenJDK en su versión 11, para escribir el código se utiliza el IDE “IntelliJ” de JetBrains en su versión “2020.3” junto al plugin “Diagrams”, además para la realización del primer diagrama UML se utilizó la plataforma de navegador “draw.io”, todo esto en el sistema operativo de Windows 10, el programa se organiza en un repositorio con los siguientes archivos:

Bitmap_21081166_LizamaFabian.java:

Archivo de código java que representa la clase Bitmap.

Pixmap_21081166_LizamaFabian.java:

Archivo de código java que representa la clase Pixmap.

Hexmap_21081166_LizamaFabian.java:

Archivo de código java que representa la clase Hexmap.

Histogram_21081166_LizamaFabian.java

Archivo de código java que representa la clase Histogram.

Image_21081166_LizamaFabian.java:

Archivo de código java que representa la clase Image.

ImageInter_21081166_LizamaFabian.java:

Archivo de código java que representa la clase ImageInter.

Main_21081166_LizamaFabian.java:

Archivo de código java que representa la clase Main.

Menu_21081166_LizamaFabian.java:

Archivo de código java que representa la clase Menu.

Pixbit_21081166_LizamaFabian.java:

Archivo de código java que representa la clase Pixbit.

Pixel_21081166_LizamaFabian.java:

Archivo de código java que representa la clase Pixel.

PixHex_21081166_LizamaFabian.java:

Archivo de código java que representa la clase PixHex.

PixRGB_21081166_LizamaFabian.java:

Archivo de código java que representa la clase PixRGB.

informe_21081166_LizamaGonzalez.pdf:

El presente informe.

autoevaluacion_21081166_LizamaGonzalez.txt

Archivo de texto con la autoevaluación de cada requerimiento funcional y no funcional.

Compilador_21081166_LizamaGonzalez.bat

Script en batch de compilación y ejecución para el proyecto.

Instrucciones de Uso

Para utilizar el programa se necesita un computador con Windows, se debe ejecutar el fichero “compilador_21081166_LizamaGonzalez.bat” presente en el repositorio del proyecto, al hacerlo se abrirá la consola de Windows “cmd” y se compilará y ejecutará el programa (en el caso de que solo quiera compilar sin ejecutar, edite el archivo y elimine la tercera línea de código “java Main_21081166_LizamaFabian”), ahí solo siga las instrucciones para utilizar el software, como aspecto a tomar en cuenta, el programa se ejecutará con una imagen de tipo Bitmap en el slot 1 de memoria.

- Importante: cuando el programa solicite ingresar el código hexadecimal ingrese 6 caracteres con letra minúscula y sin el carácter de #, ejemplo:

00ffa2

Resultados y Autoevaluación

Se espera obtener un buen resultado sin fallos en la ejecución del programa, podría llegar a fallar si no se utiliza el software como se debe otorgando valores distintos a los que se solicitan, además se tomó la decisión de no implementar el método “Decompress” como decisión estratégica para terminar el proyecto a tiempo.

Conclusiones

Utilizar el paradigma Orientado a Objetos fue bastante agradable luego de realizar los laboratorios anteriores con los otros paradigmas, ya que este pareciera ser mucho más intuitivo y se parece mucho más a conceptos que tenemos en la vida real (objetos), lo más complejo fue entender las distintas relaciones entre los objetos, a diferencia de otros lenguajes, se desarrolló la solución sin problemas mayores (omitiendo el imageDecompress ya que no implementarlo era lo esperado), por lo cual se obtuvieron los resultados esperados.

Anexos

Diagrama UML pre-solución:

https://drive.google.com/file/d/1L32dVF8AvPBL03n_5u10ws7pLKzNqH7Z/view?usp=sharing

Diagrama UML post-solución:

https://drive.google.com/file/d/15L-LHU-VnIOOG2y-9uM5sooIw_FnrSuF/view?usp=sharing