Proyecto Final "Juego de la Vida"

Diego Arturo Velázquez Trejo

Jueves 5 de diciembre, 2019

1 Estructura del proyecto

Para trabajar con el juego de la vida, primero realicé un diseño de las posibles clases que interactuarían en el programa. Las separé en: Métodos aleatorios, Pixel, Matriz, Excepciones, Juego de la Vida y las Reglas del Juego de la Vida.

Posteriormente, para hacer más rápida la iteración de la matriz con los pixeles de la imagen, decidí implementar el patrón de diseño de Iterable para que pudiera recorrer toda la imagen con un foreach sin necesidad de usar dos for y los contadores. La ventaja de lo antes mencionado radica en que ahora recibo directamente los pixeles de la iteración y ya puedo trabajar con ellos.

La clase matriz se compone de un arreglo bidimensional de pixeles, en donde cada objeto pixel almacena el arreglo de colores rgb de los pixels de la imagen.

La clase de JuegoVida implementa la interfaz Reglas, en donde están definidas las reglas de cambio de pixel definidas por el problema del proyecto final.

Finalmente, definí las excepciones de IndiceInvalido y ColorInvalido para poderlas usar dentro de los métodos aleatorios así como dentro de métodos del juego de la vida. La excepción de ImagenNoExiste se utiliza al momento de guardar la imagen.

2 Especificaciones del juego

Se necesita tener el compilador de gradle instalado. La imagen que se desea utilizar deberá estar en el directorio de resources dentro de src/main.

1. Corra el comando: gradle run
2. Ingrese a través del teclado el nombre de la imagen, debe notar que la ruta de la imagen se comenzará a escribir, si no sucede, de click a la imagen y escriba en el teclado nuevamente el nombre de la imagen.

3. Deje ejecutar el problema.

4. Si desea guardar la imagen, de click sobre ella en el momento que desee.

3 Diagrama UML

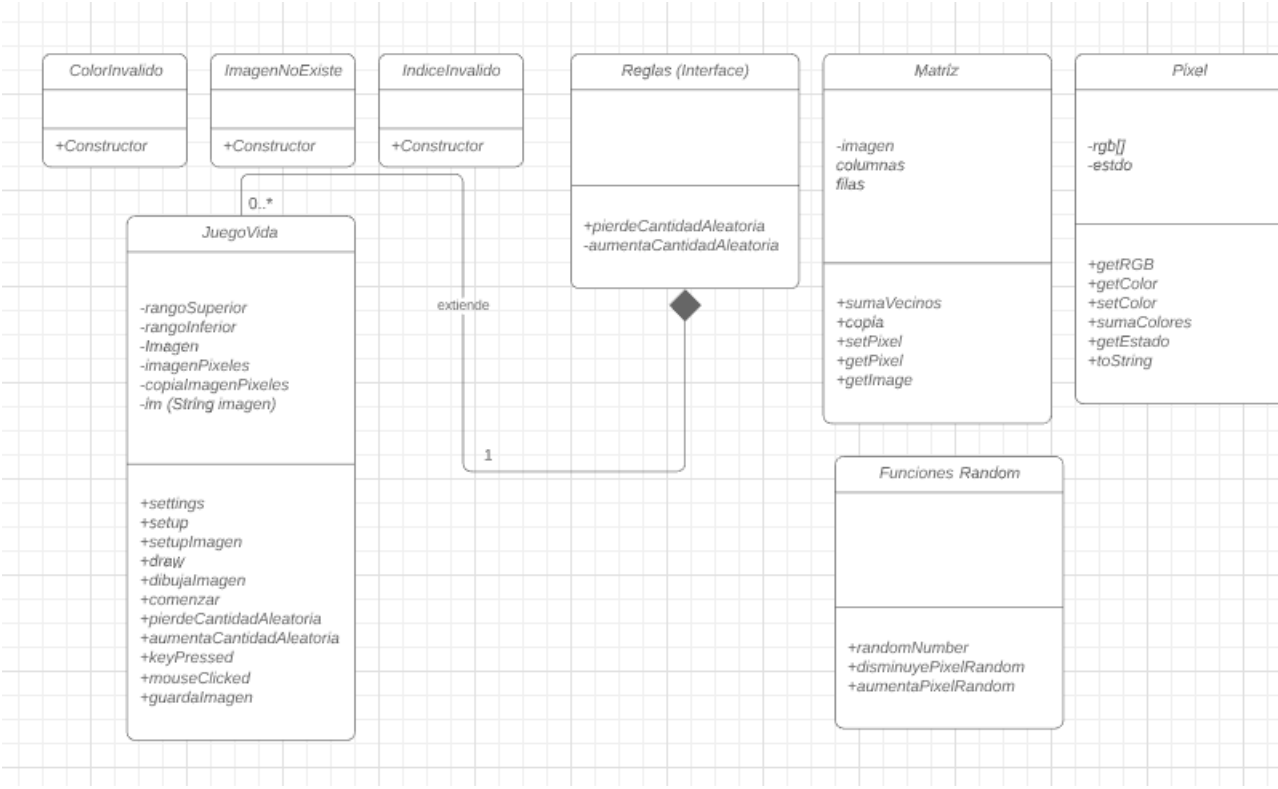

Figure 1: Imagen después del Juego de la Vida

Conclusiones: Podemos afirmar como es que en un sistema en donde los estados de objetos están interactuando, pueden llegar a hacer cosas bastante interesantes. Propusimos la matriz iterable con la finalidad de reducir la complejidad en tiempo ya que originalmente, se puede programar usando de 2 a 3 for.

Finalmente, el hecho de haber separado las funcionalidades por clase me permitió tener un mayor control sobre las funciones específicas de cada objeto y modificarlas sin alterar el flujo de todo el programa.

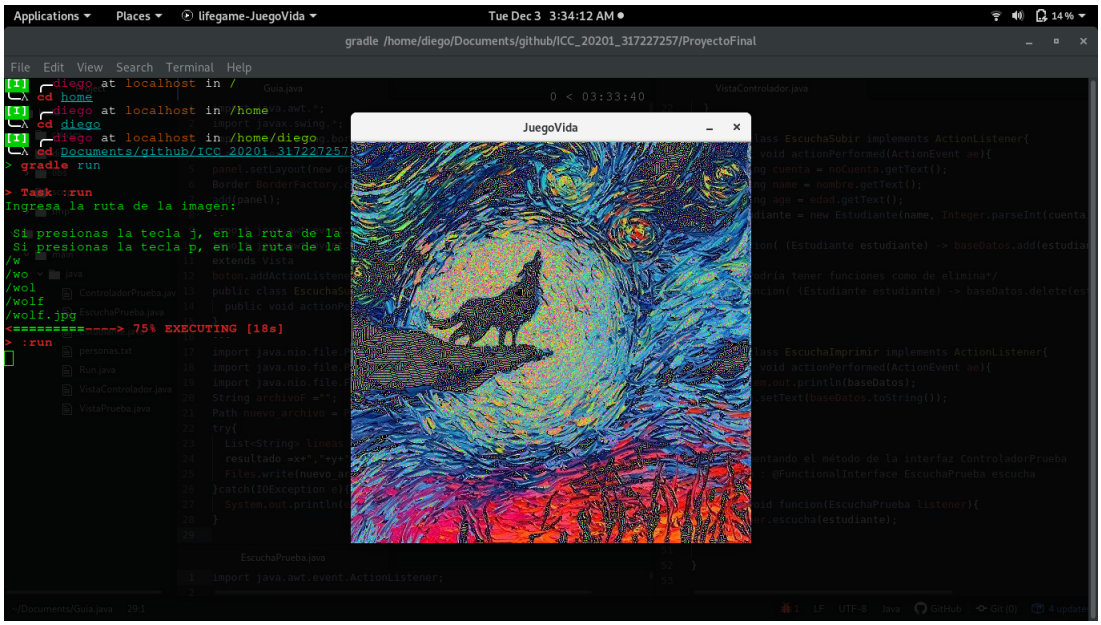

Figure 2: Imagen después del Juego de la Vida

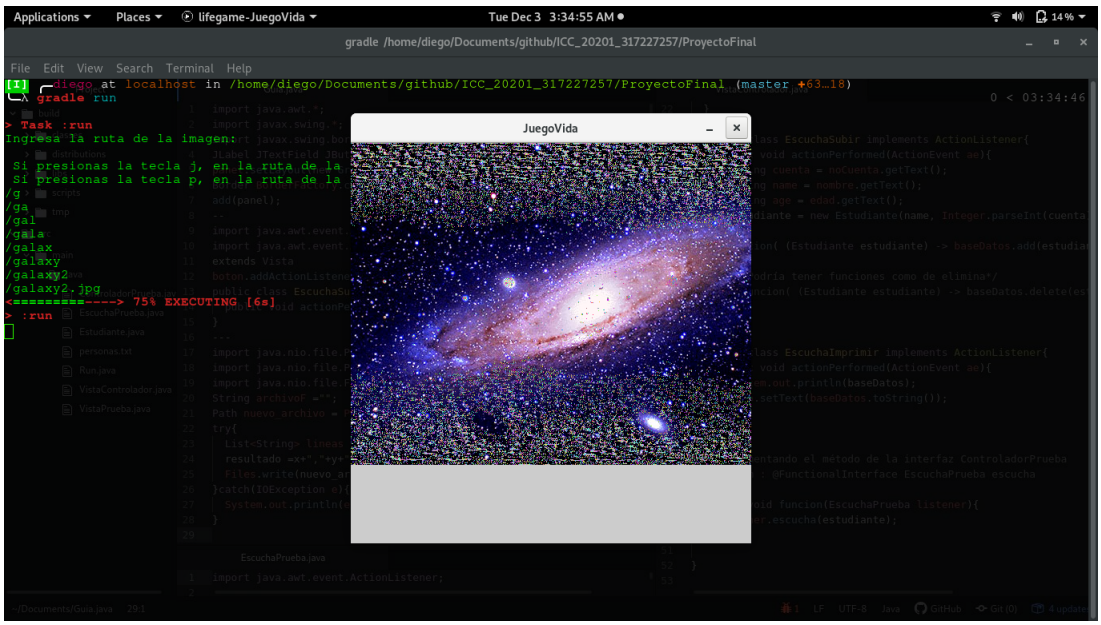

Figure 3: Imagen después del Juego de la Vida