Nombres: Andrea Bayona y David Calle

TAD VOLÚMEN

1. Conjunto mínimo de datos:

Se encuentran:
● imágenes, colección de Imagen2D, una lista que tiene en su interior las imágenes creadas.
● tamanio, entero, cantidad de imágenes que tiene la colección.

2.Operaciones:

● **Volumen():** Construye un objeto volúmen sin imágenes.
● **Volumen(nombreBase,total):** inicializa la colección de imágenes usando como criterios el nombre base y la cantidad.
● **getTamanio():** retorna el tamaño que tiene la lista de las imágenes.
● **setTamanio(int tam):**cambia el valor de tamaño por tam.
● **getImagenes():** Retorna la colección de imágenes.
● **proyeccionXPromedio():** Retorna una imagen que representa la proyección en x de la colección de imágenes usando el criterio promedio.
● **proyeccionYPromedio():** Retorna una imagen que representa la proyección en y de la colección de imagenes usando el criterio promedio.
● **proyeccionZPromedio():** Retorna una imagen que representa la proyección en z de la colección de imágenes usando el criterio promedio.
● **proyeccionXMaximo():** Retorna una imagen que representa la proyección en x de la colección de imágenes usando el criterio máximo.
● **proyeccionYMaximo():** Retorna una imagen que representa la proyección en y de la colección de imágenes usando el criterio máximo.
● **proyeccionZMaximo():** Retorna una imagen que representa la proyección en z de la colección de imágenes usando el criterio máximo.
● **proyeccionXMinimo():** Retorna una imagen que representa la proyección en x de la colección de imágenes usando el criterio mínimo.
● **proyeccionYMinimo():** Retorna una imagen que representa la proyección en y de la colección de imágenes usando el criterio mínimo.
● **proyeccionZMinimo():** Retorna una imagen que representa la proyección en z de la colección de imágenes usando el criterio mínimo.
● **proyeccionXMediana():** Retorna una imagen que representa la proyección en x de la colección de imágenes usando el criterio mediana.

● **proyeccionYMediana():** Retorna una imagen que representa la proyección en y de la colección de imágenes usando el criterio mediana.
● **proyeccionZMediana():** Retorna una imagen que representa la proyección en z de la colección de imágenes usando el criterio mediana.

TAD IMAGEN2D

1. Conjunto mínimo de datos:

● imagen, colección de colección de enteros, matriz de números en la cual cada número representa un pixel.
● formato, cadena, representa el código en el que viene dada la imágen.
● fila,entero:, representa el número de filas que tiene la lista de listas
● columna,entero, representa el número de columnas que tiene la lista de listas

2.Operaciones:

● **Imagen2D():** Inicializa una imagen sin valores.
● **Imagen2D(pNombre):**Inicializa la imagen que se encuentra en un archivo con nombre pNombre.
● **getFila():** Retorna la cantidad total de filas dentro de las imágenes.
● **setFila(tam):**cambia el valor de fila por tam.
● **getColumna():** Devuelve la cantidad total de columnas dentro de las imágenes.
● **setColumna(tam):**Permite darle el valor tam al atributo columna dentro de la clase Imagen.
● **getImagen():** retorna la matriz que representa la imagen.
● **setImagen(nuevo):**Permite cambiar la dirección por la que entra como parámetro a la cual apunta el apuntador dentro de imagen.
● **getFormato():** Devuelve el formato en el que se encuentra la imagen.
● **setFormato(nuevo):** Permite darle el valor de nuevo al formato que tiene la imagen,. el formato queda como nuevo.
● **imprimirImagen():** Permite imprimir la matriz que representa la imagen.
● **exportarImagen(nom_arch):** exporta la imagen en formato pgm.
● **buscarIntensidad(intensidad, lista):** Devuelve la posición en la que se encuentra la intensidad a buscar.
●**cargarHuffman(nombreArchivo):**Carga un archivo Huffman con nombre nombreArchivo y devuelve si se pudo o no cargar.
●**exportarHuffman(nom_arch):** Exporta el Huffman recibiendo como parámetro el nombre del archivo en el que este se encuentra.

●**calcularListaIntensidades():** Calcula las intensidades y las guarda en un lista, la cual retorna.
●**cargarArchivo(nombre):** Carga el archivo con el nombre que recibe como parámetro y retorna verdadero o falso dependiendo de sí logró cargarlo

TAD NODO

1.Conjunto Mínimo de Datos:

- contenido, tipo plantilla(T), representa la información que está almacenada en el nodo.
- hijoDerecho, apuntador a Nodo, representa al Nodo hijo derecho del nodo.
- hijoIzquierdo, apuntador a nodo, representa al Nodo hijo izquierdo del nodo.

2.Operaciones:

- **getContenido(),** retorna el valor del contenido del nodo.
- **setContenido(pContenido),** cambia el valor del contenido, por pContenido.
- **getHijoIzquierdo(),** retorna el apuntador al Nodo hijo izquierdo del Nodo.
- **getHijoDerecho(),** retorna el apuntador al nodo hijo derecho del Nodo.
- **setHijoIzquierdo(nuevo),** cambia el valor del apuntador del hijo izquierdo por nuevo.
- **bool esHoja(),** retorna si el nodo no tiene hijos.

TAD ÁRBOL

1.Conjunto Mínimo de Datos:

cabeza, apuntador a nodo, contiene un apuntador a la cabeza del árbol.

2.Operaciones:

- **Arbol(),** crea un árbol cuya cabeza es vacía
- **getCabeza(),** retorna la cabeza del árbol
- **setCabeza(c),** cambia la cabeza por c.
- **eliminarArbol(),** borra cada uno de los nodos del árbol.
- **operator(in):** retorna si el contenido de la cabeza del árbol es mayor al contenido de la cabeza del árbol in.

TAD INTENSIDAD

1.Conjunto Mínimo de Datos:

- Valor, entero que representa el valor de píxel más grande de la imagen, es decir la intensidad máxima de la imagen.
- Intensidad, entero que representa el número de veces que se repite un valor.

2.Operaciones:

- **Intensidad(valor,frecuencia):** Inicializa la intensidad con su respectivo valor y frecuencia.
- **getValor():** Devuelve el valor de la intensidad.
- **setValor(pValor):** Cambia el valor de la intensidad por pValor.
- **getFrecuencia():** Devuelve el valor de la intensidad.
- **setFrecuencia(pFrecuencia):** Cambia el valor de la intensidad por pFrecuencia.
- **operator(in):** determina si la intensidad in es menor que la frecuencia de la intensidad actual.

TAD VÉRTICE

1.Conjunto mínimo de datos:

1. Contenido, tipo plantilla(T), representa la información guardada en el vértice.

2.Operaciones:

- **Vertice()**,crea un vértice;
- **Vertice(pContenido)**, crea un vértice con contenido pContenido.
- **getContenido()**, retorna el contenido del vértice.
- **void setContenido(nuevo)**, cambia el contenido del vértice por nuevo.
- **operator(a)**,permite comparar entre el vértice a que llega como parámetro y el vértice que hay dentro de este tad, para así retornar el menor dato.

TAD ARISTA

1.Conjunto mínimo de datos:

1. Valor, representa el inicio de las aristas;
2. finArista, representa un vértice .

2.Operaciones:

- **Arista()**, se encarga de construir una arista.
- **Arista(contenido,nuevo)**, se encarga de construir una arista teniendo en cuenta los parámetros recibidos.
- **getFinArista()**, se encarga de retornar un vértice;
- **getValor()**, retorna el valor del vértice inicio;
- **setValor(val)**, asigna el val recibido al valor del vértice inicio;
- **agregarArista(vertice)**, agrega el vértice vértice a la lista de aristas.
- **setFinArista(vertice)**, asigna el vértice recibido al vértice ya creado;
- **operator(a)**,permite comparar entre la arista que llega como parámetro y la arista que hay dentro de este tad, para así retornar el menor dato.

TAD GRAFO

1.Conjunto mínimo de datos:

1. Vértices, lista de Aristas del grafo;

2.Operaciones:

- **Grafo();**
- **agregarVertice (nuevo)**, agrega un nuevo vértice a lista de vértices del grafo.
- **agregarArista(inicio, fin)**, agrega a las aristas la arista con inicio inicio y fin fin.
- **eliminarArista (inicio,fin,peso)**, elimina la arista desde el inicio, fin y con su respectivo peso que entran como parámetros y retorna verdadero si se pudo eliminar.
- **eliminarVertice(nuevo)**, elimina el vértice que entra como parámetro y retorna verdadero si se pudo eliminar.
- **tamanio(),** retorna el tamaño del grafo.
- **esTrivial(),** retorna verdadero si solo tiene un vértice y ninguna arista
- **esRegular(),**retorna verdadero si cada vértice tiene el mismo número de vecinos
- **esVacio(),**retorna verdadero si el grafo está vacío.
- **encontrarVertice(vertice)**,Busca un vértice que entra como parámetro y lo retorna la dirección en la que se encuentra.
- **buscarVertice(vertice)**, retorna verdadero si encuentra el vértice en el grafo

- **orden(),**ordena la lista de aristas y devuelve el dato menor.
- **dfs(origen,visitados),** recorre el grafo desde el parámetro origen en profundidad e imprime el recorrido.
- **bfs(origen),** recorre el grafo desde el parámetro origen por anchura y lo imprime.
- **numeroVerticesImpar()**,retorna la cantidad de vértices impares.
- **caminoEuleriano(visitados)**,Recorre cada arista sin repetirse y devuelve si fue exitosa la operación.
- **visitarLosNodos(visitados)**,Recorre los visitados para indicar el camino y devuelve si fue exitosa la operación.
- **buscarAdyacentes(origen),** busca los vecinos del vértice origen y los agrega a un conjunto para retornarlo.
- **Dijkstra(origen),** Halla el camino mínimo entre todos los vértices iniciando en origen, retorna una colección de elementos que representan el camino junto con la distancia.
- **darMatrizAdyacencia(),** retorna una matriz que representa la matriz de adyacencia del grado
- **FloydWarshall(),** encuentra el peso del camino mínimo entre cada par de vértices, retorna una matriz.
- **darElementosFuertementeConectados(),** retorna los elementos fuertemente conectados del grafo.
- **darPosicionVertice(vertice),** Permite conocer la posición de un vértice y la devuelve.

REGIÓN

1.Conjunto mínimo de datos:

1. **x,** Coordenada en x.
2. **y,** Coordenada en y.
3. **color,** Color del pixel
4. **identificador,** Número con el cual se identifica la región.
5. **colorNuevo**, El nuevo color con el cual se pinta.

2.Operaciones:

- **Region(x,y,color, id),** Permite construir una nueva región, teniendo en cuenta la x,y, el color, y el id que entran como parámetros.
- **Region(),** Permite conocer construir una nueva región por defecto.

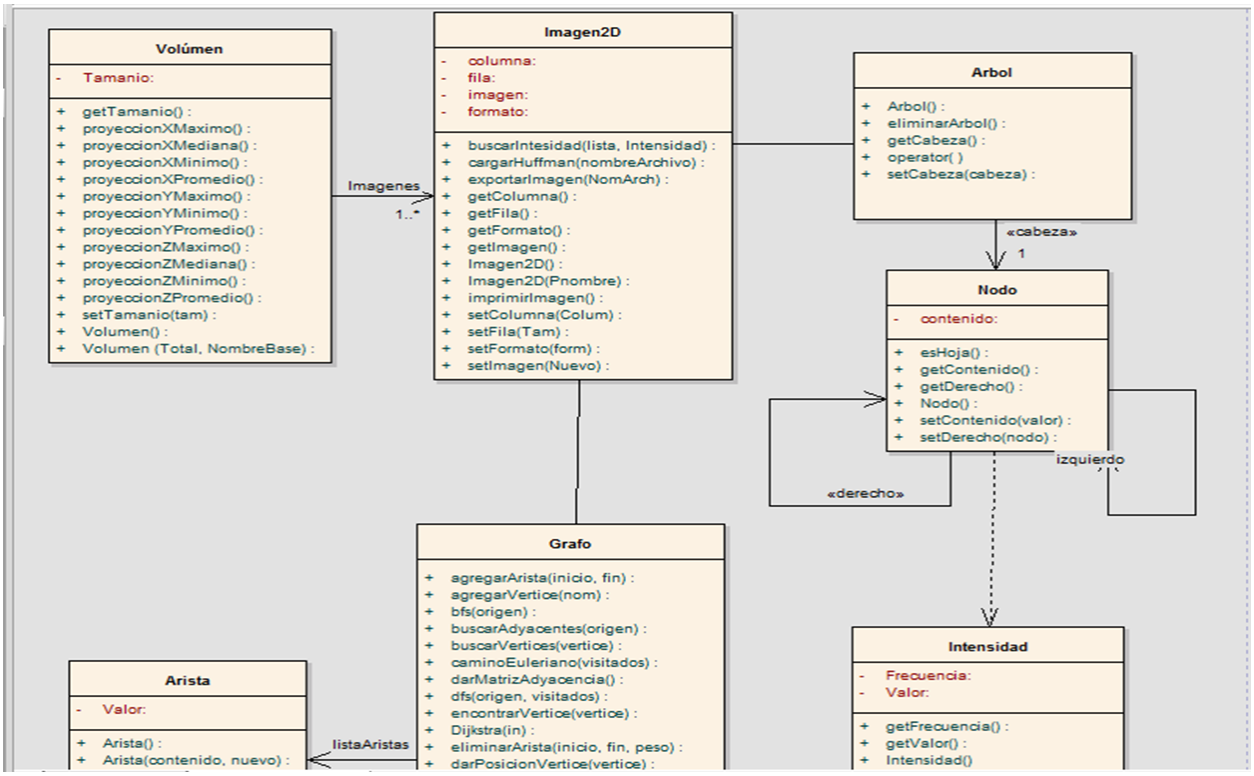
- **getColor(),** Permite conocer y retorna el color del pixel.
- **getX(),**Permite conocer y retorna la coordenada X de la región.
- **getY(),**Permite conocer y retorna la coordenada Y de la región.
- **getColorNuevo(),**Permite conocer y retorna el color nuevo.
- **getIdentificador(),**Permite conocer y retorna el identificador.
- **setColor(pColor),** Permite cambiar el valor del atributo, teniendo en cuenta el que recibe como parámetro.
- **setColorNuevo(pColorN),**Permite cambiar el valor del atributo, teniendo en cuenta el que recibe como parámetro.
- **setX(pX),**Permite cambiar el valor del atributo, teniendo en cuenta el que recibe como parámetro.
- **setY(pY)**Permite cambiar el valor del atributo, teniendo en cuenta el que recibe como parámetro.
- **setIdentificador(pId),**Permite cambiar el valor del atributo, teniendo en cuenta el que recibe como parámetro.
- **operator(r),**Permite comparar los valores de la región

Otras funciones Importantes:

- **reverse(s):** Crea una nueva cadena con la palabra invertida.
- **itoa(n,s):** Permite convertir un entero en carácter, recibiendo el entero que debe transformar y el carácter donde guardará el dato convertido
- **preOrden(nodo):** Imprime el valor de la frecuencia y la intensidad en pre-orden
- **hoffman(lista):** Crea un árbol de huffman a partir de la lista de intensidad lista.
- **codificarHoffman(nodo,valor,cadena):** retorna la representaciòn del valor valor del árbol de huffman.
- **decodificarHoffman(nodo,lectura):** Guarda en un árbol la representación de un arbol de huffman guardada en un archivo.
- **decodificarValor(nodo,codificacion):** decodifica la cadena codificación en el arbol.
- **codificarValor(nodo,valor,retorno):** retorna la codificación de valor en un arbol de huffman.
- **generarTablaDistancias(nombre,grafo),** Permite calcular la distancia entre aristas y generar una tabla apartir de la misma.
- **distanciaMaxima(grafo)**Permite calcular la distancia Máxima que hay en cuanto al recorrido y generar una tabla a partir de la misma.
- **distanciaMinima(grafo)**Permite calcular la distancia Mínima que hay en cuanto al recorrido y generar una tabla a partir de la misma.

● **cargarGrafo(imagen, grafo),** Permite cargar el grafo teniendo en cuenta las aristas ingresaads por el usuario
● **validarColores(grafo, colores),** Se encarga de validar si un vértice está coloreado con el mismo color que sus adyacentes.
● **combinatorioColores(posicion,valor, grafito,colores),** Permite encontrar los colores adecuados para colorear cada vértice.
● **colorearGrafo(grafito,numColores),**Permite colorear los vétices.

DIAGRAMA



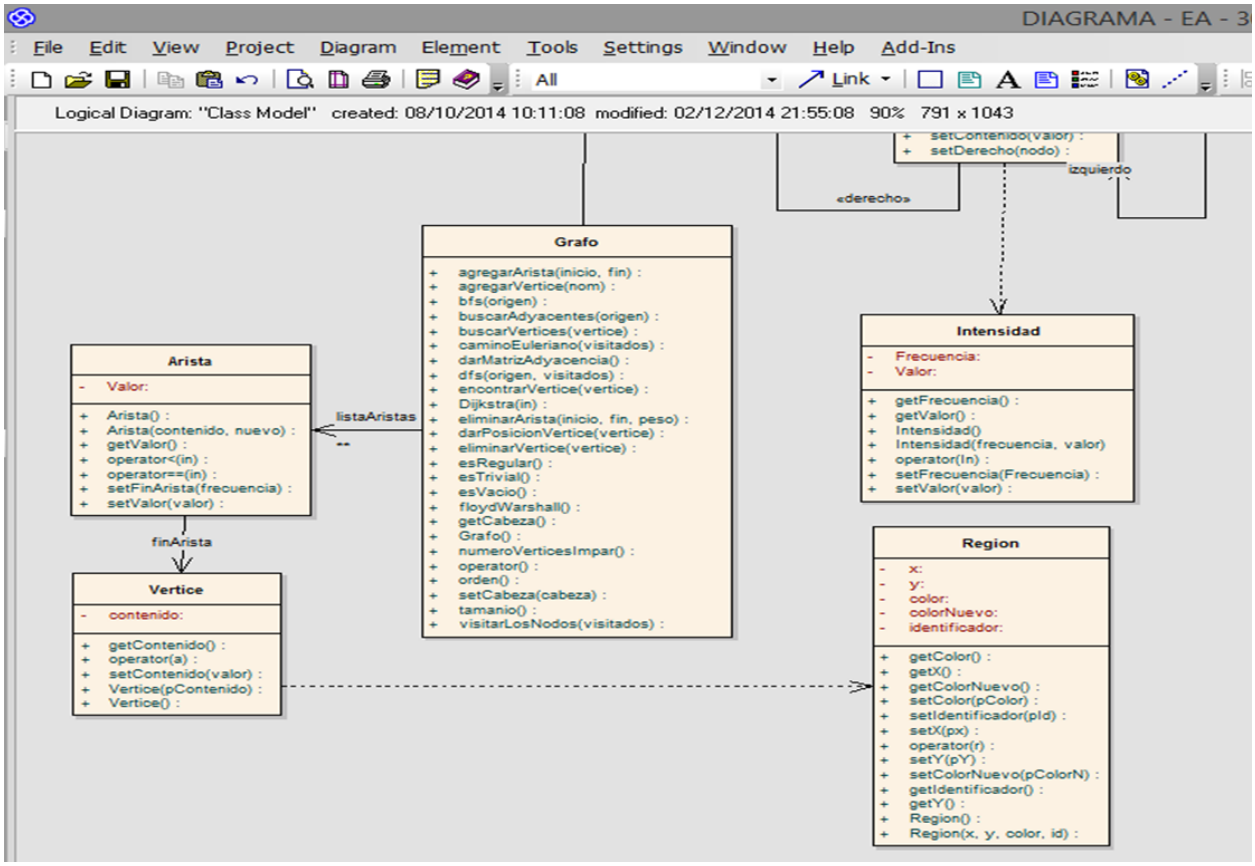


DIAGRAMA - EA - 3
File Edit View Project Diagram Element Tools Settings Window Help Add-Ins
All
Link
Logical Diagram: "Class Model" created: 08/10/2014 10:11:08 modified: 02/12/2014 21:55:08 90% 791 x 1043
+ setContenido(valor) :
+ setDerecho(nodo) :
izquierdo
«derecho»
Grafo
+ agregarArista(inicio, fin) :
+ agregarVertice(nom) :
+ bfs(origen) :
+ buscarAdyacentes(origen) :
+ buscarVertices(vertice) :
+ caminoEuleriano(visitados) :
+ darMatrizAdyacencia() :
+ dfs(origen, visitados) :
+ encontrarVertice(vertice) :
+ Dijkstra(in) :
+ eliminarArista(inicio, fin, peso) :
+ darPosicionVertice(vertice) :
+ eliminarVertice(vertice) :
+ esRegular() :
+ esTrivial() :
+ esVacio() :
+ floydWarshall() :
+ getCabeza() :
+ Grafo() :
+ numeroVerticesImpar() :
+ operator() :
+ orden() :
+ setCabeza(cabeza) :
+ tamanio() :
+ visitarLosNodos(visitados) :
Arista
- Valor:
+ Arista() :
+ Arista(contenido, nuevo) :
+ getValor() :
+ operator<(in) :
+ operator==(in) :
+ setFinArista(frecuencia) :
+ setValor(valor) :
listaAristas
finArista
Vertice
- contenido:
+ getContenido() :
+ operator(a) :
+ setContenido(valor) :
+ Vertice(pContenido) :
+ Vertice() :
Intensidad
- Frecuencia:
- Valor:
+ getFrecuencia() :
+ getValor() :
+ Intensidad()
+ Intensidad(frecuencia, valor)
+ operator(In) :
+ setFrecuencia(Frecuencia) :
+ setValor(valor) :
Region
- x:
- y:
- color:
- colorNuevo:
- identificador:
+ getColor() :
+ getX() :
+ getColorNuevo() :
+ setColor(pColor) :
+ setIdentificador(pId) :
+ setX(px) :
+ operator(r) :
+ setY(pY) :
+ setColorNuevo(pColorN) :
+ getIdentificador() :
+ getY() :
+ Region() :
+ Region(x, y, color, id) :