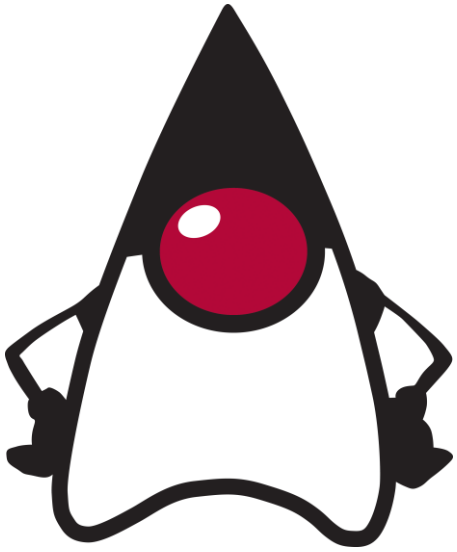
UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Paradigmas de la programación

Informe de laboratorio N°3



Software de edición o manipulación de imágenes

Andres Zelaya

Profesor Gonzalo Martínez

03 de diciembre 2022

El presente informe corresponde al laboratorio N°3 del curso; donde en cada experiencia se trabajará utilizado un paradigma de la programación distinto.

Siendo utilizado en esta ocasión el Paradigma orientado a objetos, donde es ocupado el lenguaje de programación Java.

En el informe sigue se cubrirán los siguientes tópicos:

- Descripción breve del problema
- Introducción al paradigma
- Objetivos del proyecto
- Análisis del problema,
- Diseño de solución del problema
- Aspectos de implementación
- Instrucciones y ejemplos de uso
- Resultados y autoevaluación
- Conclusión

Descripción del problema

El proyecto de este semestre es la creación de un software de edición y/o manipulación de imágenes digitales, es decir, que permita a un usuario realizar distintas operaciones sobre estas: rotar una imagen, invertirla, retocarla, transformarla y redimensionarla, entre otras. Algunos softwares más conocidos, capaces de realizar esta tarea son GIMP y Adobe Photoshop.

Este proyecto se concentrará en trabajar en imágenes BIT-D, RGB-D y Hex-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, por lo que el desafío de esta experiencia es implementar lo descrito anteriormente de forma que se acople al paradigma orientado a objetos, de modo que se puedan realizar las operaciones mencionadas.

De esta forma, en la imagen bidimensional de la Figura 1 (mostrada en el apéndice), donde es posible distinguir los colores en el espectro RGB; al incorporar la dimensión (D)epth 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, como se ilustra en la Figura 2.

Descripción del paradigma

Los objetos en la programación son abstracciones de ellos mismos, disponibles en la realidad; por ejemplo, si se habla de un zoológico, se pueden tener algunos como mamíferos (perros, gatos, jirafas, etc.) o reptiles (lagartos, serpientes, etc.). También, se comunican entre sí mediante el paso de mensajes.

El paradigma orientado a objetos está basado en el concepto de estos, es decir, unidades que pueden tener características conocidas como atributos, y comportamientos conocidos como métodos, pudiendo variar. También los métodos de un objeto pueden leer y escribir las características de este, es decir, tienen noción de sí mismos. En este paradigma, los programas son diseñados con el propósito de crearlos y que interactúen entre sí.

Esta programación establece un equilibrio entre la importancia de los procesos y los datos, mostrando un enfoque más cercano al pensamiento del ser humano. Esto es gracias a la herencia, facilitando el crecimiento y la mantenibilidad. Sus bases son: abstracción, modularidad, encapsulación y jerarquización. Para este práctico nos centraremos en las primeras 2.

La abstracción es un proceso mental de extracción de las características esenciales, ignorando los detalles superfluos. Resulta ser muy subjetiva dependiendo del interés del observador, permitiendo abstracciones muy diferentes de la misma realidad.

El modularidad es descomponer un sistema en un conjunto de partes. De ella se desprenden dos conceptos muy importantes: acoplamiento y cohesión.

- El acoplamiento entre dos módulos mide el nivel de asociación entre ellos; nos interesa buscar módulos poco acoplados
- La cohesión de un módulo mide el grado de conectividad entre los elementos que los forman; nos interesa buscar una cohesión alta.

Objetivos

Como objetivos del laboratorio se tiene aprender sobre el Paradigma y la programación orientada a objetos, para así obtener la habilidad de programar de otra forma distinta a la que se tiene costumbre actualmente.

Otro objetivo es programar correctamente en Java y aprender a utilizar las herramientas del paradigma para completar el proyecto de laboratorio y así poder tener una base para los futuros laboratorios y otros proyectos que en un futuro se desarrollen con la POO.

Análisis del problema

En este paradigma, todo se trabaja usando objetos, donde estos pueden realizar distintas acciones a sí mismos o interactuar con otros objetos.

El desafío principal de este laboratorio está en lograr que distintos objetos tales como el píxel y la imagen puedan interactuar entre si mediante las indicaciones de un usuario a través de la consola, es decir, un menú.

Debido al funcionamiento del lenguaje y del paradigma, ya analizando el contexto del problema se puede deducir la existencia de 3 clases: píxel, imagen y menú. Estas últimas pueden variar en su forma de implementación, ya sea con clases abstractas o interfaces, pero son la base de lo que debe estar implementado. (Figura 3)

Diseño de solución

En esta ocasión se trabajará con distintas clases, las cuales pueden homologar un TDA, puesto que cada clase puede contener:

- Atributos
- Constructores
- Funciones de Pertenencia
- Selectores
- Modificadores
- Otras funciones asociadas al TDA.

Para la correcta implementación y desarrollo de este laboratorio, se emplean distintas clases que logran realizar de forma correcta las diversas operaciones requeridas para esta instancia de laboratorio. Estas fueron:

- Pixel
- Image
- ImageCompressed

Sin embargo, aparte de clases, se construyeron interfaces y clases abstractas, esto se puede ver representado en el diagrama UML (Figura 4). Estos elementos fueron:

- ImageFormat – Clase abstracta
- ImageOperations – Interface
- ImageCompressedOperations - Interface

Para esta implementación se interpretará la imagen como un arreglo de pixeles, donde tanto la imagen como cada píxel es una instancia diferente de una clase. Por esto mismo es que gran parte de las operaciones de la imagen son llamados en masa a métodos que tienen lugar a nivel píxel.

Aspectos de implementación

Dado el paradigma, cada clase cuenta con sus propias características y comportamientos, donde una de estas tiene su propio archivo individual. Esta estructura contempla el uso de herencia a partir de clases abstractas y el uso de interfaces que definen el comportamiento de determinado tipo de objeto.

La naturaleza del laboratorio esta enfocada en usar netamente los TDA o archivos de clases con el propósito de seguir el paradigma de programación orientada objetos, por lo que así se evita el cambio de paradigma.

Especificaciones utilizadas por el entorno:

- Build system: Gradle
- JDK: Azul Zulu versión 11.0.17
- Versión de Java: 11.0.16.1
- Windows 10 o superior

Instrucciones de uso

Dada la naturaleza de laboratorio, se asume que se cuenta con un equipo que posee una versión de java igual o superior a 11.

Para iniciar el programa es necesario dirigirse a la carpeta “ImageEditor” y una vez dentro hacer doble click en el archivo “build&run.bat”, esto abrirá una pestaña de CMD ejecutando la implementación de laboratorio.

Estando ya en el programa:

Se inicia con el menú principal, donde el usuario podrá decidir que hacer entre diversas opciones como: Mostrar las imágenes disponibles, manipular una imagen, crear una imagen, escoger entre diversas utilidades de las imágenes o salir del programa. Cada una de estas opciones a excepción de salir del programa lleva a submenús los cuales son bastante intuitivos.

Es imperante mencionar que el programa viene con una imagen predefinida para poder probar las diversas funcionalidades sin la necesidad de crear una imagen inicial.

Resultados esperados

Se espera poder trabajar exitosamente con imágenes de distintos tipos, donde la implementación del programa sea completamente funcional y sin errores.

Todo esto logrado mediante un uso correcto del paradigma y una adecuada implementación de las distintas funcionalidades del lenguaje tales como: clases, interfaces, clases abstractas.

Posibles errores

No se esperan errores por parte de la ejecución del programa, pero si el usuario ingresa valores no correspondientes a lo solicitado es posible que se generen errores.

Resultados obtenidos

Los resultados obtenidos fueron los esperados, ya que se logró crear un programa completamente funcional con el cual el usuario es capaz de interactuar.

Se hicieron múltiples pruebas con distintos ejemplos para probar de que no hubiera fallos en la ejecución del código y que el código hiciera lo correcto, lográndose la implementación de 20/20 funciones.

Autoevaluación

Las funciones que si fueron implementadas funcionan en la totalidad de las veces probadas.

La Autoevaluación se realiza 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.

Se adjunta una taba de autoevaluación individual por funciones como anexo en la tabla 1.

Conclusiones

Luego de realizar y completar el proyecto, se puede concluir que se cumplieron los objetivos principales, ya que fue posible aprender a utilizar Java para poder completar el proyecto de laboratorio correspondiente.

Lo más difícil de esta experiencia fue hacer interactuar los distintos objetos entre sí, por ejemplo, lograr que una imagen interactúe con un grupo de píxeles.

Por otro lado, no se experimentaron complicaciones por el lado de la instalación o uso del compilador ni por el aspecto del versionamiento con Git.

Bibliografía y referencias

1. Gonzales, R. (2022). “Proyecto Semestral de Laboratorio”. Paradigmas de Programación. Enunciado de Proyecto Online. https://docs.google.com/document/d/1D6g3S3vLC-zziOsSprLIBPypkd89s2QEkJs1F_kbHm4/edit?pli=1
2. Gonzales, R (2022) “Laboratorio 3 (Paradigma Orientado a Objetos - Lenguaje Java o C#)” Paradigmas de la programación. https://docs.google.com/document/d/1ymOs4hi2NYhFbJDJIkgwZkM-qBd8MF5_BkwTk9d7qcg/edit#heading=h.oj1cr4ayhg4m
3. Chacón, S. y Straub, B. (2020). “Pro Git – Todo lo que necesitas saber sobre Git’’. Libro Online. Recuperado de: https://drive.google.com/file/d/1mHJsfvGCYcIhdrmK-lBI6a1WS-U1AAPi/view

Anexos

Figura 1. Lenna, una de las imágenes más empleadas (desde 1973) como imágenes estándar de prueba de algoritmos de compresión
Fuentes:
https://taglang.io/blog/post/Understanding_Kernel_Convolution_Part_2/

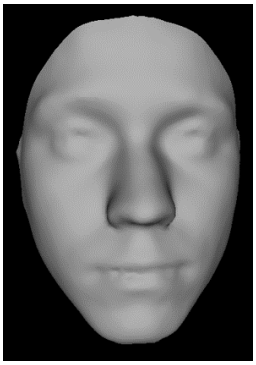


Figura 2. Reconstrucción sintética de un rostro a partir de una imagen RGBD.
Fuente:
https://www.researchgate.net/publication/316442804_Facial_depth_map_enhancement_via_neighbor_embedding/figures?lo=1

Figura 3.

Diagrama de análisis

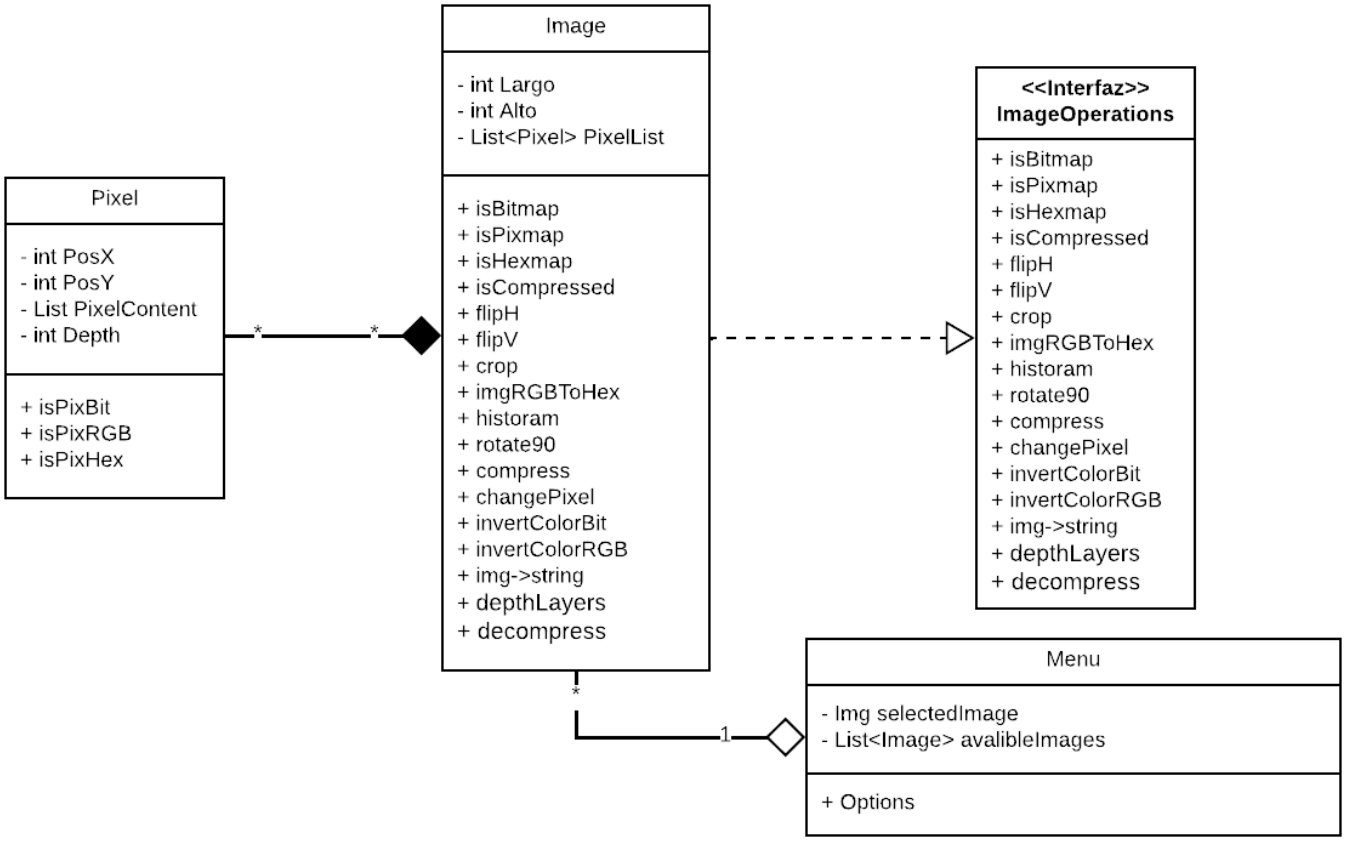


Figura 4.

Diagrama de diseño

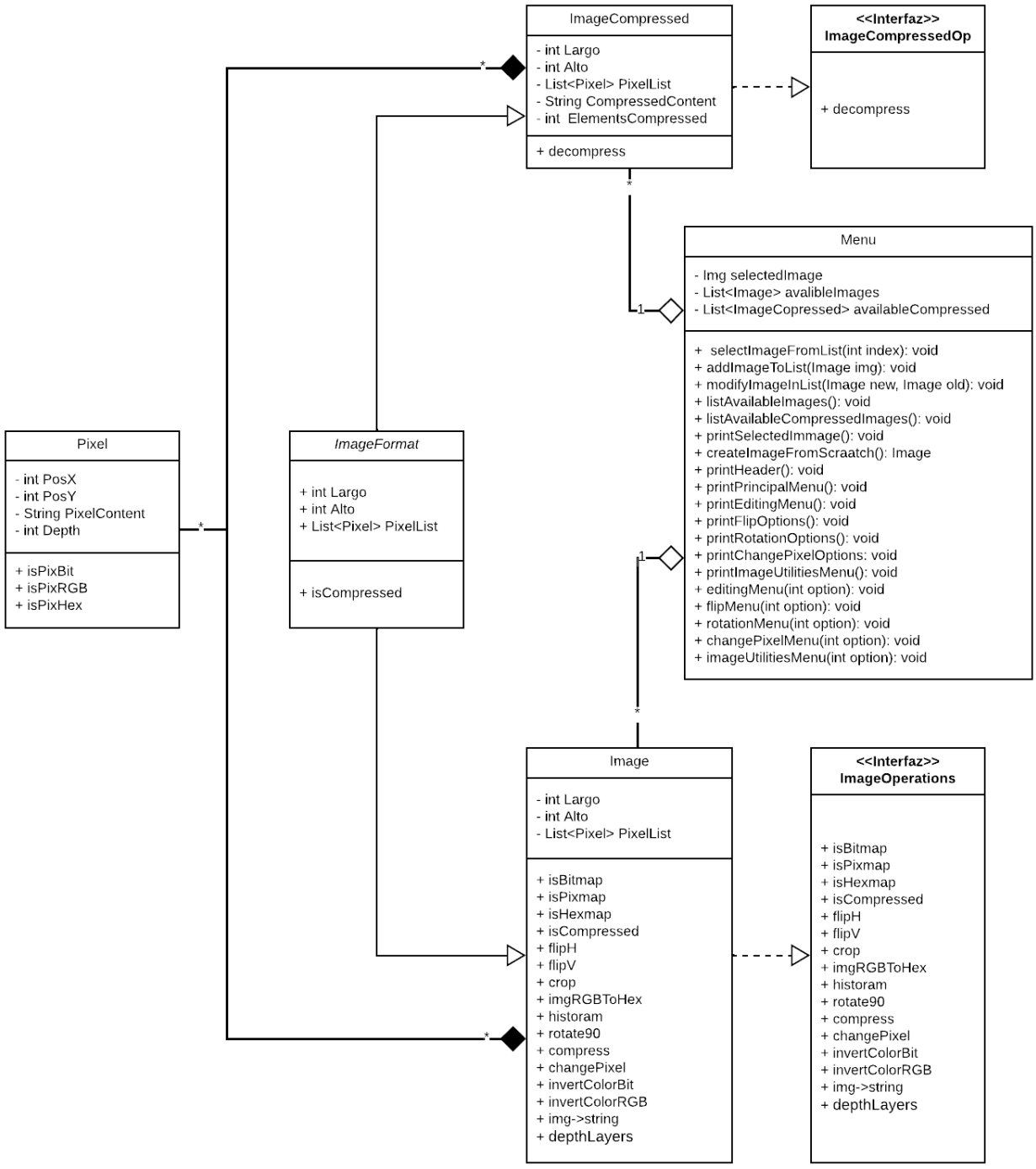


Tabla 1.

Autoevaluación de requerimientos funcionales.

TDAs	1
Menú interactivo por terminal	1
TDA image - constructor	1
TDA image - isBitmap	1
TDA image - isPixmap	1
TDA image - isHexmap	1
TDA image - isCompressed	1
TDA image - flipH	1
TDA image - flipV	1
TDA image - crop	1
TDA image - imgRGBToHex	1
TDA image - histogram	1
TDA image - rotate90	1
TDA image - compress	1
TDA image - changePixel	1
TDA image - invertColorBit	1
TDA image - invertColorRGB	1
TDA image - image->string	1
TDA image - depthLayers	1
TDA image - decompress	1