**INFORME DE LABORATORIO 3: SIMULACIÓN DE UN PROGRAMA PARA REALIZAR UN TRATAMIENTO DE IMÁGENES SIMPLIFICADO**

****

Nombre: Aracely Castro V.

Profesor: Roberto Gonzales I.

Asignatura: Paradigmas de Programación (2/2022)

Índice

[1. INTRODUCCIÓN 3](#_Toc119155547)

[**1.1** **DESCRIPCIÓN DEL PROBLEMA** 3](#_Toc119155548)

[**1.2** **DESCRIPCIÓN DEL PARADIGMA** 3](#_Toc119155549)

[2. DESARROLLO 4](#_Toc119155550)

[**2.1** **ANÁLISIS DEL PROBLEMA** 4](#_Toc119155551)

[**2.2** **DISEÑO DE LA SOLUCIÓN** 5](#_Toc119155552)

[**2.3** **ASPECTOS DE IMPLEMENTACIÓN** 5](#_Toc119155553)

[**2.3.1** **EJEMPLOS DE USO** 6](#_Toc119155554)

[**2.3.2** **RESULTADOS ESPERADOS** 6](#_Toc119155555)

[**2.3.3** **POSIBLES ERRORES** 6](#_Toc119155556)

[**2.4** **RESULTADOS Y AUTOEVALUACIÓN** 6](#_Toc119155557)

[**2.4.1** **RESULTADOS** 6](#_Toc119155558)

[**2.4.2 AUTOEVALUACIÓN** 7](#_Toc119155559)

[3. CONCLUSIÓN 7](#_Toc119155560)

[4. BIBLIOGRAFÍA Y REFERENCIAS 8](#_Toc119155561)

[5. ANEXOS 8](#_Toc119155562)

# **INTRODUCCIÓN**

Con el objetivo de aplicar conceptos del paradigma de programación orientado a objetos para el desarrollo de un programa de tratamiento de imágenes simple cuyas interacciones se puedan lograr por consola, a continuación, se dará detalle sobre el proceso de solución al problema implementado a través del IDE Apache NetBeans versión 15 con lenguaje Java. El informe constará de una breve introducción de cómo surgió el problema, una descripción del paradigma utilizado, el análisis del problema y como fue el diseño de las soluciones para algunos métodos, sus aspectos de implementación, las instrucciones necesarias para compilar el archivo con el script.bat junto con ejemplos de algunas interacciones del programa, los resultados y la autoevaluación. Finalmente, se dará una conclusión respecto a todo lo anterior.

## **DESCRIPCIÓN DEL PROBLEMA**

Al igual que en los laboratorios anteriores, se busca desarrollar la simulación de un programa para el tratamiento de imágenes simple como GIMP y Adobe Photoshop, con la diferencia de que ahora se podrá interactuar a través de la consola. Este simulador permitiría crear imágenes con distintos formatos de píxeles sobre los cuales se pueden aplicar distintas operaciones como, por ejemplo, recortar una imagen, invertir una imagen, obtener histograma de la imagen, entre otros. Para implementar esto, se debe de tener en cuenta que:

Una imagen pixmap-d es una imagen donde cada uno de sus píxeles contiene información de espacio de colores y profundidad (R)ed, (G)reen, (B)lue y (D)epth. Cada color cubre valores entre 0 y 255 formando en conjunto un color del espectro RGB. La profundidad, en cambio, ofrece información más detallada de la imagen como espacio tridimensional.

Una imagen bitmap-d es una imagen donde sus píxeles tiene asociado su color por un bit, negro como 0 y blanco como 1. Cada píxel también tendría una profundidad asociada.

Una imagen hexmap-d es similar a una imagen pixmap-d con la diferencia que los valores RGB son representados en hexadecimal con formato “#RRGGBB”.

## **DESCRIPCIÓN DEL PARADIGMA**

El Paradigma Orientado a Objetos (POO) se basa en la definición de objetos, es decir, serían abstracciones de la realidad de donde se pueden extraer características y comportamientos. Un objeto es definido en una clase donde sus características y comportamientos pueden variar. El POO tiene algunos conceptos bastante importantes a la hora de hacer el análisis del diseño de la solución y construir el código, como son:

* **Objetos:** Son instancias de una clase, es decir, representaciones activas en memoria de una clase durante la ejecución de un programa.
* **Clases**: Son la definición de los atributos y métodos de un objeto. Corresponde a la implementación de un TDA.
* **Interfaz:** Solo definen los métodos de una clase que la implemente. Corresponden a la definición del TDA.
* **Atributos:** Son las características de los objetos e indican el estado del objeto, pueden ser cualquier tipo de dato primitivo, TDA u otra clase.
* **Métodos:** Son los comportamientos de los objetos y el equivalente a las funciones en el paradigma imperativo funcional. Los métodos pueden leer o escribir en los atributos del objeto o interactuar con otros.
* **Constructor:** Tipo de método especial para crear objetos con base a una clase. Se encarga también de reservar la memoria necesaria para los datos que definen al objeto.
* **Composición:** Tipo de relación entre clases donde el tiempo de vida del objeto incluido depende del que lo incluye.
* **Herencia:** Mecanismo para que una clase (hija) herede atributos y métodos de otra clase (padre).
* **Sobreescritura:** Consiste en tener métodos con mismo nombre y argumentos, pero que realizan distintas cosas según el objeto que lo invoca.
* **Sobrecarga:** Consiste en tener métodos con mismo nombre, pero argumentos distintos, ayuda a aumentar la legibilidad del programa.
* **Diagrama Lenguaje Unificado de Modelado (UML):** Representación gráfica de las relaciones entre los objetos y componentes que forman la solución con POO. Durante el análisis de la solución solo se utilizó el Diagrama de Clases.
* **Diagrama de clase:** Representación que permite determinar las clases que componen la solución y como se relacionan entre sí, indicando atributos y métodos de cada clase.

# **DESARROLLO**

## **ANÁLISIS DEL PROBLEMA**

Se identifica que para hacer los métodos necesarios para el tratamiento de imágenes simples en Java utilizando POO, hay que implementar las siguientes clases:

* **Image:** Objeto con atributos de ancho, largo (int) y lista de pixeles (Object), pudiendo ser pixbit, pixhex, pixrgb o sus versiones comprimidas.
* **Píxel:** Objeto de clase abstracta con atributos coordX, coordY y profundidad (int).
* **Pixrgb:** Objeto hijo de Píxel con atributos colorR, colorG y colorB (int)
* **Pixrgb\_comprimido:** Objeto hijo de Píxel con atributos colorRGuardado, colorGGuardado y colorBGuardado (int).
* **Pixbit:** Objeto hijo de Píxel con atributos bit (int).
* **Pixbit\_comprimido:** Objeto hijo de Píxel con atributos bitGuardado (int).
* **Pixhex:** Objeto hijo de Píxel con atributos hex (String).
* **Pixhex\_comprimido**: Objeto hijo de Píxel con atributos hexGuardado (String).

Y para implementar la ejecución por consola:

* **Menu:** Objeto con atributos salirMenu (boolean) y una imagen (Image)
* **Main:** Objeto con método constructo main que ejecuta el programa

Todo lo anterior para poder interactuar aplicando a una imagen los siguientes métodos de Image:

* **isBitmap, isHexmap, isPixmap, isCompress (Pertenencia):** Verifica si la imagen es bitmap, hexmap, pixmap o si fue comprimida respectivamente.
* **flipH, flipV, rotate90 (Modificador):** invierte los pixeles de una imagen horizontalmente, verticalmente y gira la imagen 90° a la derecha respectivamente
* **crop (Modificador):** Recorta una imagen a partir de un cuadrante definido por cuatro puntos convirtiéndolo en una nueva imagen.
* **imgRGBToHex (Modificador):** Convierte una imagen pixmap-d a hexmap-d.
* **histogram (Otros métodos):** Muestra histograma de una imagen.
* **compress (Modificador):** Comprime el color más frecuente de una imagen.
* **changePixel (Modificador):** Reemplaza un píxel de una imagen por otro del mismo tipo.
* **InvertColorBit, InvertColorRGB (Modificador):** Invierte el color de un bitmap-d y pixmap-d respectivamente
* **imageString (Otros métodos):** Entrega una cadena string de la imagen.
* **depthLayers (Otros métodos):** Entrega una lista de imágenes separados por profundidad.
* **decompress (Modificador):** Descomprime una imagen comprimida.

Se pide que Image tenga su Interfaz (TDA) y que se pueda probar los métodos implementados por consola. Ver Figura N°1 en ANEXOS para ver el Diagrama de análisis realizado. P-9. Ver Figura N°2 en ANEXOS para ver el Diagrama de Diseño luego de terminar de desarrollar el programa. P-10.

## **DISEÑO DE LA SOLUCIÓN**

Por otro lado, se distinguen algunos casos particulares para soluciones de algunos requerimientos utilizando el IDE Apache NetBeans como lo son:

**Método histogram:** Para poder crear el histograma se desarrollaron métodos auxiliares para lograr hacer los pasos de extraer color, contar color y eliminar el color de la lista de píxeles de la imagen. Con base a lo anterior se fue formando la lista de sublistas, donde cada sublista es de solo enteros, siendo los primeros valores los colores y el último valor de la lista la cantidad asociada. El ciclo para cuando la lista de píxeles este vacía. Ver la Figura N°3, Figura N°4 y Figura N°5 en ANEXOS para poder ver la salida de histogram con imágenes pixmap-d, bitmap-d y hexmap-d. P.10-11.

**Método imgRGBToString**: Para implementar este método se utilizó un método auxiliar que regresa el equivalente de un número a string hexadecimal, luego para cada color RGB se obtuvo la parte entera y resto de cada valor dividido en 16 para transformarlo a string con el método anterior. Por medio de concatenaciones se crea el equivalente a string del color RGB para pasarlo como argumento al constructor de Pixhex, que serían los nuevos píxeles de la imagen. Ver la Figura N°6 en ANEXOS para observar cómo es la transformación de colores RGB a string hexadecimal de un píxel de imagen pixmap-d. P.11.

**Método crop:** Para poder implementar el crop se instancia un ArrayList<Object> para contener a aquellos píxeles de la imagen cuyas coordenadas x e y estuvieran dentro del rango definido por los argumentos de entradas del método. Luego de “filtrar” los píxeles se vio la necesidad de implementar un método auxiliar que cambiara las coordenadas de los nuevos píxeles para qué comenzarán desde las coordenadas (0,0) a las coordenadas (ancho-1, largo-1) para que así la imagen tenga las coordenadas bien colocadas. Ver figura N°7 en ANEXOS para observar cómo cambian las coordenadas (x, y) de una imagen 4x3 tras aplicar crop. P.12.

## **ASPECTOS DE IMPLEMENTACIÓN**

Para este proyecto es necesario utilizar algún IDE de Java, como, por ejemplo, Netbeans o inteliJ IDEA. En este caso se utilizó Apache Netbeans en su versión 15 junto con un JDK versión 11.0.17 y un archivo .bat para compilar el programa. No se usaron bibliotecas externar de Java y el programa fue ejecutado por CMD con sistema operativo Windows 10.

### **EJEMPLOS DE USO**

Se debe verificar si tiene el JDK / JRE instalado en su versión 11 y que se tenga la carpeta SRC que contenga en su interior otra carpeta llamada codigo\_fuente\_21090869\_CastroVenegas y un archivo .bat llamado “script.bat”. Para ejecutar el programa puede apretar doble clic en él o, por otra parte, puede ejecutar el archivo .bat en el CMD con las mismas líneas de comando del script.bat (estando en la ruta de la carpeta SRC). Luego sería cosa de seguir las instrucciones que se muestran en pantalla.

Dentro del menú principal puede elegir entre crear una imagen, modificar una imagen, visualizar una imagen, recargar una imagen ya diseñada o salir del programa. Al comienzo del programa se tendría una imagen Bitmap de 2x3 cargada en memoria. Durante la ejecución del programa solo se usará una imagen modificando sus atributos según el método aplicado. Cabe señalar que la entrada de color de una hexmap-d sigue el formato #RRGGBB, todo en mayúsculas. Para ver algunas pruebas con el menú interactivo, ver en ANEXOS Figura N°8. P.12-15.

### **RESULTADOS ESPERADOS**

Se espera crear una simulación de tratamiento de imágenes simple, donde el programa ejecutado por consola sea funcional en su totalidad, sin errores ni ambigüedades. Junto con lo anterior, se espera trabajar correctamente con clases y métodos y que estos últimos hagan su procedimiento correctamente, que las clases tengan representaciones correctas y que al ejecutar el programa se tenga un menú completamente funcional que permita utilizar los métodos solicitados.

### **POSIBLES ERRORES**

Si se siguen todas las instrucciones del menú interactivo no tendría que haber errores ni inconvenientes con el programa, para esto hay que ingresar correctamente los valores del píxel mientras se crea la imagen, en caso de ingresar mal el programa terminaría con un mensaje del menú y habría que ejecutar el programa nuevamente.

## **RESULTADOS Y AUTOEVALUACIÓN**

### **RESULTADOS**

Los resultados obtenidos fueron los esperados, ya que se logró crear todos los métodos solicitados para aplicar a la imagen. El programa es completamente funcional, y se modificó el código de tal manera que el programa continuará aún si las entradas de las opciones son erróneas, exceptuando el caso de cuando se ingresa mal el tipo de dato de un píxel de la imagen (poner letras como bit por ejemplo). Esto como resultado de múltiples pruebas con las opciones del programa y los métodos utilizando diferentes tipos de imágenes.

### **2.4.2 AUTOEVALUACIÓN**

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.

Como no se encontró errores tras las pruebas de los métodos en el programa, se considera que funcionan el 100% de las veces.

# **CONCLUSIÓN**

Tras desarrollar los métodos y ejecutar el programa por CMD se puede concluir que se cumplieron los objetivos principales, ya que fue posible aprender y utilizar POO en Java. Se aprendieron y usaron conceptos como son las clases, métodos, atributos, constructores, relaciones como composición, herencia, entre otros. Una de las complicaciones más grande para la realización del programa fue cuando llego en momento de hacer el menú con todas sus opciones y encontrar la manera de compilar el programa en CMD, puesto que se mantuvo por mucho tiempo una idea errónea de como debería de hacerse, ya sea con gradle o .bat. Dejando lo anterior de lado, no hubo complicaciones de comprensión de conceptos o problemas con las herramientas como Git o Java en general. Se puede decir que se pudieron aprender muchas herramientas y métodos nativos de Java antes desconocidos.

Sobre los diagramas de análisis y diseño, luego terminar el programa, se observó con sorpresa lo complicado que termino siendo el diseño de la solución cuando en el diagrama de análisis no se consideraba tan extenso. Era entendible en ese momento porque al comenzar el análisis, aparte de los getters, setters y los métodos de TDA image solicitados, no se veían aún claramente qué métodos auxiliares eran necesarios para la implementación de los métodos pedidos. En el diagrama de análisis incluso se terminó definiendo métodos que no fueron utilizados para el programa.

Por un lado, comparado con el Paradigma Funcional y el Paradigma Lógico, el paradigma Orientado a Objetos resulto ser el que más conceptos tenía asociados, también fue el más rápido de comprender por ser un paradigma de la familia de los imperativos al cual uno está acostumbrado a pensar. Por el otro, también resultó ser el más rápido de realizar, esto debido a que la IDE tenía la opción de crear automáticamente los constructores, herencia, interfaces, getters y setters para cada clase, ahorrando muchas líneas de código que uno en Scheme y Prolog tenía que escribir desde cero. Finalmente, tras terminar este último laboratorio con POO se espera que este conocimiento y Java sean de utilidad para futuros ramos y proyectos por venir.

# **BIBLIOGRAFÍA Y REFERENCIAS**

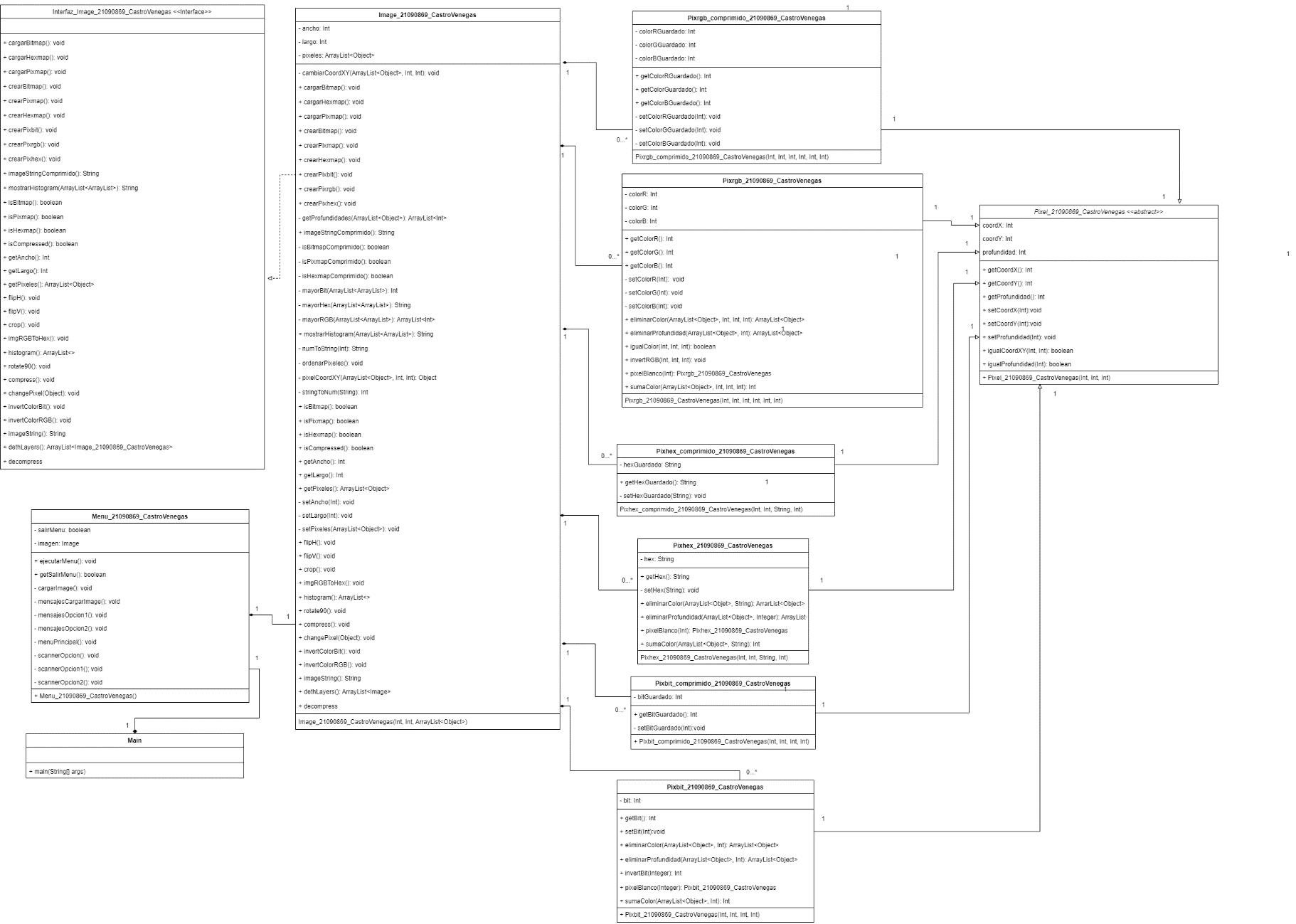
1. Gacitúa, D. (2022). “Proyecto Semestral de Laboratorio”. Paradigmas de Programación. Enunciado de Proyecto Online. Recuperado de: <https://docs.google.com/document/d/1ymOs4hi2NYhFbJDJIkgwZkM-qBd8MF5_BkwTk9d7qcg/edit>
2. Gacitúa, D. (2022). “5 - P. Orientado a Objetos”. Paradigmas de Programación. Material de clases Online. Recuperado de: <https://uvirtual.usach.cl/moodle/course/view.php?id=10036&section=20>
3. Rodríguez, A. (2022) “Documentar proyectos java con javadoc. comentarios, símbolos, tags (deprecated, param, return, etc.) (CU00680B)”. Material Entrega nº80 del curso "Aprender programación Java desde cero". Recuperado de: <https://www.aprenderaprogramar.com/attachments/article/646/CU00680B%20javadoc%20documentar%20proyectos%20java%20tags%20deprecated%20param%20return.pdf>

# **ANEXOS**

**Diagrama, Esquemático

Descripción generada automáticamenteFigura N°1: Diagrama de análisis inicial de la solución**

En el análisis inicial se consideró un diseño de una imagen compuesta por diferentes tipos de píxeles hijos de la clase píxel, también se definió una relación compuesta entre image para cada tipo de píxel, Image-Menu y Menu-Main porque una de las clases solo existirá mientras la otra exista.

**Figura N°2: Diagrama de diseño después de terminar la implementación de la solución**

Se mantuvo las relaciones del análisis y se agregaron más métodos a algunas clases.

**Figura N°3: Histograma de una imagen pixmap-d**

Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente

**Figura N°4: Histograma de una imagen bitmap-d**

Texto

Descripción generada automáticamente con confianza media Texto

Descripción generada automáticamente

**Figura N°5: Histograma de una imagen hexmap-d**

Texto, Carta

Descripción generada automáticamenteUn conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza media

**Figura N°6: Transformación interna de un pixrgb a pixhex de una imagen**

Texto

Descripción generada automáticamente

**Ver figura N°7: Como es aplicar crop a una imagen de 4x3**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Figura N°8: Interacción del programa por consola: Algunos ejemplos de uso**

Como se solicitó, una vez iniciado el programa, aparecerá cargado una imagen Bitmap de 2x3 lista para modificar. Cabe señalar que el programa no lleva tildes o signos para que se vea bien por CMD.

Texto

Descripción generada automáticamente

El usuario puede cargar otro tipo de imagen ya hecha seleccionando la opción 4

Texto

Descripción generada automáticamente

Puede crear una imagen seleccionando la opción 1 del menú. Luego tiene tres tipos de imágenes que puede crear.

Texto

Descripción generada automáticamente

Se asume que el usuario siempre ingresará valores correctos, así que si ingresa el tipo de dato del píxel incorrecto el programa terminará. Es por eso por lo que:

* Para el caso de un pixhex, su string se ingresa empezando con signo # y el valor hexadecimal en mayúsculas y todos los demás valores son enteros positivos.
* Para el caso de un bitmap, su valor bit es solo uno o cero y todos los demás valores son enteros positivos.
* Para el caso de un pixmap, todos sus valores son positivos, pero sus colores RGB están en un rango de 0 a 255.

Texto

Descripción generada automáticamente

Si por casual ingreso mal los valores de la imagen, en especial las coordenadas, no la modifique, cree una nueva imagen para reemplazar la anterior. Una vez que se tenga la imagen, puede seleccionar la opción 2 del menú para modificarla, pudiendo escoger de entre todas las opciones disponibles.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

En el caso de que el método aplicado haya modificado directamente la imagen, tendrá que seleccionar la opción 3 del menú principal para visualizar la imagen modificada. Esto fue el resultado de aplicar flipV a la imagen cargada N°1. Así se tendría que probar con todas las otras opciones disponibles.