**UNIVERSIDAD DE SANTIAGO DE CHILE**
**FACULTAD DE INGENIERÍA**
**DEPARTAMENTO DE INGENIERÍA INFORMÁTICA**

# Informe De Laboratorio: Creación De Una Aplicación Estilo Photoshop En Java

Integrantes: Joaquín Saldivia
Profesor: Roberto Gonzales
Asignatura: Paradigmas de programación

**UNIVERSIDAD DE SANTIAGO DE CHILE**
**FACULTAD DE INGENIERÍA**
**DEPARTAMENTO DE INGENIERÍA INFORMÁTICA**

# Contenido

# 1.- Introducción

En este informe se expondrá una solución al problema de laboratorio el cual es realizar una aplicación para editar imágenes, utilizando el paradigma orientado a objetos, donde se utilizará el lenguaje Java para la implementación de este.

## 1.1.- Descripción del problema

Se desea construir un programa que permita crear y editar imágenes al estilo de Photoshop o GIMP, agregando también la cualidad de profundidad a las imágenes. Se debe tener en cuenta que se tienen las siguientes estructuras como base: Imagen: Siendo la estructura principal del programa este se compone de un ancho, largo y pixeles, pudiendo aplicarles distintos tipos de edición. Pixeles: Son los que representan la imagen como tal teniendo una estructura de posición en el ancho y largo, color del píxel y su profundidad, este tendrá 3 tipos de formato y son pixbit, pixrgb y pixhex.

## 1.2.- Descripción del paradigma

El paradigma orientado a objetos es un paradigma imperativo, que funciona a base de clases, métodos y atributos.
Los elementos más importantes de este paradigma son:

**Clases:** Son definición de un objeto el cual tiene atributos y métodos, representando la implementación de un TDA.

**Objetos:** Son instancias de una clase, refiriéndose a una representación activa

**Atributos:** Corresponde a lo que compone una clase, y también puede ser otra clase.

**Metodos:** Son los comportamientos de un objeto o también se le conocen como las “funciones” que lo componen

**Encapsulamiento:** Se utiliza para definir que va a ser visible dentro de una clase teniendo 3 tipos: privado que es visible solo para la clase misma, protected que es visible para todo el paquete de trabajo y public que es visible para todas las clases externas.

## 1.3.- Objetivos

Como objetivo se tiene desarrollar el entendimiento del paradigma orientado a objetos a través de Java con la idea de realizar por completo la solución al problema planteado

# 2.- Desarrollo

## 2.1.- Análisis del problema

Se tienen 2 entidades principales las que son la imagen y los pixeles que formaran parte de esta donde estos pueden ser de 3 tipos pixbit, pixhex y pixrgb, además la imagen tiene que aceptar distintos tipos de cambios también los pixeles deben aceptar cambios para operar con éxito los predicados en las imágenes y los predicados mencionadas que debe aceptar son:

bitmap?, pixmap?, hexmap?, compressed?, flipH, flipV, crop, rgbtoHex, histogram, rotate90, compress, changePixel, invertColorBit, invertColorRGB, imageToString, depthLayers,decompress

También se solicita que cada TDA utilizado este en un archivo propio.

## 2.2.- Diseño de solución

Al diseñar la solución se ocuparon los tipos de datos nativos del lenguaje y secrean 2 TDA que serán utilizados para simplificar el problema los que son:

TDA Image:

- Representación: una lista con la composición 2 int representando ancho y largo, una lista de pixeles y un color comprimido

- Constructor: se entregan solo los primeros 3 de los antes mencionados, esdecir: 2 int y una lista de pixeles que puede ser vacía para crear la lista imagen

- Métodos: Son los métodos solicitados más getters y setters

TDA Pixel:

- Representación: una lista que contiene 2 int indicando las posiciones x e y,un color que puede ser una lista de 3 int, un string o un int y la profundidad dependiendo del tipo de color que posea se le agregara un apellido al píxel.

- Constructor: al entregar 2 int un color y la profundidad se crea una lista queserá conocido como pixbit, pixrgb o pixhex por lo antes mencionado.

- Métodos: Son los getters y setters

Luego la relación que se estableció inicialmente entre estos TDA se podrá ver en Anexo 1 y su versión final en el Anexo 4, además de mencionar que, dada la similitud de varios métodos estas se descomponen en predicados más pequeñas dado el tipado del lenguaje.

## 2.3.- Aspecto de implementación

El compilador utilizado es Java junto con JDK en su versión 11, se trabaja con clases y objetos, se implementan métodos que permiten acceder a sus atributos.

El código se estructura por TDA, para facilitar la edición a la hora de agregar contenido a alguno de estos, utilizando import para traer módulos de Java y sin necesidad de importar los TDA ya que se trabajan en un mismo paquete.

## 2.4.- Ejemplos de uso

Se tiene un archivo App una instancia de menú que permite iniciar la UI para trabajar las imágenes.

Ahora en el anexo 2 se puede observar una manera de crear una imagen y la forma de visualizarla como de modificarla.

También se espera poder aplicar a las imágenes los métodos de edición sin ningún problema. Refiriéndose a los errores no se espera ningún error en concreto.

## 2.5.- Resultados y autoevaluación

Los resultados son los esperados, logrando hacer un programa completamente funcional, donde se intentó colocando entradas erróneas aplicando más de una vez una función a una imagen e intentando imprimir imágenes comprimidas.

Se logran completar los 17 métodos propuestos para el proyecto

La autoevaluación radica entre valores de 0 (no implementado) hasta 1 (implementado y sin errores) aumentando en una escala de 0.25 y se puede observar en el anexo 4. Se les asigna a todos los predicados 1 dado que al momento de realizar diversas pruebas estas trabajan correctamente.

## 3.- Conclusión

Una vez terminado el trabajo se logra cumplir los objetivos, mejorando el entendimiento del paradigma orientado a objetos, así como también lograr una aplicación en Java para editar y crear imágenes.

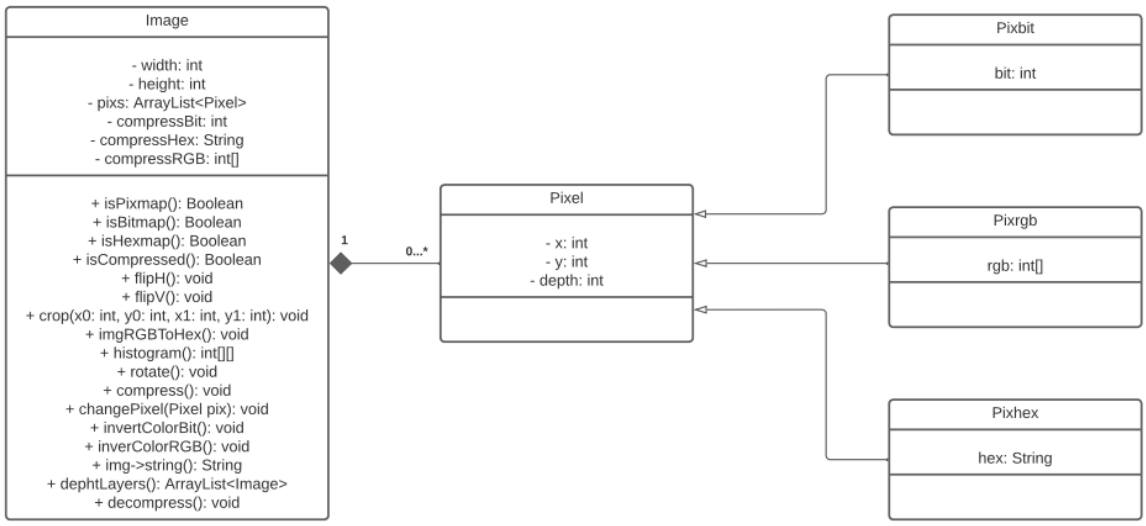
La mayor limitación de este paradigma y lo que causo más dificultad al empezar este proyecto fue la forma en la que se trabaja mediante clases, también dificulto el lenguaje de Java al ser tipado lo que implicaba tener que implementar varias veces un mismo método. Además se notan diferencias entre este paradigma y los paradigmas lógico y funcional, dado que se trabaja de manera imperativa y se nota una gran diferencia al momento de desarrollar los métodos, finalmente al trabajar en Scheme si había un fallo se regresaba un error y en Swi-Prolog se retorna false pero Java retorna un listado de bibliotecas internas registrando errores que es casi imposible determinar la falla.

## 4.- Bibliografía

1. Flores, V. (2022). “Proyecto semestral de laboratorio”. https://docs.google.com/document/d/14XEMPsuHicTIK7Hv3OuvW3wlG-SE1Ms9sJq9xwVLJSc/edit

## 5.- Anexos

1. Diagrama uml inicial




2. Ejemplos de uso:

```
### Manipulador de imagenes ###
Escoja su opcion:
1. Crear una imagen
2. Operaciones de imagen
3. Visualizar imagen
4. Terminar programa

INTRODUZCA SU OPCION: 3

0 1 0 1 1
0 0 1 0 0
1 1 0 1 0
0 1 1 0 0
```

```
### Manipulador de imagenes ###
Escoja su opcion:
1. Crear una imagen
2. Operaciones de imagen
3. Visualizar imagen
4. Terminar programa

INTRODUZCA SU OPCION: 1
```

```
### Manipulador de imagenes ###
Escoja su opcion:
1. Crear una Bitmap
2. Crear una Pixmap
3. Crear una Hexmap
4. Volver

INTRODUZCA SU OPCION: 2
```

```
### Manipulador de imagenes ###
Ingrese las dimensiones:
Ancho: 3
Largo: 2

### Manipulador de imagenes ###
1. Pixeles Manuales
2. Pixeles Automaticos
3. Volver

INTRODUZCA SU OPCION: 2
```

```
### Manipulador de imagenes ###
Escoja su opcion:
1. Crear una imagen
2. Operaciones de imagen
3. Visualizar imagen
4. Terminar programa

INTRODUZCA SU OPCION: 3

[21, 208, 202] [199, 161, 184] [193, 230, 92]
[211, 179, 227] [66, 75, 202] [33, 213, 27]
```

```
### Manipulador de imagenes ###
Escoja su opcion:
1. Verificar Bitmap
2. Verificar Pixmap
3. Verificar Hexmap
4. Aplicar FlipH
5. Aplicar FlipV
6. Aplicar Crop
7. Ver Histograma
8. Aplicar Rotate90
9. Aplicar Compress
10 Cambiar un Pixel
11 Invertir Color Bit
12 Invertir Color RGB
13 Aplicar DepthLayers
14 Aplicar Decompress
15. Volver

INTRODUZCA SU OPCION: 13
```

```
[255, 255, 255] [255, 255, 255] [193, 230, 92]
[255, 255, 255] [255, 255, 255] [255, 255, 255]

[21, 208, 202] [199, 161, 184] [255, 255, 255]
[255, 255, 255] [255, 255, 255] [255, 255, 255]

[255, 255, 255] [255, 255, 255] [255, 255, 255]
[211, 179, 227] [255, 255, 255] [255, 255, 255]

[255, 255, 255] [255, 255, 255] [255, 255, 255]
[255, 255, 255] [66, 75, 202] [255, 255, 255]

[255, 255, 255] [255, 255, 255] [255, 255, 255]
[255, 255, 255] [255, 255, 255] [33, 213, 27]
```

3. Autoevaluación:

| Autoevaluacion | Puntaje |
|---|---|
| TDA | 1 |
| image | 1 |
| isPixmap? | 1 |
| isHexmap? | 1 |
| isBitmap? | 1 |
| isCompressed? | 1 |
| flipH | 1 |
| flipV | 1 |
| crop | 1 |
| rgbToHex | 1 |
| histogram | 1 |
| rotate90 | 1 |
| compress | 1 |
| changePixel | 1 |
| invertColorBIt | 1 |
| invertColorRGB | 1 |
| imgToString | 1 |
| depthLayers | 1 |
| decompress | 1 |

4. Diagrama uml final: