

DESCRIPCIÓN DE CLASES DEL DOMINIO

Compressor_Controller

Clase encargada de comunicar los compresores con otras capas (presentación y persistencia). Proporciona métodos de entrada y salida además realizar el cálculo de las estadísticas.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
compressor	No	-	Objeto compresor.
inputFile	No	-	File referente al archivo origen.
outputFile	No	-	File referente al archivo destino.
in	No	-	Buffer de lectura.
out	No	-	Buffer de escritura.
time	No	-	Tiempo transcurrido durante la compresión.

MÉTODOS:

Nombre: selectFiles

Parámetros:

- **inputPath** Path del fichero origen.
- **outputPath** Path del fichero destino

Descripción: Establece cuáles serán los ficheros de origen y destino.

Resultado: Se han establecido los archivos de origen y destino.

Nombre: getCompressedName

Parámetros: **fileName** Path del archivo original.

Descripción: Proporciona el path de un archivo destino en base a su archivo origen, con el objetivo que vayan a parar ambos al mismo directorio, con el mismo nombre, pero diferente extensión.

Resultado: Devuelve path del fichero destino.

Nombre: getCompressedName

Parámetros: **file** Fichero que referencia al fichero original.

Descripción: Proporciona el path de un archivo destino en base a su archivo origen, con el objetivo que vayan a parar ambos al mismo directorio, con el mismo nombre, pero diferente extensión

Resultado: Devuelve path del fichero destino.

Nombre: startCompression

Parámetros: **inputPath** Path del archivo original.

Descripción: Inicia la compresión del archivo referenciado por el path inputPath hacia un nuevo archivo en la misma ubicación que la indicada por inputPath.

Resultado: Inicia la compresión, y una vez finaliza recoge las estadísticas.

Nombre: startCompression

Parámetros:

- **inputPath** Path del archivo original.
- **outputPath** Path del directorio donde se creará el archivo comprimido.

Descripción: Inicia la compresión del archivo referenciado por el path inputPath hacia un nuevo archivo en outputPath.

Resultado: Inicia la compresión, y una vez finaliza recoge las estadísticas.

Nombre: getTime

Parámetros: -

Descripción: Getter del tiempo transcurrido en milisegundos.

Resultado: Retorna el tiempo transcurrido en milisegundos.

Nombre: getOriginalSize

Parámetros: void

Descripción: Getter del tamaño en Bytes del fichero original.

Resultado: Retorna el tamaño en Bytes del fichero original.

Nombre: getCompressedSize

Parámetros: void

Descripción: Getter del tamaño en Bytes del fichero comprimido.

Resultado: Retorna el tamaño en Bytes del fichero comprimido.

Nombre: getCompressionRatio

Parámetros: void

Descripción: Getter del ratio de compresión absoluto de la compresión realizada.

Resultado: Retorna el ratio de compresión absoluto de la compresión realizada.

Nombre: readByte

Parámetros: void

Descripción: Lee un byte del fichero origen.

Resultado: Retorna un entero que contiene el byte leído o -1 si no había nada que leer.

Nombre: readNBytes

Parámetros: byte[] **word** Cadena de bytes sobre la que se introducirá la lectura.

Descripción: Lee N bytes del fichero origen en una cadena de bytes que se le pasa por parámetro.

Resultado: Retorna la cantidad de bytes leída o -1 si no había nada que leer.

Nombre: closeReader

Parámetros: void

Descripción: Cierra el buffer de lectura.

Resultado: Se cierra el buffer de lectura.

Nombre: readAllBytes

Parámetros: void

Descripción: Lee todos los bytes del fichero origen y los guarda en una cadena.

Resultado: Retorna una cadena de bytes con todos los bytes del fichero origen.

Nombre: readFileString

Parámetros: void

Descripción: Lee todos los caracteres del fichero origen y los guarda en un String.

Resultado: Retorna un String con todos los caracteres del fichero origen.

Nombre: writeByte
Parámetros: byte **B**
Descripción: Escribe un byte en fichero de salida.
Resultado: Se ha escrito un byte en fichero de salida.

Nombre: writeBytes
Parámetros: byte[] **word** Cadena de bytes que se desea escribir en el fichero de salida.
Descripción: Escribe una cadena de bytes en el fichero de salida.
Resultado: Se ha escrito una cadena de bytes en fichero de salida.

Nombre: closeWriter
Parámetros: void
Descripción: Cierra el buffer de escritura.
Resultado: Se cierra el buffer de escritura.

Decompressor_Controller

Clase encargada de comunicar los descompresores con otras capas (presentación y persistencia). Proporciona métodos de entrada y salida además realizar el cálculo de las estadísticas.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
decompressor	No	-	Objeto descompresor.
inputFile	No	-	File referente al archivo origen.
outputFile	No	-	File referente al archivo destino.
in	No	-	Buffer de lectura.
out	No	-	Buffer de escritura.
time	No	-	Tiempo transcurrido durante la compresión.

MÉTODOS:

Nombre: selectFiles

Parámetros:

- **inputPath** Path del fichero origen.
- **outputPath** Path del fichero destino

Descripción: Establece cuáles serán los ficheros de origen y destino.

Resultado: Se han establecido los archivos de origen y destino.

Nombre: getCompressedName

Parámetros: **fileName** Path del archivo original.

Descripción: Proporciona el path de un archivo destino en base a su archivo origen, con el objetivo que vayan a parar ambos al mismo directorio, con el mismo nombre, pero diferente extensión.

Resultado: Devuelve path del fichero destino.

Nombre: getCompressedName

Parámetros: **file** Fichero que referencia al fichero original.

Descripción: Proporciona el path de un archivo destino en base a su archivo origen, con el objetivo que vayan a parar ambos al mismo directorio, con el mismo nombre, pero diferente extensión

Resultado: Devuelve path del fichero destino.

Nombre: startDecompression

Parámetros: **inputPath** Path del archivo original.

Descripción: Inicia la descompresión del archivo referenciado por el path inputPath hacia un nuevo archivo localizado en el mismo directorio que el original.

Resultado: Inicia la compresión, y una vez finaliza recoge las estadísticas.

Nombre: startDecompression

Parámetros:

- **inputPath** Path del archivo original.
- **outputPath** Path del directorio donde se creará el archivo comprimido.

Descripción: Inicia la descompresión del archivo referenciado por el path inputPath hacia un nuevo archivo en outputPath.

Resultado: Inicia la compresión, y una vez finaliza recoge las estadísticas.

Nombre: getTime

Parámetros: -

Descripción: Getter del tiempo transcurrido en milisegundos.

Resultado: Retorna el tiempo transcurrido en milisegundos.

Nombre: readByte

Parámetros: void

Descripción: Lee un byte del fichero origen.

Resultado: Retorna un entero que contiene el byte leído o -1 si no había nada que leer.

Nombre: readNBytes

Parámetros: byte[] **word** Cadena de bytes sobre la que se introducirá la lectura.

Descripción: Lee N bytes del fichero origen en una cadena de bytes que se le pasa por parámetro.

Resultado: Retorna la cantidad de bytes leída o -1 si no había nada que leer.

Nombre: closeReader

Parámetros: void

Descripción: Cierra el buffer de lectura.

Resultado: Se cierra el buffer de lectura.

Nombre: readAllBytes

Parámetros: void

Descripción: Lee todos los bytes del fichero origen y los guarda en una cadena.

Resultado: Retorna una cadena de bytes con todos los bytes del fichero origen.

Nombre: writeByte

Parámetros: byte **B**

Descripción: Escribe un byte en fichero de salida.

Resultado: Se ha escrito un byte en fichero de salida.

Nombre: writeBytes

Parámetros: byte[] **word** Cadena de bytes que se desea escribir en el fichero de salida.

Descripción: Escribe una cadena de bytes en el fichero de salida.

Resultado: Se ha escrito una cadena de bytes en fichero de salida.

Nombre: closeWriter

Parámetros: void

Descripción: Cierra el buffer de escritura.

Resultado: Se cierra el buffer de escritura.

Decompressor

Clase con métodos abstractos para implementar un descompresor. Contiene una variable que hace referencia a su controlador para poder comunicarse con otras capas.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
controller	No	-	Controlador del compresor.

MÉTODOS:

Nombre: decompress ABSTRACTA

Parámetros: -

Descripción: Función encargada de descomprimir el archivo proporcionado por la controladora y escribirlo en el archivo de salida también a través de la controladora.

Nombre: getExtension ABSTRACTA

Parámetros: -

Descripción: Retorna la extensión del archivo original (actualmente comprimido).

Nombre: setController

Parámetros: Decompressor_Controller controller

Descripción: Setter de la variable controller.

Resultado: Set de la variable controller.

Compressor_LZ78

Extensión de la clase **Compressor** mediante el algoritmo LZ-78.

Para comprimir mediante el algoritmo LZ78 vamos generando un diccionario de pares (véase la clase **Pair**) guardado en la variable `comp_file`. Esta implementación se ha llevado a cabo con un árbol de búsqueda que se va generando durante la compresión (véase la clase **Tree**). Desde la función `compress()`, vamos haciendo lecturas byte a byte del archivo comprimido. Para cada byte leído se realiza la compresión del mismo. Dependiendo de la situación del compresor y la disposición de los bytes del fichero original, el algoritmo puede presentar 2 estados diferentes:

- **Estado de repetición:** A cada byte leído, si la cadena formada por los anteriormente leídos (y no insertados) con el actual está presente en el árbol de búsqueda, se solicita otro byte más para aumentar la cadena considerada.
- **Estado de nueva aparición:** Repetido el estado anterior suficientes veces, eventualmente llegaremos a una situación en la que la cadena de bytes considerada no está presente en el árbol (no había sido vista antes). Cuando sucede esto, se crea un nuevo par, formado por el índice que referencia a la entrada en el diccionario que representa la cadena inspeccionada en la anterior iteración y un offset igual al último byte leído. Este par es introducido en el diccionario, y la cadena equivalente se mostrará presente en el árbol de búsqueda.

La variación entre los dos estados la controlamos con el retorno booleano de la función `compress`. Mientras retorna `false`, significa que están habiendo repeticiones. En el momento que la cadena inspeccionada no está presente, realiza las inserciones necesarias y retorna `true`, indicando que se debe resetear la búsqueda en el árbol, de nuevo a la raíz, ya que mientras hay repeticiones, va profundizando en el árbol (en más detalle a continuación).

Coste de las búsquedas: Respecto al coste de las búsquedas, el árbol utilizado (**Tree**) permite profundizar byte a byte en el, sin tener que realizar una búsqueda de una cadena completa cada vez. Debido al funcionamiento de este algoritmo, si estamos buscando una cadena de N bytes, es porque previamente hemos buscado la cadena correspondiente a los primeros $N-1$ bytes y el resultado de la búsqueda ha sido positivo, así que no tendría sentido volver a recorrer los $N-1$ niveles del árbol para consultar si el siguiente byte está presente. Por lo tanto, cada búsqueda es constante, $O(255)$.

CARDINALIDAD

Dado que la clase funciona con la clase **Tree**, y esta utiliza elementos estáticos, sólo debe existir una instancia de la misma. Por lo tanto, la cardinalidad es 1.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
<code>comp_file</code>	No	<code>ArrayList<>()</code> vacío	Archivo sobre el que se escribe la compresión actual.
<code>next_index</code>	No	0	Siguiente índice que se debe utilizar como referencia en el diccionario.

MÉTODOS:

Nombre: getExtension

Parámetros: void

Descripción: Se retorna el valor de la extensión personal de los archivos comprimidos del compresor.

Resultado: String ".lz78"

Nombre: compress

Parámetros:

- byte B
- Tree tree
- boolean top_search

Descripción: Comprime el byte recibido por parámetro.

Resultado: En cualquier caso, el árbol recuerda el nodo referente a este Byte como última visita. El comportamiento varía según el valor del parámetro top_search:

- **top_search = true**

El byte B está insertado en el primer nivel del árbol (o bien ya lo estaba, o se ha realizado en esta instancia con índice = comp_file.size()-1).

- **top_search = false**

El byte B está insertado en el nivel i-ésimo del árbol, donde i = número de bytes de la cadena en consideración y por lo tanto también i = numero llamadas previas con top_search = false desde la última llamada con top_search = true. O bien ya estaba insertado, o se ha realizado en esta llamada con índice = comp_file.size()-1).

Nombre: compress

Parámetros: void

Descripción: Función encargada de controlar la compresión. Se comunica con la controladora del compresor, desde la que obtiene la información para comprimir y va haciendo llamadas a la función compress(byte, Tree, boolean) para que comprima cada byte leído. Una vez todo leído y comprimido, verifica que el algoritmo no haya quedado en estado de repetición. En caso de ser así, añade un byte = 0 al final para que la última entrada quede registrada. Finalmente, llama a la función write_compressed_file(), sobre la que delega la escritura del archivo comprimido.

Resultado: El fichero inputFile de la controladora ha sido comprimido mediante el algoritmo LZ-78 y ha sido escrito en el fichero outputFile de la controladora.

Nombre: write_compressed_file

Parámetros: void

Descripción: Escribe en el fichero de salida mediante la controladora del compresor el diccionario comprimido.

Resultado: Se ha escrito el tamaño de y el contenido de comp_file a través de la controladora del compresor.

Decompressor_LZ78

Mediante la controladora, lee un archivo comprimido mediante el algoritmo LZ78 y lo descomprime, escribiendo el resultado en un archivo de salida también a través de la controladora. Para cada entrada recibida, de par índice-byte, es añadida al diccionario para futuras referencias a la misma a la vez que toda cadena descomprimida es añadida al fichero destino por orden de aparición.

CARDINALIDAD

La cardinalidad es 1, ya que solo se puede descomprimir un archivo a la vez, y por lo tanto, de esta clase solo existirá una instancia a la vez.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
dictionary	No	-	Archivo sobre el que se escribe la descompresión actual.
length	No	-	Número de entradas que contendrá dictionary.

MÉTODOS:

Nombre: getExtension

Parámetros: void

Descripción: Se retorna el valor de la extensión personal de los archivos descomprimidos por este descompresor.

Resultado: String “_decompressed.txt”

Nombre: decompress

Parámetros:

- **byte[]** indexB
- **byte** offset

Descripción: Descomprime el par de entrada índice-byte que entra por parámetro.

Resultado: Retorna la cadena de bytes referente al par de entrada que ha recibido como parámetro. Además de haber incluido en el diccionario de la clase la referencia a la cadena retornada.

Nombre: decompress

Parámetros: void

Descripción: Función encargada de controlar la descompresión. A medida que va obteniendo datos, va decidiendo qué cantidades leer en función de lo leído hasta el momento y va tratando las entradas recibidas, incluyendo las en el diccionario y escribiéndolas directamente en el fichero de salida a través de la controladora.

Resultado: El archivo de entrada se ha descomprimido y ha sido escrito a través de la controladora.

Compressor_LZW

La clase que implementa la compression de un fichero mediante el algoritmo LZW.

CARDINALIDAD

Puede haber un compresor alavez por que el árbol que se utiliza como diccionario utiliza unas variables estáticas que son necesarias hasta que el árbol no se borra.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
<i>extension</i>	Sí	".lzw"	La extensión del los ficheros comprimidos por el algoritmo
<i>BYTE_SIZE</i>	Sí	8	La dimensión de un byte
<i>dictionary</i>	No	null	El diccionario de los patrones encontrados
<i>nextIndex</i>	No	0	El índice de la siguiente palabra a insertar
<i>pattern</i>	No	null	La el contenedor para guarda el patrón
<i>codewordSize</i>	No	8	La longitud en bits para escribir la codificación

MÉTODOS:

Nombre: getExtension

Parámetros: void

Descripción: Retorna la extensión de los ficheros comprimidos con este algoritmo

Resultado: La extensión de los ficheros comprimidos

Nombre: compress

Parámetros: void

Descripción: Comprime un fichero mediante el algoritmo LZW. La función lee byte a byte desde el fichero a comprimir mediante el controlador, y construye la palabra más grande de bytes seguidos que no está en el diccionario hasta el momento. Cuando la encuentra, la añade al diccionario con el menor índice no utilizado, escribe el código de esta palabra menos el último carácter en la salida y empieza a buscar otra empezando con el último carácter leído.

Resultado: void

Nombre: toByteArray

Parámetros: p - el ArrayList a convertir

Descripción: Convierte un ArrayList de Bytes en un byte array

Resultado: El byte array equivalente a p

Nombre: concatenate

Parámetros: p - el byte array a concatenar

b - el byte a concatenar

Descripción: Concatena un byte array con un byte

Resultado: Un byte array que representa la concatenación de p y b

Nombre: toByteArray
Parámetros: **codeword** - número a convertir
Descripción: Convierte un int en un array con elementos de 8 bits (Es igual a pasar un integer desde la base 10 a base 256)
Resultado: Un array con la representación de **codeword** en base 256

Nombre: inicializar
Parámetros: void
Descripción: Inicializa el diccionario del compresor. Inserta todos los caracteres de ASCII extendido en el diccionario del compresor. Esto forma el diccionario basico. Además, establece la longitud de los códigos a escribir a 16 bits y reinicia la palabra acumulada.
Resultado: void

Decompressor_LZW

La clase que implementa la decompression de un fichero mediante el algoritmo LZW.

CARDINALIDAD

Puede haber un compresor alavez por que el árbol que se utiliza como diccionario utiliza unas variables estáticas que son necesarias hasta que el árbol no se borra.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
BYTE_SIZE	Sí	8	La dimensión de un byte
dictionary	No	null	El diccionario de los patrones
codewordSize	No	8	La longitud en bits para escribir la codificación

MÉTODOS:

Nombre: inicializar
Parámetros: void
Descripción: Inicializa el diccionario del descompresor. Inserta todos los caracteres de ASCII extendido en el diccionario del compresor. Esto forma el diccionario basico. Además, establece la longitud de los códigos a leer a 16 bits.
Resultado: void

Nombre: getExtension
Parámetros: void
Descripción: Retorna la extensión de los ficheros descomprimidos con este algoritmo. Se anade el sufijo "_decompressed" para no sobrescribir el fichero original comprimido.
Resultado: La extensión del fichero descomprimido

Nombre: decompress

Parámetros: *void*

Descripción: Descomprime un fichero codificado con el algoritmo LZW. La función lee todos los códigos escritos en el fichero comprimido usando la controladora. Si el código leído está en el diccionario entonces se escribe en la salida la cadena de caracteres correspondiente al código leído y se añade al diccionario la cadena resultante a la concatenación de la palabra correspondiente al penúltimo código descomprimido y el primer carácter de la palabra actual, en otro caso se añade al diccionario y se escribe en la salida la palabra correspondiente a la concatenación de la palabra del penúltimo código descomprimido y el primer carácter de la misma palabra.

Resultado: *void*

Nombre: getNextIndex

Parámetros: **codeword** - el byte array a convertir

Descripción: Convierte un byte array en un int. Cada elemento del array representa un dígito en la base 256.

Resultado: Un número entero que representa el resultado de la conversión

Compressor_JPEG

Extensión de la clase Compressor mediante el algoritmo JPEG. Esta clase comprime los archivos utilizando el algoritmo de JPEG. En ella encontramos la función compress y sus funciones auxiliares. También hay una función que devuelve el tipo de extensión que tendrá la salida. La función decompress hace los pasos del JPEG: Blocksplitting, color space transformation, quantization, DCT, zigzag reading y Huffman encoding usando las clases auxiliares Block, Huffman y Triplet.

CARDINALIDAD

Solo existe una instancia de esta clase, ya que se llama desde la clase Compressor_Controller solamente una vez, una clase que solo tiene una instancia.

MÉTODOS:

Nombre: getExtension

Parámetros: -

Descripción: Se retorna el valor de la extensión personal de los archivos comprimidos del compresor.

Resultado: String ".jpeg"

Nombre: compress

Parámetros: -

Descripción: Implementación de la función abstracta de la clase Compressor utilizando el algoritmo JPEG.

Resultado: void

Nombre: RGBtoYCbCr

Parámetros: float R, float G, float B

Descripción: Función para pasar de RGB a la base de color YCbCr usando las fórmulas del estándar de Jfif.

Resultado: Triplet<Integer, Integer, Integer>

Nombre: readHeaders

Parámetros: byte s[]

Descripción: Función que lee los headers del ppm y ignora los comentarios.

Resultado: Triplet<Integer, Integer, Float>

Nombre: stringBinToByte

Parámetros: LinkedList<Integer> bits, ArrayList<Byte> arrayBytes

Descripción: Función que por cada 8 enteros que representan 8 bits, crea un byte y lo añade a arrayBytes.

Resultado: Triplet<Integer, Integer, Float>

Decompressor_JPEG

Extension de la clase Decompressor mediante el algoritmo JPEG. Esta clase descomprime un archivo antiguamente comprimido con la classe Compressor_JPEG. Hace los pasos que anteriormente hicimos con el Compressor a la inversa. Utiliza, como el compresor, las classes Huffman, Block y Triplet para conseguirlo.

CARDINALIDAD

Solo existe una instancia de esta clase, ya que se llama des de la classe Compressor_Controller solamente una vez, una clase que solo tiene una instancia.

MÉTODOS:

Nombre: getExtension

Parámetros: -

Descripción: Se retorna el valor de la extensión personal de los archivos comprimidos del decompresor.

Resultado: String “_decompressed.ppm”

Nombre: decompress

Parámetros: -

Descripción: Implementación de la función abstracta de la clase Decompressor utilizando el algoritmo JPEG.

Resultado: void

Nombre: YCbCrToRGB

Parámetros: int Y, int Cb, int Cr

Descripción: Funcion para canviar de YCbCr a base de colores RGB.

Resultado: Triplet<Byte, Byte, Byte>

Nombre: byteToBin

Parámetros: byte[] s, int[] bits

Descripción: Función que convierte un array de bytes a una cadena de bits representada con enteros.

Resultado: void

Nombre: readBlock

Parámetros: ArrayList<Integer> data, int i, String tipus

Descripción: Funcion que lee un bloque.

Resultado: Block

Block

Clase auxiliar para la implementación del algoritmo JPEG. Representa un subconjunto de los píxeles de una imagen, normalmente de 8x8 pixeles. En ella encontramos alguna función de transformación de estos valores que se usan en el algoritmo de JPEG.

CARDINALIDAD

Hay un bloque por cada 64 pixels de la imagen, ya que dividimos la imagen en bloques de 8x8.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
valors	No	null	Matriz con todos los valores del bloque
DCTvalors	No	null	Matriz con todos los valores del bloque después de aplicar la DCT
width	No	null	Anchura de la imagen
height	No	null	Altura de la imagen
type	No	null	Tipo de bloque (Y, Cb, Cr)
PI	No	Math.PI	Valor de PI
sqrt2	No	sqrt(2)	Raíz quadrada
QTY	No	Tabla de cuantización Y	Matriz que representa la tabla de cuantización de luminancia
QTr	No	Tabla de cuantización Cr	Matriz que representa la tabla de cuantización de crominancia

MÉTODOS:

Nombre: Block

Parámetros: int width, int height, String type

Descripción: Constructora con definición de atributos.

Resultado: -

Nombre: inverseDCT

Parámetros: -

Descripción: Función que aplica la trasformada discreta del coseno inversa a los valores del bloque y guarda su resultado a DCTvalors.

Resultado: void

Nombre: zigzag

Parámetros: int [] file, int ind

Descripción: Guarda en el array file a partir del índice los valores de DCTvalors recorridos en zigzag.

Resultado: void

Nombre: zigzagInvers

Parámetros: ArrayList<Integer> arr, int index

Descripción: Guarda en DCTvalors el contenido de arr recorriendo DCTvalors en zigzag.

Resultado: void

Nombre: inverseQuantizationY

Parámetros: -

Descripción: Multiplica cada valor por la tabla de cuantization de lumincancia

Resultado: void

Nombre: inverseQuantizationC

Parámetros: -

Descripción: Multiplica cada valor por la tabla de cuantization de crominancia

Resultado: void

Huffman

Clase auxiliar para la implementación del algoritmo JPEG. Esta clase comprime un array de enteros usando el algoritmo de compresión de Huffman. Para ello, cuenta la frecuencia de aparición de los enteros en la cadena y después genera un árbol de Huffman a partir de estos, para generar los códigos de Huffman de cada entero. También puede descomprimir una cadena de bits que haya sido comprimida anteriormente con esta clase, ya que al comprimir un archivo, se pasa al principio de este el diccionario utilizado para hacerlo.

CARDINALIDAD

Se crea una instancia de esta clase cada vez que se llama al compresor o al descompresor JPEG.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
dictionary	No	new HashMap()	Diccionario de enteros (clave) y su clave de Huffman en binario en un String.

MÉTODOS:

Nombre: encode

Parámetros: int[] file, LinkedList<Integer> bits

Descripción: Función para comprimir usando Huffman el array file.

Resultado: void

Nombre: isSeparador

Parámetros: int[] bits, int index

Descripción: Función que nos dice si en la posición index del array de enteros bits empieza un separador de forma 011111111111111110.

Resultado: boolean

Nombre: decode

Parámetros: int[] file, ArrayList<Integer> valores

Descripción: Descomprime una cadena de bits comprimidos con Huffman en un ArrayList de enteros.

Resultado: void

Nombre: addSeparador

Parámetros: LinkedList<Integer> bits

Descripción: Función que añade un separador a la cadena de bits.

Resultado: void

Nombre: addDictionary

Parámetros: LinkedList<Integer> bits

Descripción: Función que añade el diccionario de Huffman utilizado para comprimir al principio de una cadena de bits.

Resultado: void

Nombre: makeDict

Parámetros: Node root, String s

Descripción: Función que recorre recursivamente el árbol para crear los códigos de Huffman i añadirlos al diccionario.

Resultado: void

Nombre: calculateFreq

Parámetros: int[] file

Descripción: Función que recorre el array de enteros y calcula la frecuencia con la que aparecen los enteros.

Resultado: LinkedHashMap<Integer, Integer>

Compressor_LZSS

Extensión de la clase [Compressor](#) mediante el algoritmo LZ-SS.

Para comprimir mediante el algoritmo LZSS vamos a ir iterando por los caracteres del texto que tenemos que comprimir y buscando coincidencias con los caracteres que ya hemos visto. Para eso, vamos a guardar los caracteres que visitamos en un searchBuffer, y para cada iteración buscaremos en el searchBuffer una coincidencia con el carácter actual y, como mínimo, con los dos caracteres siguientes.

CARDINALIDAD

Solo existe una instancia de esta clase, ya que se llama desde la clase Compressor_Controller solamente una vez, una clase que solo tiene una instancia.

Nombre	Estático	Valor por defecto	Descripción breve
searchBuffer	No	null	HashMap<Byte, ArrayList<Integer> >, en que se guardará como key los bytes que vamos encontrando y en el arraylist las posiciones en que aparecen estos.
act	No	null	ArrayList<Pair<Integer, Byte>> que servirá para actualizar el searchBuffer y que tenga como máximo 4095 posiciones guardadas. Añadiremos los bytes y las posiciones que nos vamos encontrando.

MÉTODOS:

Nombre: getExtension

Parámetros: void

Descripción: Retorna la extensión de los ficheros descomprimidos con este algoritmo. Se añade el sufijo "_decompressed" para no sobrescribir el fichero original comprimido.

Resultado: La extensión del fichero descomprimido

Nombre: coincidence

Parámetros: byte[] itemb, int p

Descripción: busca coincidencias en el searchBuffer y retorna un carácter codificado si encuentra coincidencia de más de 3. Si la coincidencia es menor, retorna 0x0000.

Resultado: Short

Nombre: codify

Parámetros: int maxdesp, int maxpos, int p

Descripción: Codifica un short en que los primeros 12 bits serán maxpos (offset relativo desde p) y los últimos 4 serán maxdesp.

Resultado: Short

Nombre: groupBits

Parámetros: boolean[] a

Descripción: agrupa los bits representados por el array de boolean a en bytes

Resultado: byte[]

Nombre: arrayListToArray

Parámetros: arrayList<Byte> a

Descripción: Crea y retorna un array de bytes con los mismos elementos que a

Resultado: byte[]

Nombre: mergeArrays

Parámetros: byte[] a, byte[] b

Descripción: Concatena los arrays a y b y retorna un array con la primera parte a y la segunda b.

Resultado: byte[]

Nombre: actualitzar

Parámetros:

Descripción: Actualiza el searchBuffer, mientras act.size sea menor que 4095, borra las posiciones de act y searchBuffer que hace mas tiempo que se han añadido.

Resultado: void

Decompressor_LZSS

Extensión de la clase [Decompressor](#) mediante el algoritmo LZ-SS.

Para decomprimir mediante el algoritmo LZSS convertiremos el fichero con el texto comprimido en un array de bytes leyendo byte a byte, y diferenciaremos en dos partes: Los bits codificados en bytes y los caracteres codificados o no. Una vez hayamos diferenciado las partes y hayamos puesto arrays, iteraremos por cada elemento del array de bits (que corresponde a la primera parte). Si en la iteración actual encontramos un valor false (0), copiaremos el elemento de la posición que evaluamos del array de bytes en el resultado. Si encontramos un valor true (1), descodificaremos los siguientes dos bytes que toquen, cogiendo los 12 bits de mas peso como offset y los 4 de menos peso como desplazamiento, y buscaremos los caracteres correspondientes en el array de resultado.

CARDINALIDAD

Solo existe una instancia de esta clase, ya que se llama desde la clase Decompressor_Controller solamente una vez, una clase que solo tiene una instancia.

MÉTODOS:

Nombre: getExtension

Parámetros: void

Descripción: Retorna la extensión de los ficheros descomprimidos con este algoritmo. Se añade el sufijo "_decompressed" para no sobrescribir el fichero original comprimido.

Resultado: La extensión del fichero descomprimido

Nombre: mergeBytes

Parámetros: byte high, byte low

Descripción: Crea un short con el byte de mas peso a y el byte de menos peso b

Resultado: Short

Pair privada LZ-78

Clase auxiliar para la implementación del diccionario del compresor mediante el algoritmo LZ-78. Dado que la compresión se basa en pares de índice y valor, esta clase representa cada entrada del diccionario generado.

CARDINALIDAD

Esta clase es instancia múltiples veces, ya que en la compresión mediante LZ-78 se crea una instancia de esta clase para cada entrada del diccionario creado.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
index	No	-	Índice referente a su antecesor (es igual a 0 en caso de no tener antecesor).
offset	No	-	Byte que concatenado con los bytes de todos sus antecesores, forma la cadena de bytes representada por una entrada de este tipo.

MÉTODOS:

Nombre: Pair (constructora)

Parámetros:

- int **i**
- byte **b**

Descripción: Constructora de la clase **Pair** dado el contenido de sus variables.

Resultado: Se ha creado una instancia de la clase Pair con los valores pasados por parámetro.

Tree

Árbol de búsqueda utilizado para agilizar la compresión en los algoritmos LZ-78 y LZ-W. Cada árbol contiene su información propia (tratamiento a nivel de nodo) más la información de sus hijos (tratamiento a nivel de árbol).

CARDINALIDAD

Se crean múltiples instancias de la clase árbol para construir árboles más grandes.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
id	No	-	Byte identificador del nodo del árbol.
index	No	-1	Índice referente a los diccionarios de los algoritmos de compresión guardado en el nodo del árbol.
sons	No	lista vacía	Lista de arboles hijo.
last_visited_node	Si	-	Referencia al último nodo/arbol visitado (utilizado para hacer búsquedas byte a byte sin tener que recorrer el árbol completo).
next_index	Si	-	Siguiente índice que se debe asignar a un nuevo nodo/árbol (Dado que los algoritmos asignan índices de forma creciente y continua, con esta variable se mantiene en todo momento constancia de cuál es el siguiente que se debe asignar.).

MÉTODOS:

Nombre: find

Parámetros:

- **word** Cadena de bytes a buscar.

Descripción: Busca recursivamente una palabra (cadena de bytes) en el árbol de búsqueda.

Resultado: Retorna el "index" del nodo buscado o -1 si este no existía.

Nombre: find

Parámetros:

- **word** Cadena de bytes a buscar.
- **digit** Índice sobre la cadena word que se inspecciona en cada llamada.

Descripción: Función recursiva sobre la que se realizan las búsquedas comprobando byte a byte.

Resultado: Retorna, al final de la recursión, el "index" del nodo buscado o -1 si este no existía.

Nombre: clear

Parámetros: void

Descripción: El árbol se vacía y restaura a uno nuevo.

Resultado: El árbol se ha reiniciado. Ahora está vacío.

Nombre: getTree

Parámetros:

- byte **B**

Descripción: Función que busca y retorna el árbol con id = B en el nivel correspondiente.

Resultado: Retorna el árbol con id = B si existe. Retorna null en caso contrario.

Nombre: progressive_find

Parámetros:

- **B** Byte identificador (id) del nodo que se quiere buscar.
- **restart** Indica si se debe realizar la búsqueda sobre los hijos de la raíz del árbol (true) o sobre los hijos del último nodo visitado (false).

Descripción:

El funcionamiento varía según el valor de restart:

- Si restart == true: La búsqueda se lleva a cabo sobre los hijos de la raíz del árbol.
- Si restart == false: La búsqueda se lleva a cabo sobre los hijos del último nodo visitado.

Se busca un nodo con id = B.

- Si este existe se retorna su contenido index.
- Si este no existe, se crea e inserta un nodo con id = B, index = next_index y una lista de hijos vacía; y se retorna index.

Resultado: Retorna el "index" del nodo buscado. Si este no existía, index = next_index.

Nombre: insert

Parámetros:

- **word** Cadena de bytes a insertar.
- **index** Index que se quiere guardar como dato asociado a la palabra word.

Descripción: Inserta una palabra (cadena de bytes) en el árbol de búsqueda (si esta no está presente).

Resultado: La palabra word ha sido insertada con index = index.

Nombre: insert

Parámetros:

- **word** Cadena de bytes a insertar.
- **index** Index que se quiere guardar como dato asociado a la palabra word.
- **digit** Índice sobre la palabra word que indica el byte que se debe insertar en esta llamada.

Descripción: Inserta recursivamente (byte a byte) una palabra (cadena de bytes) en el árbol de búsqueda.

Resultado: La palabra word ha sido insertada con index = index.

Nombre: putTreeIfAbsent

Parámetros:

- byte **B**

Descripción: Busca el nodo/árbol con id = B. Si existe: retorna el nodo/árbol. Si no existe: crea un nodo/árbol con id = B, index = next_index y un árbol vacío; y lo inserta en la lista de hijos.

Resultado: Retorna el nodo con id = B de la lista de hijos.

Nombre: findNextByte

Parámetros:

- byte **B**

Descripción:

Función auxiliar de la función "progressive_find". Se busca un nodo/arbol con id = B entre los hijos de last_visited_node.

- Si este existe se retorna su contenido index.
- Si este no existe, se crea e inserta un nodo con id = B, index = next_index y una lista de hijos vacía; y se retorna index.

Resultado: El Nodo buscado pasa a ser el último nodo visitado (last_visited_node).

Nombre: areYou

Parámetros:

- byte **B**

Descripción:

Función que retorna si el nodo/arbol tiene como identificador el parámetro id y además contiene un índice válido (está "presente").

Resultado: Devuelve True si el parámetro id se corresponde con su identificador y ademas su índice es valido (index >= 0). False en caso contrario.

DESCRIPCIÓN DE CLASES DE LA PERSISTENCIA

Persistence_Controller

Controladora de la capa de persistencia, encargada de todos los procesos de entrada y salida. La comunicación para las operaciones de entrada y salida con el dominio va ligada siempre a un identificador asociado a un fichero sobre el que se desea leer/escribir. Todos los procesos de lectura y escritura se realizan mediante buffers. También incluye métodos para obtener información de archivos y directorios (nombre, tamaño, etc.)

CARDINALIDAD

Es una clase singleton, por lo tanto se instancia una sola vez (cardinalidad es uno).

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
persistence_controller	Si	nueva instancia de Persistence_Controller	Referencia a la única instancia de la Controladora de Persistencia. (Patrón singleton)
readFiles	No	lista vacía	Lista de archivos/directorios para realizar operaciones de lectura (la posición en la lista es el identificador asociado al mismo).
writeFiles	No	lista vacía	Lista de archivos/directorios para realizar operaciones de escritura (la posición en la lista es el identificador asociado al mismo).

MÉTODOS:

Nombre: getPersistenceController()

Parámetros: void

Descripción: Getter de la única instancia de la Controladora de Persistencia.

Resultado: Retorna la instancia de la Controladora de Persistencia.

Nombre: newInputFile

Parámetros: void

Descripción: Añade al sistema un fichero (puede ser directorio) sobre el que se pueden realizar lecturas y se le asocia un identificador.

Resultado: Devuelve Identificador asociado al fichero.

Nombre: newOutputFile

Parámetros:

- String path

Descripción: Añade al sistema un fichero (puede ser directorio) sobre el que se pueden realizar escrituras y se le asocia un identificador.

Resultado: Devuelve Identificador asociado al fichero.

Nombre: newOutputFile

Parámetros:

- String **name**
- int **padre**

Descripción: Añade al sistema un fichero (puede ser directorio) sobre el que se pueden realizar escrituras, se la asocia un identificador y una ruta = ruta del fichero padre + nombre.

Resultado: Devuelve Identificador asociado al fichero.

Nombre: newDir

Parámetros:

- String **path**

Descripción: Añade al sistema un nuevo directorio.

Resultado: Devuelve Identificador asociado al directorio.

Nombre: newDir

Parámetros:

- String **name**
- int **padre**

Descripción: Añade al sistema un nuevo directorio.

Resultado: Devuelve Identificador asociado al directorio.

Nombre: newDir

Parámetros:

- String **name**
- int **padre**

Descripción: Añade al sistema un nuevo directorio.

Resultado: Devuelve Identificador asociado al directorio.

Nombre: clear

Parámetros: void

Descripción: Función que restaura al completar la persistencia.

Resultado: Las listas de archivos referenciados se han vaciado.

Nombre: setReadLimit

Parámetros:

- int **id**
- int **num**

Descripción: Establece un límite virtual de bytes a leer de un archivo. A partir de establecer un límite, todas las funciones de lectura trabajan en función de este mismo.

Resultado: Se ha establecido un límite de bytes a leer en el archivo asociado al identificador.

Nombre: rmReadLimit

Parámetros:

- int **id**

Descripción: Elimina el límite virtual, volviendo así todas las funciones de lectura a su estado natural de funcionamiento.

Resultado: Se ha eliminado el límite de bytes a leer en el archivo asociado al identificador.

Nombre: getReadLimit

Parámetros:

- int **id**

Descripción: Getter de la cantidad de bytes restantes por leer según el límite establecido..

Resultado: Retorna el límite. En caso de no estar establecido o esté haberse agotado, se retorna un -1.

Nombre: getReadLimit

Parámetros:

- int id

Descripción: Getter de la cantidad de bytes restantes por leer según el límite establecido..

Resultado: Retorna el límite. En caso de no estar establecido o esté haberse agotado, se retorna un -1.

Nombre: getName

Parámetros:

- int id

Descripción: Proporciona el nombre del archivo de lectura asociado al identificador recibido por parámetro .

Resultado: Retorna el nombre del fichero identificado por el parámetro id.

Nombre: getExtension

Parámetros:

- int id

Descripción: Proporciona la extensión del archivo de lectura asociado al identificador recibido por parámetro.

Resultado: Retorna la extensión del fichero (si la tiene).

Nombre: getInputFileSize

Parámetros:

- int id

Descripción: Getter del tamaño en Bytes de un fichero de lectura.

Resultado: Retorna el tamaño en Bytes del fichero original.

Nombre: getOutputFileSize

Parámetros:

- int id

Descripción: Getter del tamaño en Bytes del fichero comprimido.

Resultado: Retorna el tamaño en Bytes del fichero original.

Nombre: getWrittenBytes

Parámetros:

- int id

Descripción: Getter de la cantidad de bytes escritos hasta el momento en un archivo referenciado por id.

Resultado: Retorna la cantidad de bytes escritos en el fichero referenciado por id.

Nombre: readByte

Parámetros:

- int id

Descripción: Lee un byte del fichero origen.

Resultado: Entero que contiene el byte leído o -1 si no había nada que leer.

Nombre: readBytes

Parámetros:

- int id
- byte[] word

Descripción: Lee N bytes del fichero origen en una cadena de bytes que se le pasa por parámetro.

Resultado: Cantidad de bytes leída o -1 si no había nada que leer.

Nombre: readAllBytes

Parámetros:

- int **id**

Descripción: Lee todos los bytes del fichero origen y los guarda en una cadena.

Resultado: Cadena de bytes con todos los bytes del fichero origen.

Nombre: closeReader

Parámetros:

- int **id**

Descripción: Cierra el buffer de lectura.

Resultado: Buffer de lectura se ha cerrado.

Nombre: writeByte

Parámetros:

- int **id**

Descripción: Escribe un byte en fichero de salida.

Resultado: Se ha escrito un byte en el fichero de salida.

Nombre: writeBytes

Parámetros:

- int **id**
- byte[] **word**

Descripción: Escribe una cadena de bytes en el fichero de salida.

Resultado: Se ha escrito un conjunto de bytes en el fichero de salida.

Nombre: closeWriter

Parámetros:

- int **id**

Descripción: Cierra buffer de escritura.

Resultado: Buffer de escritura se ha cerrado.

Nombre: modifyLong

Parámetros:

- int **id**
- long **position**
- long **content**

Descripción: Reemplaza los 8 Bytes localizados en la posición indicada de un fichero y por otros 8 Bytes codificados en un long que se recibe por parámetro.

Resultado: Se han reemplazado los 8 bytes localizados en la posición indicada por los pasados por parámetro.

Nombre: isFolder

Parámetros:

- int **id**

Descripción: Función que dice si un fichero es directorio o no lo es.

Resultado: Retorna true en caso de ser un directorio o falso en caso contrario.

Nombre: getFilesFromFolder

Parámetros:

- int id

Descripción: Función que retorna una lista con todos los ficheros que contiene una carpeta.

Resultado: Todos sus ficheros han sido añadidos al sistema y estos están asociados a un identificador. y retorna una lista que contiene los identificadores que referencian a los archivos que contiene el directorio referenciado por el parámetro.

Nombre: makeHierarchy

Parámetros:

- String path

Descripción: Función que retorna la jerarquía completa de ficheros a partir del path pasado por parámetro.

Resultado:

Devuelve una matriz de 2xN donde N es el número de identificadores/ficheros que hay dentro del path. Cada uno de los identificadores se corresponde con las columnas de la matriz.

- Fila 0: Contiene 1 si el fichero es un directorio, o 0 si no lo es.
- Fila 1: Contiene el identificador del directorio padre del fichero. (El mismo en el caso de la raíz).

OutputFile

Extensión de la clase File de uso exclusivo para ficheros de escritura. Además de contar con todas las funcionalidades propias de un File, cuenta con un buffer propio de escritura y un contador de bytes escritos en cada fichero.

CARDINALIDAD

Esta clase es instancia múltiples veces, ya surge una instancia nueva para cada archivo con el que se quiera escribir. Durante la ejecución, para cada compresión realizada en un destino diferente, habrá una instancia de esta clase.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
out	No	null	Buffer de escritura del archivo asociado.
active	No	false	Control sobre el buffer de escritura. Si está activo o no.
append	No	false	Control sobre la primera vez que se escribe.
num	No	0	Contador de cuantos bytes se han escrito.

MÉTODOS:

Nombre: OutputFile (constructora)

Parámetros:

- String **pathname**

Descripción: Constructora modificada que crea un File con el path pasado por parámetro.

Resultado: Se ha creado un fichero nuevo en el path pasado por parámetro.

Nombre: getBuffer

Parámetros: **void**

Descripción: Retorna el buffer de escritura. Si este no esta activo, lo activa.

Resultado: Retorna el buffer de escritura. Si este no esta activo, lo activa.

Nombre: getNum

Parámetros: **void**

Descripción: Getter del contador de cuantos bytes se han escrito.

Resultado: Retorna la cantidad de bytes escritos hasta el momento.

Nombre: sumNum

Parámetros:

- int i

Descripción: Incrementa el contador de bytes escritos.

Resultado: Se ha Incrementado el contador de bytes escritos.

Nombre: closeBuffer

Parámetros: **void**

Descripción: Cierra el buffer de escritura.

Resultado: Cierra el buffer de escritura.

Nombre: flushBuffer
Parámetros: void
Descripción: Realiza un flush del buffer de escritura.
Resultado: Realiza un flush del buffer de escritura.

Nombre: isActive
Parámetros: void
Descripción: Función que consulta si el buffer de escritura está activo o no.
Resultado: Retorna un booleano que indica si el buffer de escritura está abierto o no.

InputFile

Extensión de la clase File de uso exclusivo para ficheros de lectura. Además de contar con todas las funcionalidades propias de un File, cuenta con un buffer propio de lectura y un sistema de límites de lectura en bytes.

CARDINALIDAD

Esta clase es instancia múltiples veces, ya surge una instancia nueva para cada archivo del que se quiera leer. Durante la ejecución, para cada archivo que se deba comprimir, se crea una instancia de esta clase.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
in	No	null	Buffer de lectura del archivo asociado.
active	No	false	Control sobre el buffer de lectura. Si está activo o no.
num	No	0	Límite de bytes que se pueden leer.
límite	No	false	Control de si las lecturas son con límite de bytes o no.

MÉTODOS:

Nombre: InputFile (constructora)

Parámetros:

- String **pathname**

Descripción: Constructora modificada que crea un File con el path pasado por parámetro.

Resultado: Se ha creado un fichero nuevo con el path pasado por parámetro.

Nombre: getBuffer

Parámetros: void

Descripción: Retorna el buffer de lectura. Si este no esta activo, lo activa.

Resultado: Retorna el buffer de lectura. Si este no esta activo, lo activa.

Nombre: getNum

Parámetros: void

Descripción: Getter del límite actual (cantidad restante) de bytes que se pueden leer.

Resultado: Retorna el número de bytes que se pueden leer según el límite establecido.

Nombre: setNum

Parámetros:

- long **num**

Descripción: Setter del límite de bytes que se pueden leer.

Resultado: Set del límite de bytes que se pueden leer.

Nombre: subNum

Parámetros:

- int n

Descripción: Función que decrementa el límite de bytes a leer.

Resultado: Devuelve la cantidad restada. Si este ha llegado a 0, se desactiva y se establece a -1.

Nombre: rmNum

Parámetros: void

Descripción: Desactiva el numero limite de bytes de lectura.

Resultado: Desactiva el numero limite de bytes de lectura.

Nombre: closeBuffer

Parámetros: void

Descripción: Cierra el buffer de lectura.

Resultado: Cierra el buffer de lectura.

Nombre: isLimited

Parámetros: void

Descripción: Función para saber si el límite de bytes de lectura está activo.

Resultado: Retorna un booleano que indica si el límite está activo o no.

Nombre: isActive

Parámetros: void

Descripción: Función que consulta si el buffer de lectura está activo o no.

Resultado: Retorna un booleano que indica si el buffer de lectura está abierto o no.

Domain Controller

Controladora de Dominio, que gestiona todas las operaciones hechas durante la compresión y la descompresión. Está conectada a la controladora de persistencia i a la de presentación, y se comunica con ellas cuando es necesario.

CARDINALIDAD

Solo hay una sola instancia de esta clase ya que es la controladora de dominio.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
persistence_controller	No	-	Instancia de controladora de persistencia
presentation_controller	No	null	Instancia de controladora de presentación
time	No	-	Tiempo de operación
comp_size	No	-	Tamaño de la carpeta comprimida
orig_size	No	-	Tamaño de la carpeta original

MÉTODOS:

Nombre: Domain_Controller (constructora)

Descripción: Constructora modificada que inicializa al instancia de persistence controller.

Nombre: readByte

Parámetros: int id

Descripción: Lee un byte del fichero origen.

Resultado: Entero que contiene el byte leído o -1 si no habia nada que leer.

Nombre: readNBytes

Parámetros: int id, byte[] word

Descripción: Lee N bytes del fichero identificado por el numero id en una cadena de bytes que se le pasa por parametro. El N representa la longitud de la cadena que tiene como parametro.

Resultado: Cantidad de bytes leída o -1 si no habia nada que leer.

Nombre: closeReader

Parámetros: int id

Descripción: Cierra el buffer de lectura que tiene el identificador id.

Nombre: readAllBytes

Parámetros: int id

Descripción: Lee todos los bytes del fichero origen y los guarda en una cadena.

Resultado: Cadena de bytes con todos los bytes del fichero origen.

Nombre: writeByte
Parámetros: int id
Descripción: Escribe un byte en fichero de salida.
Resultado: Void

Nombre: writeBytes
Parámetros: int id, byte[] word
Descripción: Escribe una cadena de bytes en el fichero de salida.
Resultado: Void

Nombre: closeWriter
Parámetros: int id
Descripción: Cierra buffer de escritura.
Resultado: Void

Nombre: getInputFileSize
Parámetros: int id
Descripción: Calcula la dimension del fichero de entrada con el identificador id
Resultado: La dimension en bytes del fichero

Nombre: getOutputFileSize
Parámetros: int id
Descripción: Calcula la dimension del fichero de salida con el identificador id
Resultado: La dimension en bytes del fichero1

Nombre: getNameNE
Parámetros: String path
Descripción: Elimina la extension de una direccion. Si en la direccion path esta una carpeta, devuelve el path de entrada si no, devuelve el path sin la extension
Resultado: Si en el path es una carpeta, devuelve el path de la entrada si no, la direccion sin la extension

Nombre: isFolder
Parámetros: String path
Descripción: Verifica si el elemento de la direccion path es una carpeta o no
Resultado: true - si es una carpeta, no - en caso contrario

Nombre: getBestCompressor
Parámetros: int in, int alg
Descripción: Calcula el algoritmo de compression en dependencia del fichero y algoritmo que recibe como parametro.
Resultado: Si la extension es .ppm entonces retorna el algoritmo 3, si la extension es .txt y $0 \leq alg \leq 2$, retorna el algoritmo correspondiente si no retorna el algoritmo mas adecuado para este fichero. Si el fichero es de otro tipo retorna en dependencia del tamano del fichero el algoritmo mas apropiado.

Nombre: getPathAndName
Parámetros: String file
Descripción: Calcula la direccion y el nombre de un fichero eliminando la extension
Resultado: la direccion con el nombre del fichero sin la extension

Nombre: writeFolderMetadata
Parámetros: int id, Hierarchy h

Descripción: Escribe la estructura de una carpeta/fichero en un fichero codificado por id. Al principio del fichero se escribe el tamaño de la cabecera (4 bytes), a continuación va el número de ficheros comprimidos en este fichero (2 bytes) seguido de la codificación en bits para cada fichero de si es una carpeta (1) o si es un fichero (0)($\text{ceil}(\# \text{elementos}/8)$). A continuación para cada fichero se escribe el identificador de la carpeta padre (1 byte), para el fichero raíz el padre es el mismo. La extensión del fichero comprimido(cadena vacía en el caso de si es una carpeta) seguido de un salto de línea, seguido de todos los nombres, salvo el elemento raíz, que se van a comprimir separados por saltos de líneas.

Resultado: void

Nombre: encodeMeta

Parámetros: long compressor, long i

Descripción: Codifica el identificador de un compresor y la longitud de un fichero es un long. Los primeros 2 bits de mayor peso representa el algoritmo utilizado para comprimir el fichero, los otros 62 bits representan el tamaño del fichero comprimido.

Resultado: Void

Nombre: makeHierarchy

Parámetros: String inputPath, String outputPath, boolean sobrescribir

Descripción: Desde un fichero comprimido lee la cabecera y crea una tabla de 2XN donde la primera fila representa si un elemento es carpeta o fichero y la segunda representa para cada elemento cual es su padre. Además pide a persistencia que cree todos los ficheros y carpetas de esta jerarquía.

Resultado: una tabla de 2XN donde la primera fila representa si un elemento es carpeta o fichero y la segunda representa para cada elemento cual es su padre, el N es el número de elementos

Nombre: toInt

Parámetros: byte[] aux

Descripción: Convierte un array de bytes en un integer.

Resultado: el valor convertido

Nombre: toLong

Parámetros: byte[] aux

Descripción: Convierte una array de bytes en un long.

Resultado: el valor convertido

Nombre: getRatio

Parámetros: -

Descripción: Obtiene el ratio de la última compresión

Resultado: Obtiene el ratio de la última compresión

Nombre: deleteFile

Parámetros: String s

Descripción: Borra el archivo con path s

Resultado: void

Nombre: compress

Parámetros: (String in, boolean sobrescribir

Descripción: Comprime un fichero con el algoritmo más adecuado. Si el fichero es PPM el ratio de compresión es 5

Resultado: Void

Nombre: compress

Parámetros: String in, byte ratio, boolean sobrescribir

Descripción: Comprime un fichero PPM.

Resultado: Void

Nombre: compress

Parámetros: String in, int alg, boolean sobrescribir

Descripción: Comprime un fichero con el algoritmo específico

Resultado: Void

Nombre: compress

Parámetros: String in, String out, boolean sobrescribir

Descripción: Comprime un fichero con el algoritmo más adecuado y lo guarda en la dirección y con el nombre <<out>> Si el fichero es PPM el ratio de compresión es 5

Resultado: Void

Nombre: compress

Parámetros: String in, String out, byte ratio, boolean sobrescribir

Descripción: Comprime un fichero PPM.

Resultado: void

Nombre: compress

Parámetros: String in, String out, int alg, boolean sobrescribir

Descripción: Comprime un fichero con un algoritmo específico. Si el fichero es PPM el ratio de compresión es el por defecto (5)

Resultado: Void

Nombre: compress

Parámetros: String inputPath, String outputPath, int alg, byte ratio, boolean sobrescribir

Descripción: Comprime un fichero cualquiera con el algoritmo específico. Si el fichero es PPM utiliza el ratio especificado como parametro. La compresión empieza con identificar la jerarquía del elemento que se quiere comprimir. Haberlo identificado, se crea y se escribe la estructura, la extensión del elemento a comprimir y si es una carpeta, los nombres de cada elemento de esta carpeta.

Resultado: Void

Nombre: decompress

Parámetros: String inputPath, String outputPath, boolean sobrescribir

Descripción: Descomprime un fichero con la extensión ".egg" La decompression empieza por leer la cabecera del fichero a descomprimir. Haberlo acabado, se genera la jerarquía del fichero descomprimido, mientras esta generando la jerarquía esta pidiendo a la capa de datos que replique la jerarquía en carpetas y ficheros (la extensión de la raíz de la jerarquía se halla al leer la cabecera). Después, para cada elemento de esta jerarquía escribe su contenido en los ficheros creados.

Resultado: Void

Nombre: visualiceFile

Parámetros: String path

Descripción: Visualiza los ficheros ".txt" y ".ppm"

Resultado: Void

Private class Hierarchy:

Nombre: getFilesList

Parámetros: -

Descripción: Obtiene todos los identificadores de las carpetas y ficheros de la jerarquia

Resultado: una lista con identificadores de todos los ficheros y carpetas

Nombre: Hierarchy (constructora)

Parámetros: int[][] src

Descripción: Constructora de la clase que crea un objeto a partir de una tabla 2xN, en la cual la primera linea representa si el elemento con el identificador numero de columna es carpeta o fichero y la segunda representa el antecesor de este elemento en la jerarquia

Nombre: toByteArray

Parámetros: -

Descripción: Codifica una jerarquia en una array de bytes. Los primeros 4 bytes de la tabla representa la longitud del array en el que se convertira la jerarquia. Los siguientes 2 representan el numero de elementos en la jerarquia (carpetas y ficheros). Los siguientes $\text{ceil}(\# \text{elemento} / 8)$ bytes representan el tipo de cada elemento de la jerarquia (carpeta o fichero). Los siguientes $\# \text{elementos}$ bytes representan el identificador del antecesor de cada elemento en particular.

Resultado: la codificacion de la jerarquia

Nombre: getLeafs

Parámetros: -

Descripción: Obtiene todos los identificadores de los ficheros de la jerarquia

Resultado: una lista con los identificadores de todos los ficheros de la jerarquia

Nombre: getLeafsAux

Parámetros: int i

Descripción: Funcion auxiliar para obtener todos los identificadores de los ficheros de la jerarquia

Resultado: una lista con todos los identificadores de todos los ficheros en el elemento i

DESCRIPCIÓN DE CLASES DE LA PRESENTACIÓN

Presentation_Controller

Controladora de la presentación, encargada de gestionar la GUI y el envío de información de entrada a domain controller. Está conectada solamente con la controladora de dominio.

CARDINALIDAD

Solamente se instancia una vez.

ATRIBUTOS

Nombre	Estático	Valor por defecto	Descripción Breve
domain_controller	Si	nueva instancia de Domain_Controller	Referencia a la única instancia de la Controladora de Dominio.
seleccionarArchivo	No	nueva instancia de seleccionarArchivo	Referencia a la vista de la GUI seleccionarArchivo
metodoCompresion	No	nueva instancia de metodoCompresion	Referencia a la vista de la GUI metodoCompresion
seleccionarDestino	No	nueva instancia de seleccionarDestino	Referencia a la vista de la GUI seleccionarDestino
jpeGselect	No	nueva instancia de jpegSelect	Referencia a la vista de la GUI jpegSelect
welcome	No	nueva instancia de welcome	Referencia a la vista de la GUI welcome
end	No	nueva instancia de end	Referencia a la vista de la GUI end
action	No	0	Variable de control para saber la operación seleccioanda. 0 es comprimir y 1 descomprimir
algorithm	No	null	Variable de control para saber que algoritmo de compresion se ha escogido
carpeta	No	false	Variable de control para saber si el usuario ha seleccionado una carpeta para comprimir
SourcePath	No	null	Ruta hasta el archivo o carpeta que el usuario ha seleccionado para comprimir o descomprimir
OutPath	No	null	Ruta hasta la carpeta que el usuario ha seleccionado para guardar el archivo comprimido o descomprimido.
JPEGratio	No	5	Ratio de compresión del JPEG seleccionado

			por el usuario
sobreEscribir	No	null	Booleano que es true si el usuario quiere sobreEscribir el archivo.
mode	No	null	Parámetro de control para la visualización final de los archivos. 1 -> compresion de un .txt 2 -> compresion de un ppm -1 -> descompresion txt -2 -> descompresion ppm)

Nombre: Presentation_Controller

Parámetros:

Descripción: Creadora de la clase Presentation controller, crea la referencia a donaim controller.

Resultado:

Nombre: initializeInterface

Parámetros:

Descripción: Crea la instancia de JFrame que usaremos, y la configura para que el form displayeado sea Welcome, que es el primero que se tiene que ver. A parte define el action Listener para cuando aprietas la cruz de la ventana.

Resultado:

Nombre: switchToMetodoCompresion

Parámetros:

Descripción: Cambia el form mostrado por el form MetodoCompresion

Resultado:

Nombre: switchToSeleccionarDestino

Parámetros:

Descripción: Cambia el form mostrado por el form SeleccionarDestino

Resultado:

Nombre:switchToJPEGselect

Parámetros:

Descripción: Cambia el form mostrado por el form JPEGSelect

Resultado:

Nombre: switchToSeleccionarArchivo

Parámetros:

Descripción: Cambia el form mostrado por el form SeleccionarArchivo

Resultado:

Nombre: switchToEnd

Parámetros:

Descripción: Cambia el form mostrado por el form End

Resultado:

Nombre: switchToWelcome

Parámetros:

Descripción: Cambia el form mostrado por el form Welcome

Resultado:

Nombre: isFolder

Parámetros: string path

Descripción: Llama al método isfolder de domain controller.

Resultado: boolean

Nombre: close

Parámetros:

Descripción: Borra la carpeta temp y cierra el frame

Resultado:

Nombre: sendInfo

Parámetros:

Descripción: Función que llama a los métodos de compresión o descompresión correspondientes de domain controller con la información introducida por el usuario en la GUI

Resultado:

Nombre: visualizeFile

Parámetros: string path

Descripción: Función que llama el método visualizeFile de domain controller

Resultado:

Nombre: decompress

Parámetros: String out, String in, boolean sobreEscribir

Descripción: Función que llama el método decompress de domain controller con los parámetros correspondientes

Resultado:

Nombre: deleteTemp

Parámetros:

Descripción: Borra la carpeta temporal Temp

Resultado:

Clases de GUI

Para la GUI hemos usado clases de java con forms, cada clase representa una vista distinta de la interfície y cada clase se instancia una sola vez desde presentation controller. Los atributos de estas correspondrán con los ítems de la paleta de Swing y las únicas funciones pertinentes son las constructoras respectivas de cada clase y las denominadas action listeners, que son las encargadas de modificar los atributos de presentation controller con la información proporcionada por el usuario cuando es pertinente.

CARDINALIDAD

Solamente se instancia una vez, desde presentacion controller.