Proyecto 2. Esteganografía por el método LSB

Pérez Romero Natalia Abigail

diciembre 2021

1. Introducción

En este documento explicaré el proceso de resolución del proyecto.

2. Definición de problema

El problema que se nos pide que resolvamos es ocultar y develar un texto en una imagen.

3. Análisis del programa

Utilicemos el proceso de solución de problemas:

3.1. Entrada

El programa debe funcionar en dos modalidades, para ocultar y para develar:
Ocultar Se debe proporcionar:

1. El nombre del archivo que contiene el texto a ocultar.
2. El nombre del archivo de imagen.
3. El nombre del archivo de imagen resultante con los datos ocultos.

Develar Se debe proporcionar:

1. El nombre del archivo con la imagen que contiene los datos ocultos.
2. El nombre del archivo en el que se guardará el texto develado.

3.2. Salida

Dependiendo de la opción seleccionada el programa debe entregar lo que se especifica. **Ocultar.** El archivo de imagen con el texto oculto. **Develar.** El archivo con el texto claro.

En la siguiente sección explicaré la elección de las herramientas para solucionar el problema.

4. Selección de la mejor alternativa

4.1. Elegir arsenal

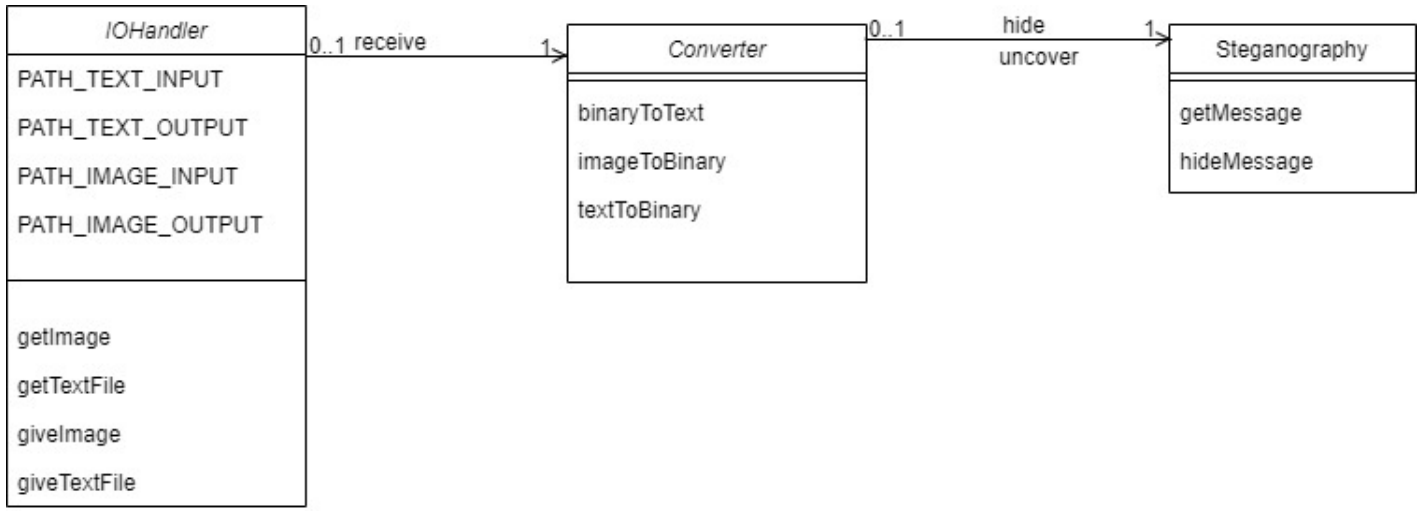
El paradigma de programación que utilizaré será orientado a objetos, por la familiaridad que tengo a este paradigma además es obligatorio en el lenguaje Java-

A pesar de que el paradigma de programación es orientado a objetos es posible dividir el proyecto en problemas más simples:

1. Manejar la entrada del programa, es decir, recuperar un archivo de texto o una imagen de un fichero.
2. Manejar la salida del programa, en este caso crear un archivo de texto o una imagen.
3. Ocultar texto en una imagen
4. Recuperar texto de una imagen
5. Convertir un texto a binario, y de binario a texto
6. Convertir una imagen a binario, y de binario a imagen

De forma que la clase IOHandler se encarga de 1 y 2 por lo que le envía el texto recuperado del archivo a Steganography que resuelve de 3 y 4, y la clase Converter apoya a Steganography al convertir los archivos al formato que requiere.

Este es el diagrama de clases:


IOHandler
PATH_TEXT_INPUT
PATH_TEXT_OUTPUT
PATH_IMAGE_INPUT
PATH_IMAGE_OUTPUT
getImage
getTextFile
giveImage
giveTextFile
0..1
receive
1
Converter
binaryToText
imageToBinary
textToBinary
0..1
hide
uncover
1
Steganography
getMessage
hideMessage

Utilice el gestor de proyectos Maven para construir el proyecto por lo que los test se hicieron la biblioteca JUnit. Por que son herramientas que he utilizado anteriormente pero que aún no domino.

4.2. Escatimar trabajo inútil

Use distintas clases que ofrece Java:
Estas clases son las que tienen que ver con manipulación de imágenes:

1. java.awt.Image
2. java.awt.image.PixelGrabber
3. javax.imageio.ImageIO
4. java.awt.image.BufferedImage
5. java.awt.image.MemoryImageSource
6. java.awt.Toolkit

PixelGrabber es la clase que me parece más importante para este proyecto porque en uno de sus métodos constructores recibe un arreglo de enteros que guardará la información de cada pixel, y en la documentación muestra un ejemplo de como realizar operaciones para manipular cada pixel independientemente y muestra como separar los canales. Las clases MemoryImageSource, Toolkit y BufferedImage son definitivamente importante para este proyecto, porque me permiten reconstruir una imagen a partir de un arreglo de enteros. Estas clases se utilizan para manejar errores de algunos de los métodos de las clases anteriores:

1. java.io.IOException
2. java.awt.AWTError

Estas clases se utilizan para manipular archivos:

1. java.io.File

La clase java.util.Scanner se utilizo para leer un archivo y la entrada por linea de comandos. La clase java.util.Arrays se utilizo para duplicar un arreglo, en vez de recorrerlo.

4.3. Elegir el modelo de datos

Usar arreglos era lo más sencillo por que las clases que use de Java ya los usaban.

4.4. Pruebas

A continuación describimos que pruebas se hicieron y porque:

1. binaryStringTest() Este test verifica una cadena que se pueda pasar de texto a binario y binario a texto sin modificaciones.

2. readTextTest() Este test verifica que la lectura de un archivo sea lo esperado.

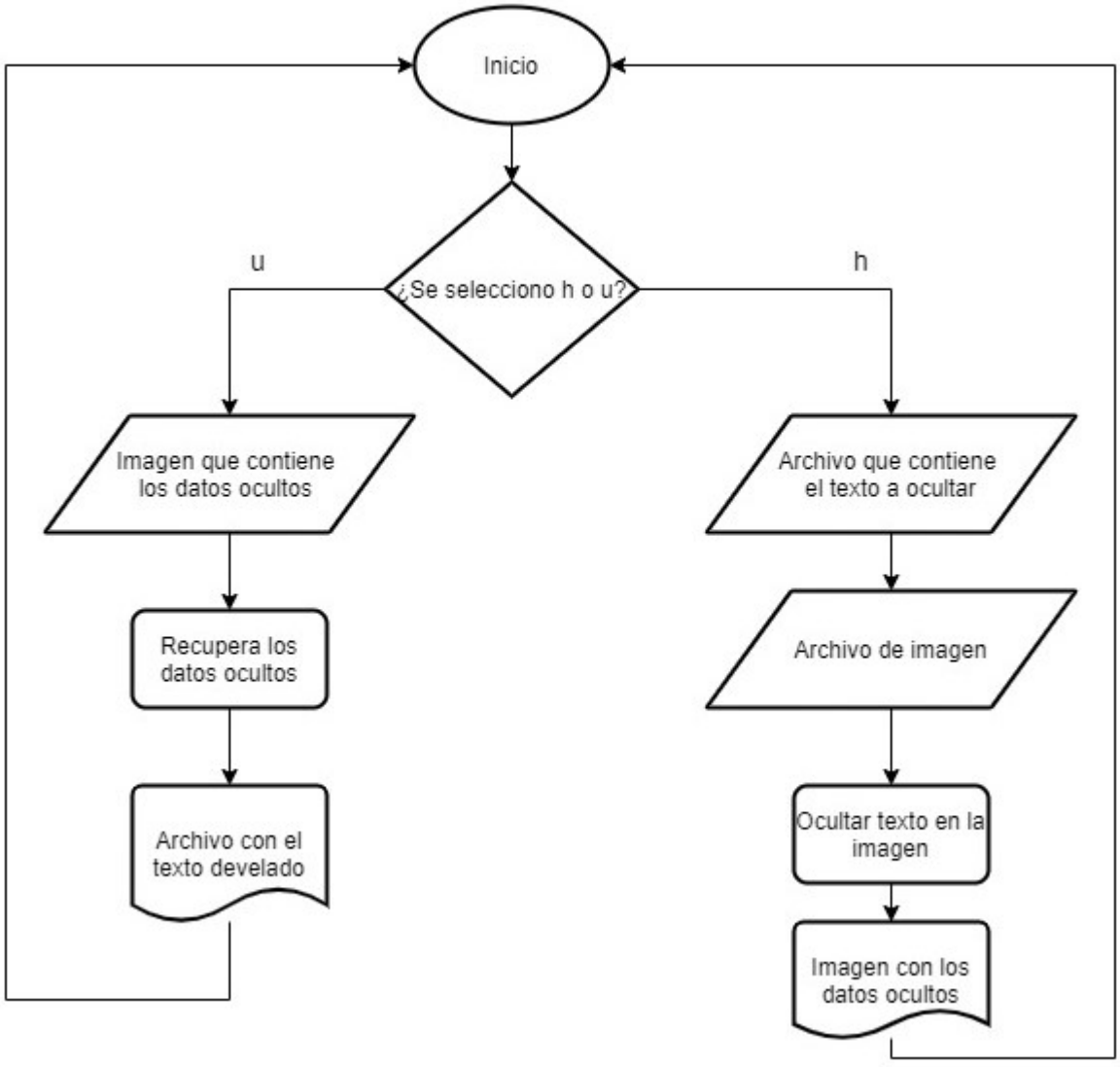
3. readWriteTextTest() Este test verifica que los métodos de escribir y leer estén correctos, comprobando que lo que se escribe un método es lo mismo que va leer el otro método.

4. processedTextTest() Este test verifica que el mensaje que se va a ocultar solo tenga caracteres validos, es decir, solo mayúsculas de la A a la Z. Utiliza código ASCII por lo que no se puede ocultar la Ñ.

5. Diagrama de flujo o pseudocódigo

6. El futuro

El proyecto me parece complicado por lo que, una vez que pueda garantizar que el proyecto es correcto, cobraría $3,000$ por que a mi parecer requiere conocimientos intermedios de un lenguaje de programación y de operaciones con bits. Me gustaría realizar más pruebas para verificar que todos los métodos son correctos. Y tal vez una interfaz gráfica.


Inicio
¿Se selecciono h o u?
u
h
Imagen que contiene los datos ocultos
Recupera los datos ocultos
Archivo con el texto develado
Archivo que contiene el texto a ocultar
Archivo de imagen
Ocultar texto en la imagen
Imagen con los datos ocultos