UNIVERSIDAD
DE SANTIAGO
DE CHILE

INFORME DE LABORATORIO 3:
SIMULACIÓN DE EDITOR DE IMÁGENES EN JAVA

Nombre: Diego Armijo Palominos
Profesor: Victor Flores
Fecha: 03/12/2022

TABLA DE CONTENIDOS

1. INTRODUCCIÓN

Un paradigma es todo aquel modelo, patrón o ejemplo que debe seguirse en determinada situación. En la programación, se refieren a este como una manera o estilo de programación de software. Existen diferentes formas de diseñar un lenguaje de programación y varios modos de trabajar para obtener los resultados que necesitan los programadores. Se trata de un conjunto de métodos sistemáticos aplicables en todos los niveles del diseño de programas para resolver problemas computacionales. En este informe se introducirá un problema y su respectiva solución en el paradigma orientado a objetos de programación, para ser más específicos, en el lenguaje de programación **Java**.

1.1 DESCRIPCIÓN DEL PROBLEMA

Se pide desarrollar un programa tipo “Photoshop”, este tiene que permitir crear una imagen y hacerle ciertas modificaciones las cuales describiremos más adelante. Esta imagen se rige por ciertas propiedades que debemos tener en cuenta:

- **Image**: En base a ciertos parámetros, se crea una imagen que contiene el ancho, alto y los píxeles de la imagen.
- **Pixbit**: Tipo de píxel que acepta la imagen, este es de tipo bit.
- **Pixrgb**: Tipo de píxel que acepta la imagen, este es de tipo rgb.
- **Pixhex**: Tipo de píxel que acepta la imagen, este es de tipo hexadecimal.

Todo esto lo implementaremos a través de clases y métodos que explicaremos en el punto 1.2.

1.2 DESCRIPCIÓN DEL PARADIGMA

La programación orientada a objetos (POO) es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas de computadoras. En este paradigma las entidades centrales son los objetos, que son tipos de datos que encapsulan con el mismo nombre de estructura de datos, operaciones o algoritmos que manipulan esos datos.

En el paradigma orientado a objetos hay que tener en cuenta algunos ejemplos bastantes importantes a la hora de construir código utilizando este paradigma, como por ejemplo:

Objetos: Son instancias de una clase, es decir, representaciones activas de estas.

Clases: Son la definición de las características de un objeto, las cuales tienen atributos y métodos. Una clase corresponde a decir que es la implementación de un TDA.

Atributos: Corresponde a lo que compone una clase, como son distintos tipos de datos.

Métodos: Son los comportamientos de los objetos, es decir, las “funciones” que componen a las clases. Estos métodos expresan los comportamientos que puede realizar un objeto sobre sí mismo o sobre otros objetos.

Para organizar mejor las ideas y la información, se suelen utilizar diagramas, donde el más famoso es el **UML** (diagrama de clase), el cual muestra las relaciones que existen entre las distintas clases que componen un código.

Cabe mencionar que existen varias técnicas que componen este paradigma, donde se incluye la herencia, abstracción, polimorfismo y encapsulamiento.

1.3 OBJETIVOS

El objetivo principal de este proyecto es poder desarrollar el programa de forma correcta utilizando todas las herramientas que nos brinda el **PPO** y lenguaje de programación **Java**. Otro objetivo es que mientras se desarrolla este proyecto ir aprendiendo y desarrollando la habilidad de programar de otra forma distinta a la que se está acostumbrado tradicionalmente.

2. DESARROLLO

2.1 ANÁLISIS DEL PROBLEMA

Para generar una imagen, antes hay que crear los píxeles de esta. Cada píxel se rige por ciertas reglas y condiciones, es por eso que hay verificar que los parámetros que ingresa el usuario sean valores que estén dentro del rango aceptado por las condiciones del píxel. También, para poder generar una imagen es necesario saber las dimensiones, ya que en base a estas tendremos que generar las llamadas respectivas para pedir los valores del píxel. Todo esto se irá generando mediante múltiples menús, donde el usuario tendrá que ir ingresando una opción por pantalla.

Pixbit: Corresponde a un conjunto de asociaciones, donde un usuario genera un píxel de tipo bit. En Java se representará como una lista.
Pixrgb: Corresponde a un conjunto de asociaciones, donde un usuario genera un píxel de tipo rgb. En Java se representará como una lista.
Pixhex: Corresponde a un conjunto de asociaciones, donde un usuario genera un píxel de tipo hexadecimal. En Java se representará como.
Image: Corresponde a un conjunto de asociaciones, donde un usuario genera una imagen. En Java se representará como una lista.
Es necesario tener los creadores de pixbit-d | pixrgb-d | pixhex-d para que la imagen pueda ser creada. Cada TDA (Tipo de Dato Abstracto) está dentro de un archivo único, por lo tanto solo bastará con consultar al archivo Main.java.

2.2 DISEÑO DE SOLUCIÓN

Para el diseño de esta solución debemos crear y utilizar los Tipos de Datos Abstractos (**TDA**). Los TDA creados son utilizados para la construcción de las operaciones de otros TDA. Los TDA más importantes son:

Clase Pixbit:

- **Representación:** Se utiliza para crear un Pixel. Cada vez que se haga ingreso de las variables a los métodos respectivamente , se hará entrega de un pixel.
- **Constructor:** Dado un entero coordenada X, un entero coordenada Y, un bit(0/1), y un entero depth, se crea una lista que contiene las propiedades del píxel.
- **Selectores, modificadores y otros métodos:** ver Anexos.

Clase Pixmap:

- **Representación:** Se utiliza para crear un Pixel. Cada vez que se haga ingreso de las variables a los métodos respectivamente , se hará entrega de un pixel.
- **Constructor:** Dado un entero coordenada X, un entero coordenada Y, un entero R, un entero G, un entero B, y un entero depth, se crea una lista que contiene las propiedades del píxel.
- **Selectores, modificadores y otros métodos:** ver Anexos.

Clase Pixhex:

- **Representación:** Se utiliza para crear un Pixel. Cada vez que se haga ingreso de las variables a los métodos respectivamente , se hará entrega de un pixel.
- **Constructor:** Dado un entero coordenada X, un entero coordenada Y, un String hex, y un entero depth, se crea una lista que contiene las propiedades del píxel.
- **Selectores, modificadores y otros métodos:** ver Anexos.

Clase Image:

- **Representación:** Se utiliza para crear una imagen. Cada vez que se haga ingreso de las variables a los métodos respectivamente , se hará entrega de una imagen.
- **Constructor:** Dado un entero width, un entero height, y una lista de pixeles, se crea una lista que contiene las propiedades de la imagen.
- **Selectores, modificadores y otros métodos:** ver Anexos.

2.3 ASPECTOS DE IMPLEMENTACIÓN

2.3.1 COMPILADOR

Para este proyecto es necesario utilizar un IDE de Java, como, por ejemplo, Netbeans o IntelliJ IDEA. Para este proyecto específicamente se utilizó IntelliJ IDEA en su versión 2022.2.4. En el caso de JDK, se utilizó la versión 11 para compilar el programa. Solo se utilizaron funcionalidades pertenecientes a la librería estándar de Java.

2.3.2 ESTRUCTURA DEL CÓDIGO

Este proyecto fue implementado con Gradle, la cual es una herramienta que permite la automatización de compilación de código abierto. Solo basta con compilar con Javac.

2.4 INSTRUCCIONES DE USO

2.4.1 EJEMPLOS DE USO

Lo primero que debemos hacer es instalar el JDK en su versión 11 (verificar que no existan otras versiones instaladas y de ser necesario establecer la versión 11 como principal).
Para el caso de IntelliJ IDEA se debe abrir la carpeta del proyecto y ejecutar el programa (verificar que se encuentre la carpeta src). Cuando el programa se ejecute se deberá introducir una opción del menú.
En el menú en primera instancia le pedirá al usuario el tipo de imagen que quiere crear (pixbit|pixmap|pixhex), luego lo dirigirá a otro menú el cual le permitirá elegir entre visualizar la imagen, ver a qué tipo pertenece la imagen, si la imagen está comprimida, modificar la imagen, crear una nueva imagen o salir del programa. Si elige modificar podrá acceder a la mayoría de los requerimientos funcionales.

2.4.2 RESULTADOS ESPERADOS

Se espera poder crear una imagen válida mediante las opciones que ingrese el usuario por consola, así también, poder modificarla y simular el uso de un editor de imágenes. Todo esto mediante el correcto uso del paradigma orientado a objetos y todos los recursos que nos aporta, los cuales los mencionamos en el apartado **1.2 DESCRIPCIÓN DEL PARADIGMA.**

2.4.3 POSIBLES ERRORES

Los posibles errores pueden ser valores mal ingresados en la entrada que pueden causar que el programa se caiga.

2.5 RESULTADOS Y AUTOEVALUACIÓN

2.5.1 RESULTADOS OBTENIDOS

Los resultados obtenidos fueron los esperados, ya que se logró poder crear una imagen válida, todas las funcionalidades básicas y la mayoría de las opcionales. El programa funciona correctamente, incluyendo casos donde las entradas colocadas son erróneas o donde se decide ingresar otro número - String que no corresponde a ninguna opción o simplemente no es válido.

2.5.2 AUTOEVALUACIÓN

Se realizó de la siguiente forma: 0: No realizado - 0.25: Funciona 25% de las veces - 0.5: Funciona un 50% de las veces - 0.75: Funciona 75% de las veces - 1: Funciona el 100% de las veces. (ver tabla en anexos).

3. CONCLUSIÓN

Tras finalizar el proyecto, se concluye que se cumplió el objetivo principal, ya que se logró realizar esta simulación de imagen de forma correcta, aprendiendo a utilizar el paradigma orientado a objetos de programación.

El POO es complejo para quien no haya salido del paradigma imperativo, ya que entender la idea de clases, métodos y objetos puede ser un poco confuso, no obstante, a lo largo del desarrollo de este proyecto, se ha obtenido un entendimiento

exponencial del paradigma, ya que es muy parecido a la “realidad” de cómo funcionan las cosas en el mundo donde vivimos. Si bien no se lograron realizar los requerimientos en su totalidad, se completó la gran mayoría.

4. BIBLIOGRAFÍA Y REFERENCIAS

1. *Qué es la Programación Orientada a Objetos*. (2021, November 26). Intelequia; Intelequia. https://intelequia.com/blog/post/3072/qu%C3%A9-es-la-programaci%C3%B3n-orientada-a-objetos#:~:text=La%20Programaci%C3%B3n%20Orientada%20a%20Objetos%20(POO)%20es%20un%20paradigma%20de,concepto%20de%20clases%20y%20objetos.
2. Miriam. (2020, November 2). *¿Qué es la Programación Orientada a Objetos?* Profile Software Services. https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/
3. *Paradigma de la programación orientada a objetos*. (2017). Github.io. https://ferestrepoca.github.io/paradigmas-de-programacion/poo/poo_teoria/index.html

5. ANEXOS

Diagrama de análisis.

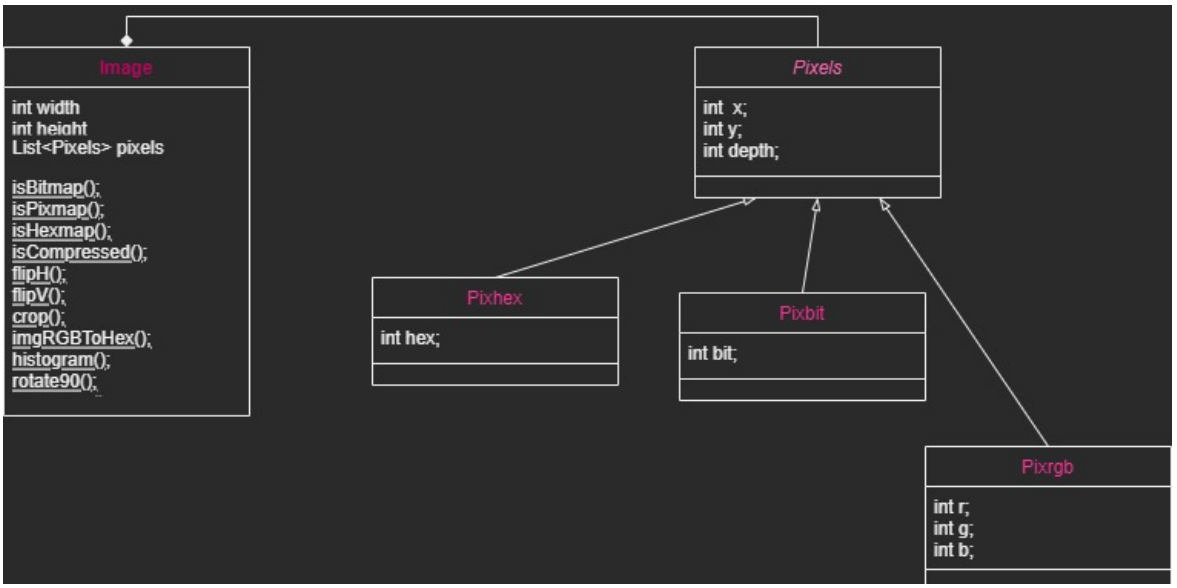
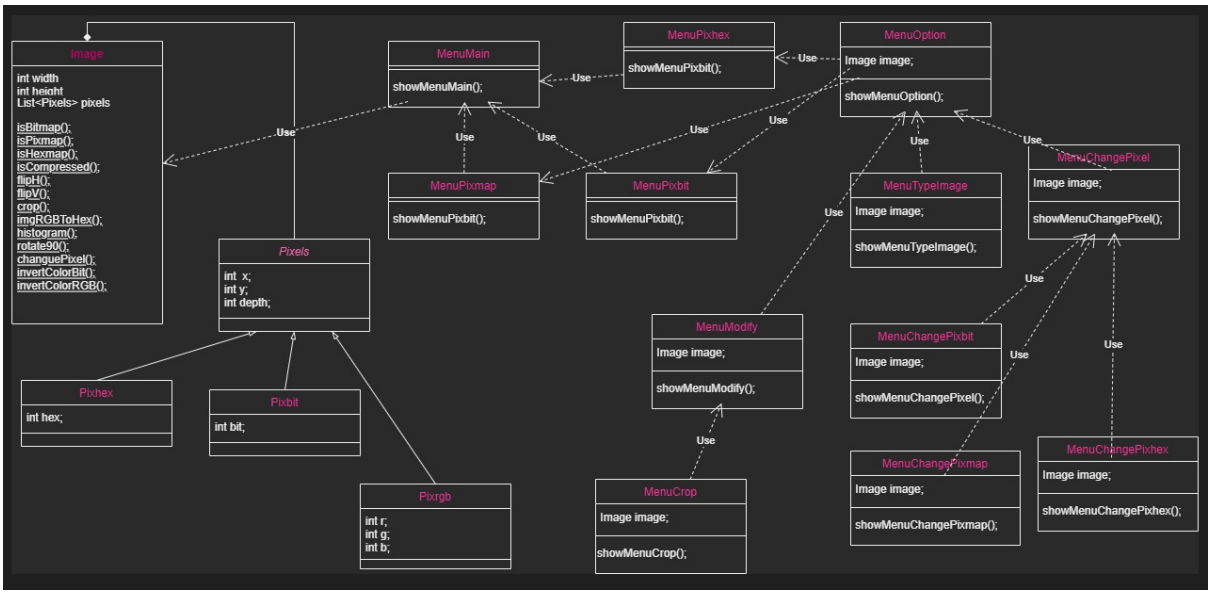


Diagrama de diseño.



Autoevaluación.

Requerimientos Funcionales	Evaluación
Clases y estructuras	1
Menu interactivo por terminal	1
constructor	1
isBitmap	1
isPixmap	1
isHexmap	1
isCompressed	1
flipH	1
flipV	1
crop	1
imgRGBToHex	1
histogram	1
rotate90	1
compress	0
changePixel	1
invertColorBit	1
invertColorRGB	1
toString	1
depthLayers	0
decompress	0