SIPPY

Entrega 2

Sergio Vázquez

Víctor Blanco

Yaiza Cano

Miguel Ángel

ÍNDEX

1. Descripció textual dels casos d'ús:

Cas d'us general de Sippy:

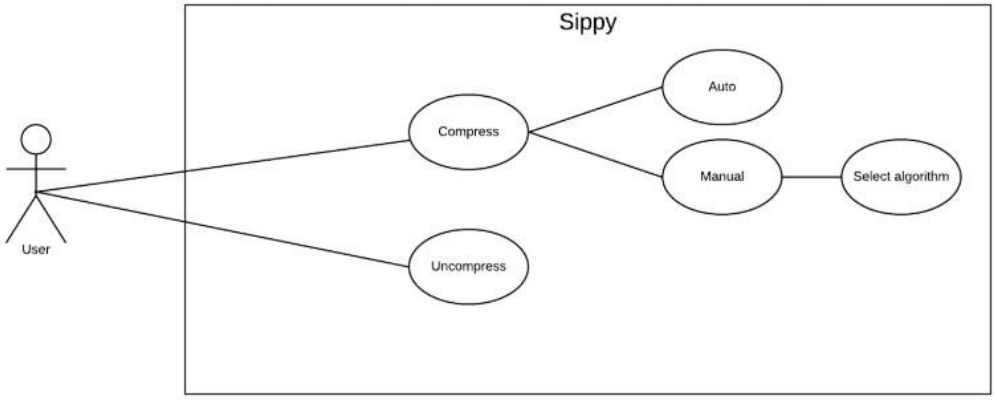
Al executar Sippy, i després de veure el missatge inicial, se'ns ofereixen dues opcions, comprimir i descomprimir.

Comprimir: Al seleccionar comprimir el programa ens sol·licitarà el path del arxiu a comprimir, de l'estil ../a.txt o ../a.ppm, són els dos tipus d'arxius suportats per Sippy en aquesta versió 1.0. Indiquem el path i el programa ens torna a oferir dues opcions, compressió manual o automàtica.

Manual: El programa ens oferirà 4 opcions, els 4 algoritmes que implementa Sippy, LZSS, LZ78 i LZW per a txt i JPEG per a ppm.
Seleccionem l'algorisme que volem utilitzar i començarà la compressió. Per a finalitzar ens mostra les estadístiques de compressió.

Automàtica: Sippy decideix per nosaltres quin algorisme usar per a la compressió tenint en compte quin tipus d'arxiu hem introduït.

Descomprimir: Al seleccionar descomprimir el programa ens sol·licitarà el path del arxiu a descomprimir, de l'estil ../a.sippy, l'indiquem i començarà la descompressió, per a finalitzar ens mostra les estadístiques de descompressió.



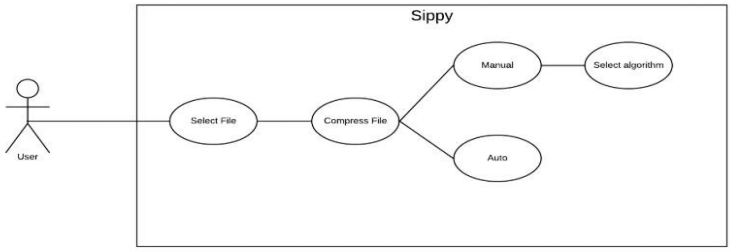
Cas d'us Compressió d'un arxiu

Al seleccionar comprimir el programa ens sol·licitarà el path del arxiu a comprimir, de l'estil ../a.txt o ../a.ppm, són els dos tipus d'arxius suportats per Sippy en aquesta versió 1.0. Indiquem el path i el programa ens torna a oferir dues opcions, compressió manual o automàtica.

Manual: El programa ens oferirà 4 opcions, els 4 algoritmes que implementa Sippy, LZSS, LZ78 i LZW per a txt i JPEG per a ppm.

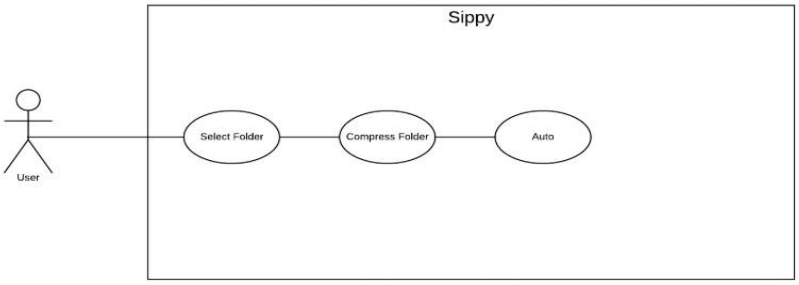
Seleccionem l'algorisme que volem utilitzar i començarà la compressió, per a finalitzar ens mostra les estadístiques de compressió.

Automàtica: Sippy decideix per nosaltres quin algorisme usar per a la compressió tenint en compte quin tipus d'arxiu hem introduït.



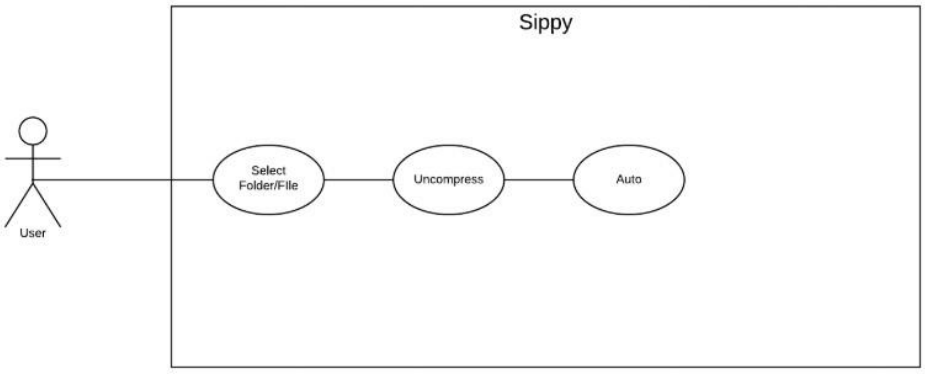
Cas d'us Compressió d'una carpeta

Al seleccionar comprimir el programa ens sol·licitarà el path del arxiu a comprimir, si indiquem una carpeta, el programa la comprimirà sense sol·licitar l'algorisme de compressió a usar, es a dir de forma automàtica.



Cas d'ús descompressió

Descomprimir: Al seleccionar descomprimir el programa ens sol·licitarà el path del arxiu a descomprimir, de l'estil ../a.sippy, l'indiquem i començarà la descompressió, per a finalitzar ens mostra les estadístiques de descompressió.



2. Descripció detallada de les classes

Nom de la clase: **BaseAlgorithm**

Explicació: Interfície per a la implementació dels mètodes comprimir i descomprimir dels algorismes, i per a la implementació del mètode de lectura del arxiu, que per defecte llegeix els bytes a través del mètode Files.readAllBytes().

Nom de la clase: **Algorithm**

Explicació: Enum amb tots els algorismes disponibles al sistema i relacionats amb la seva corresponent id.

Atributs:

- Nom: algorithm ○ Valor per defecte: No té
    - Exemples de valors que pot prendre l'atribut: LZW, LZ78, JPEG, … ○ És estàtic? No
- Nom : id ○ Valor per defecte: No té ○ Descripció: Id amb el que s'identifiquen els algorismes. ○ Exemples de valors que pot prendre l'atribut: 0, 1, 2… ○ És estàtic? No

Mètodes:

- Nom: getId ○ Paràmetres: No té
    - Resultat: retorna l'id de l'algorisme
    - Descripció: Retorna l'id de l'algorisme que s'està utilitzant
- Nom: getAlgorithm ○ Paràmetres: No té ○ Resultat: retorna l'algorisme. ○ Descripció: Retorna l'algorisme que s'esta utilitzant
- Nom: valueOf ○ Paràmetres: id ○ Resultat: un algorisme
    - Descripció: Retorna l'algorisme que s'identifica per el paràmetre d'entrada

Nom de la clase: **File**

Explicació : Classe del model per a mantenir la informació dels itemNC de tipus file.

Mètodes:

- Nom: File ○ Paràmetres: path
    - Descripció: Constructora que assigna un path, crida a la constructora de File(file) amb un nou file(java.io) generat pel path.
- Nom: File ○ Paràmetres: file ○ Descripció: Constructora que assigna un file.
- Nom: getItems ○ Paràmetres: No té ○ Resultat: Llista de files ○ Descripció: Retorna la llista d'ítems file.
- Nom: zipInfo ○ Paràmetres: algorithm ○ Resultat: ItemC
    - Descripció: Retorna un ItemC amb la informació necesaria per a l'operació zip
- Nom: getSize ○ Parametres: No té ○ Resultat: double
    - Descripció: Retorna la mida de la file
- Nom: getDefaultAlgorithm ○ Parametres: No té ○ Resultat: algorithm ○ Descripció: Mètode abstracte, descripció a les subclases.
- Nom: isAlgorithmSupported ○ Parametres: No té ○ Resultat: boolean
    - Descripció: Mètode abstracte, descripció a les subclasses.

Nom de la clase: **FilePpm**

Explicació : Classe del model per a mantenir la informació dels File de tipus ppm.

Mètodes:

- Nom: FilePpm ○ Paràmetres: path ○ Descripció: Constructora que assigna un path.

- Nom: FilePpm ○ Paràmetres: file ○ Descripció: Constructora que assigna un file.
- Nom: getDefaultAlgorithm ○ Paràmetres: No té ○ Resultat: algorithm JPEG
    - Descripció: Mètode que retorna l'algorisme per defecte a l'hora de comprimir arxius ppm.
- Nom: isAlgorithmSupported ○ Paràmetres: algorithm ○ Resultat: boolean
    - Descripció: Retorna true només si l'algorisme indicat per paràmetre és el JPEG

Nom de la clase: **FileTxt**

Explicació : Classe del model per a mantenir la informació dels File de tipus fileppm.

Mètodes:

- Nom: FileTxt ○ Paràmetres: path ○ Descripció: Constructora que assigna un path.
- Nom: FileTxt ○ Paràmetres: file ○ Descripció: Constructora que assigna un file.
- Nom: getDefaultAlgorithm ○ Paràmetres: No té ○ Resultat: algorithm JPEG
    - Descripció: Mètode que retorna l'algorisme per defecte a l'hora de comprimir arxius txt.
- Nom: isAlgorithmSupported ○ Paràmetres: **Algorithm** ○ Resultat: boolean
    - Descripció: Retorna true només si l'algorisme indicat per paràmetre no és el JPEG

Nom de la clase: **Folder**

Explicació : Classe del model per a mantenir la informació dels Folder.

Atributs:

- Nom: items ○ Valor per defecte: No té. ○ Descripció:

Llista d'itemsNC que conté la
folder. ○ És estàtic? No.

Mètodes:

- Nom: Folder ○ Paràmetres:
  path.
  - Descripció: Constructora que assigna un path i crida al mètode populateList() explicat més abaix.
- Nom: Folder ○ Paràmetres:
  file.
  - Descripció: Constructora que assigna un file .
- Nom: getSize() ○ Paràmetres:
  No té. ○ Resultat: double.
  - Descripció: Mètode que retorna la mida de tots els elements de la carpeta.
- Nom: addItem() ○ Paràmetres:
  itemNC. ○ Resultat: Afegeix
  l'ítem a la llista ítems.
  - Descripció: Afegeix l'itemNC indicat per paràmetre a la llista d'items, en cas que sigui el primer element, crea la llista.
- Nom: getItems() ○ Paràmetres:
  No té. ○ Resultat: Llista
  d'ítems.
  - Descripció: Recorre tota la llista ítems obtenint cada ítem que aquesta conté i l'afegeix a la llista que posteriorment retorna.
- Nom: populateList() ○
  Paràmetres: No té.
  - Descripció: mètode usat per a generar els elements ItemNC que s'afegeixen al paràmetre ítem i que estan continguts a la carpeta del file. Es cridat des de el constructor i serveix per tenir sempre en forma de llista tots els ítems continguts a la carpet

Nom de la clase**: ItemC :**

Explicació**:** Representació d'un ítem comprès. S'utilitza principalment per a zipejar, a l'operació inversa només s'utilitza per a guardar l'objecte file.

Atributs:

- Nom: file ○ Valor per defecte: No té. ○ Descripció: fitxer compres.
    - És estàtic? No.
- Nom: method
    - Valor per defecte: No té. ○ Descripció: algorisme usat per a comprimir. ○ És estàtic? No.

Mètodes:

- Nom: ItemC ○ Paràmetres: file.
    - Descripció: Constructora que assigna un file.
- Nom: getFile ○ Paràmetres: No té. ○ Resultat: File.
    - Descripció: Retorna la file.
- Nom: setFile ○ Parametres: file.
    - Descripció: Assigna la file.
- Nom: getMethod ○ Paràmetres: No té. ○ Resultat: Algorithm.
    - Descripció: Retorna l'algorisme usat per a la compressió.
- Nom: setMethod ○ Paràmetres: method.
    - Descripció: Assigna l'algorisme.
- Nom: getSize ○ Paràmetres: No té. ○ Resultat: double. ○ Descripció: Retorna la mida de la file.

Nom de la classe: **ItemNC :**

Explicació**:** Representació d'un ítem no comprès. S'utilitza principalment a l'operació de compressió.

Atributs:

- Nom: file ○ Valor per defecte: No té. ○ Descripció: fitxer comprès. ○ És estàtic? No.

Mètodes:

- Nom: ItemNC ○ Paràmetres: path.
  - Descripció: Constructora que assigna un path.
- Nom: ItemNC
  - Parametres: file.
  - Descripció: Constructora que assigna un file.
- Nom: getFile ○ Paràmetres: No té. ○ Resultat: File.
  - Descripció: Retorna la file.
- Nom: setFile ○ Paràmetres: file.
  - Descripció: Assigna la file.
- Nom: getSize ○ Paràmetres: No té. ○ Resultat: double.
  - Descripció: Operació abstracta que retorna la mida d'un fitxer i si és una folder retorna la suma de mides de tots els seus arxius.
- Nom: getItems ○ Paràmetres: No té. ○ Resultat: Llista de files.
  - Descripció: Operació abstracta que retorna els ítems.
- Nom: create ○ Paràmetres: file. ○ Resultat: ItemNC.
  - Descripció: Si el paràmetre és una folder, la crea, en cas contrari segons la seva extensió crea una filePpm o una fileTxt. ○ És estàtic? Si

Nom de la classe: **Statistics**

Explicació: Estadístiques per la l'operació de zip

Atributs:

- Nom: initialSize ○ Valor per defecte: No té.
  - És estàtic? No.
- Nom: finalSize ○ Valor per defecte: No té.
  - És estàtic? No.
- Nom: elapsedTime ○ Valor per defecte: No té.
  - És estàtic? No.
- Nom: initialTime ○ Valor per defecte: No té.
  - És estàtic? No.

Mètodes:

- Nom: Statistics ○ Paràmetres: initialSize
  - Descripció: Constructora que assigna la mida inicial i inicia el comptador de temps.
- Nom: stopTimer ○ Paràmetres: No té ○ Descripció: Atura el comptador de temps i assigna l'elapsedTime.
- Nom: getInitialSize ○ Paràmetres: No té. ○ Resultat: double.
  - Descripció: Retorna el tamany inicial.
- Nom: getFinalSize ○ Paràmetres: No té ○ Resultat: double.
  - Descripció: Retorna la mida final.
- Nom: setFinalSize ○ Paràmetres: finalSize.
  - Descripció: Assigna la mida final al indicat per paràmetre.
- Nom: getElapsedTime ○ Paràmetres: No té. ○ Resultat: double. ○ Descripció: Retorna l'elapsedTime.

Nom de la clase: **Unzip**

Explicació : Crea un unzipStream amb l'ItemC proveït i el descomprimeix, la classe UnzipStream és responsable de manegar la descodificació de les dades i de la creació dels arxius.

Atributs:

- Nom: item ○ Valor per defecte: No té. ○ Descripció: ítem que es vol descomprimir. ○ És estàtic? No.

Mètodes:

- Nom: Unzip ○ Paràmetres: ítem.
  - Descripció: Assigna l'atribut ítem a l'item que obté per paràmetres.
- Nom: execute ○ Paràmetres: No té. ○ Descripció: Instancia l'unzipStream.

Nom de la clase: **Zip**

Explicació : Crea un ZipStream i afegeix cada ítem dins del ItemC. El qual es comprimirà usant l'algorisme proveït o el per defecte en cas de que s'indiqui que es vol fer automàtic.

Atributs:

- Nom: item ○ Valor per defecte: No té.
  - Descripció: ítem que es vol comprimir.
  - És estàtic? No.
- Nom: algorithm ○ Valor per defecte: No té. ○ Descripció: algorisme amb el que es comprimirà l'arxiu.
  - Valors que pot prendre: LZW,JPEG… ○ És estàtic? No.

Mètodes:

- Nom: Zip ○ Paràmetres: item,algorithm.
  - Descripció: Constructora que assigna l'ítem i l'algorisme als paràmetres de la classe.

- Nom: execute ○ Parametres: No té
    - Descripció: Instància les estadístiques i el ZipStream i s'afegeixen el/s ítems al zipStream, al finalitzar s'atura el timer de les estadístiques i se li assigna el finalSize el cual obtenim del ZipStream.

3. Algorismes i estructures de dades usades:

Algorisme per a la compressió automàtica: Hem desenvolupat un mecanisme per tal que si l'usuari selecciona la metodologia de compressió automàtica, Sippy escogeix per ell, seleccionarà per defecte el JPEG per a arxius ppm i l'LZW per a arxius txt.

ArrayList: Hem usat una llista per guardar els ítems que conté un folder. Hem escollit la estructura de dades ArrayList ja que podem afegir ítems dinàmicament, en contraposició en un vector no. No ens fa falta cap Llista en particular, ja que no necessitem guardar un ordre en els elements a comprimir.

A continuació cada membre del grup ha fet una explicació del seu algorisme:

LZSS

Per a implementar LZSS he creat algunes classes addicionals i així simplificar

el problema:

- LZSS: Classe principal.
- DecodeWindow: Representa la finestra de descompressió.
- EncodedString: Representa una cadena de text en format
- <offset, length>.
- FlagHelper: Classe auxiliar per lidiar amb els flags que indiquen si el
- següent byte a llegir es un literal o un EncodedString en la fase de
- descompresió.
- InputString: Representa l'imput a comprimir.
- WindowBuffer: Representa la finestra de compressió.

Estructures de dades:

Utilitzo principalment Arrays de chars per a les finestres de compressió i descompressió.

StringBuilder per a les cadenes d'input i output. Per a representar els flags uso la classe de Java BitSet.

Flags i estructura del fitxer:

Donat que la unitat mínima d'escriptura per als fitxers és d'1 byte, per a representar flags d'1 bit al escriure-ho al fitxer, aquest augmentava la mida en F bytes on F és el número de flags. Hi ha un flag per cada literal i un percada EncodedString. Per tal de pal·liar aquest problema, s'usa un byte per a representar 8 flags, ara s'utilitzen F/8 bytes per a tots els flags. Els 2 elements que es codifiquen son un literal, és a dir, una subcadena de mida1 on la seva representació és el mateix literal + 1 bit de flag que indica que el byte fa referència a un ASCII char i un parell offset,length codificat en 2 bytes per a les repeticions.

Algorismes:

-Cerca al diccionari: Cerca seqüencial y recorrent el buffer de Search fent match de totes les coincidències de subcadenes que comencin per lookAheadBuffer[0]. He intentat optimitzar aquesta cerca usant l'algoritme de Knuth–Morris–Pratt pero no he obtingut els resultats que esperaba a temps per a la primera entrega.

-Per a la segona entrega s'implementarà aquesta cerca usant arbres binaris de cerca.

Limitacions:

El programa comprimeix i descomprimeix fitxers ASCII correctament. Per a fitxers amb altres charsets només es suporta la compressió. Això és degut a que alhora de descomprimir fitxers UTF-8 per exemple, al llegir un byte, es mira si aquest és més petit de 128, en cas que sí es tracta d'un caràcter ASCII. El problema és que si el byte a llegir és major a 127 jo ho tracto com un caràcter unicode i llegeixo un byte més. Aquí ve el problema, hi ha casos en que el byte llegit és major a 127 però a continuació no ve un byte de caràcter unicode. És a dir que hi ha vegades que caràcters no ASCII també es codifiquen en 1 byte i a hores d'ara no he sabut arreglar el problema.

D'altra banda, donat que les cerques al diccionari es fan de manera seqüencial, la funció de comprimir tarda molt en fitxers més grans de 600KB. Per donar una referència, per un arxiu de menys de 400KB la compressió és ràpida

Referent al ràtio de compressió, per no empitjorar el temps de resposta, he optat per tenir mantenir la finestra de cerca a 1 byte, és a dir 2^8 posicions. Això fa que baixi una mica la compressió, ronda per un 20% aproximadament per fitxerstipus 'Lorem ipsum'. Amb el llibre DON QUIJOTE es comprimeix un 21% i tarda 11 minuts.

JPEG

JPEG és un mètode de compressió i descompressió d'imatges digitals que es basa en aproximacions inexactes i en descartar part de la informació per representar el contingut.

Encoding:

Consisteix en 4 passos:

1. Convertir la representació de colors de la imatge d'RGB a Y'CbCr: Y es la component "luma" i representa la il·luminació; CbCr són les dues components cromàtiques (representen color).
2. Ara, amb la imatge transformada, es divideix en blocs de 8x8 píxels para cada Y, Cb i Cr i, a cada bloc se li aplica la DCT (transformació discreta del cosinus) que fa un espectre de freqüència.
3. Les components de les amplituds de les freqüències es quantifiquen, aquest pas té a veure amb la qualitat.
4. El resultat dels blocs de 8x8 que han passat pels passos anteriors, es llegeixen en zigzag i es comprimeixen seguint Huffman, un algoritme sense pèrdues.

Decoding:

Consisteix en els mateixos passos que al codificar, però fets a la inversa.

1. Es llegeixen els 0's i 1's de la imatge codificada seguint l'algoritme de Huffman, que ens proporciona directament els blocs de 8x8 i se'ls aplica una lectura en zigzag inversa.
2. A continuació, es quantifiquen inversament.
3. Se'ls aplica la IDCT (transformació discreta inversa del cosinus).
4. Es converteixen els colors YCbCr a RGB.

Structures:

Per la realització d'aquesta primera versió del codi, s'han utilitzat principalment:

- Arrays i ArrayLists: per emmagatzemar dades d'un mateix tipus (p.e. els blocs de 8x8, o les matrius redefinides de quantitatiu), degut a la facilitat i rapidesa a l'hora d'accedir als valors.
- HashMaps: per emmagatzemar les entrades de les taules de Huffman, degut a que cada entrada de les taules va associada a una clau única.
- BinaryTrees i Nodes: classes creades per a l'algoritme de Huffman al descomprimir, degut a l'eficiència i rapidesa de la lectura.
- Queue i LinkedList: per emmagatzemar dades de l'entrada d'un arxiu .sippy en ordre, per l'ordenació FIFO que proporciona.

LZW (No s'ha canviat res)

Hashmap: Per tal de guardar la relació String-Int i Int-String.

També podríem usar dues llistes de tamanys simètrics relacionades entre si, però considero que un map és mes òptim per guardar relacions entre ambdós valors.

Encoding:

La compressió consisteix principalment en els passos següents:

- Bucle que recorre totes les posicions del vector d'entrada :
- Agafa el caràcter corresponent del vector de bytes que tenim com a entrada i comprova si l'adició del carácter al stringBuilder(String per anar construint cadenes) es troba al diccionari.
- En cas que no es trobi, obté la posició corresponent al stringBuilder, i en una iteració carrega en la part baixa del buffer els 12 bits d'aquesta posició. Darrerament obtindrà altres 12 bits d'una altra posició i els posarà en la part alta del buffer i escriurà al ArrayOutputStream (sortida) els 3 bytes llegits entre les dues iteracions, un cop escrit, el buffer es reseteja a 0.
- En cada iteració que compleixi la condició anterior (que no estigui al diccionari), si el diccionari té una mida inferior a 4096, insereix la construcció del String + caràcter al diccionari.
- En cas que si contingui aquesta construcció i no entri al primer if, la nova construcció serà la anterior més el nou caràcter llegit.

Decode:

Consisteix en els passos següents:

- Bucle que recorre totes les posicions del vector d'entrada, encara que aquí abans d'entrar al bucle ja ha descodificat el primers 12 bits:
- Al bucle principal, va llegint de 3 en 3 bytes i els va descodificant (va llegint el diccionari pre omplert i el que hem anat omplint al llarg de totes les iteracions anteriors).

LZ78(No s'ha canviat res)

Explicació algorisme:
L'algorisme LZ78 es basa en construir un diccionari basat en repeticions de cadenes de caràcters. La descodificació és tornar a reconstruir aquest diccionari i seguir la "ruta" per a reconstruir una cadena de caràcters.
Quant a la codificació, l'algorisme recorre caràcter a caràcter el string que es vol codificar, per cada caràcter es comprova si ja ha existit en el document, en cas que no sigui així, s'afegeix una entrada a l'output amb l'últim caràcter del string no trobat i apuntant a la posició anterior (que al seu torn apuntés a una posició anterior amb un altre caràcter, fins que s'arriba al 0, i això construeix el string d'aquesta posició). En cas de que sigui afirmatiu, es passa al següent caràcter i es torna a comprovar si existeix.

4. Relació/llista de les classes implementades per cada membre del grup

Yaiza Cano Duarte: JPEG i diverses classes usades per al JPEG .

Miguel Ángel Cabrera: LZSS i diverses classes usades per al LZSS .

Víctor Blanco García: Automatic, BaseAlgorithm, LZ78, Pair, Algorithm, Constants StreamControllerImpl, StreamController, UnZipStreamImpl, UnZipStream, ZipStream, UnZipStream, ItemMapper, File, FilePpm,FileTxt, Folder, InterfaceController, Log.

Sergio Vázquez Montes de Oca: LZW, FilesControllerImpl, FilesController, FolderBO , ItemBO , DataFactory , ItemC , ItemNC , Statistics , Transaction , UnZipTransaction, ZipTransaction, ConsoleApp, InterfacePane2, Bytes, StatisticsUtils.

Les classes del diagrama UML van ser definides a classe per tots els membres del grup i implementades posteriorment com es mostra al llistat anterior.

5.1 Manual d'us de l'aplicació per Consola

1. Al executar Sippy, i després de veure el missatge inicial, se'ns ofereixen dues opcions, comprimir i descomprimir.

    1.1 Comprimir: Al seleccionar comprimir el programa ens sol·licitarà el path del arxiu a comprimir, de l'estil ../a.txt o ../a.ppm, són els dos tipus d'arxius suportats per Sippy en aquesta versió 1.0. Indiquem el path i el programa ens torna a oferir dues opcions, compressió manual o automàtica.

        1.1.1 Manual: El programa ens oferirà 4 opcions, els 4 algoritmes que implementa Sippy, LZSS, LZ78 i LZW per a txt i JPEG per a ppm. Seleccionem l'algorisme que volem utilitzar i començarà la compressió, per a finalitzar ens mostra les estadístiques de compressió.

        1.1.2 Automàtica: Sippy decideix per nosaltres quin algorisme usar per a la compressió tenint en compte quin tipus d'arxiu hem introduït.

    1.2 Descomprimir: Al seleccionar descomprimir el programa ens sol·licitarà el path del arxiu a descomprimir, de l'estil ../a.sippy, l'indiquem i començarà la descompressió, per a finalitzar ens mostra les estadístiques de descompressió.

5.2. Manual d'us de la Interficie

1. Al executar Sippy veiem lo següent:

2. A la pantalla principal de l'aplicació disposem de les dos funcionalitats principals, clicant sobre Compress veuríem lo següent:

3. Ara hem de seleccionar l'arxiu/carpeta a comprimir i un cop estigui seleccionat Sippy ens mostrarà lo següent:

3.1 Seleccionem manual com a opció de compressió

Aquí tenim la pantalla final, on podem seleccionar l'algorisme per comprimir, un cop seleccionat si ho desitgem podem donar-li un nom diferent al original omplint el camp d'Output. Al prémer sobre el botó de Sippejar, ens comprimirà l'arxiu i ens generarà les estadístiques.

3.2 Seleccionem automàtic:

Aquí no necessitem res més, està pensat per a ser una funcionalitat plenament automàtica, al polsar sobre el botó comprimirà i generarà estadístiques.

4. Descomprimir:

Segueix els mateixos passos que a Comprimir però al seleccionar el path, ens apareix lo següent.

Igualment, al polsar sobre el botó, es descomprimirà i generarà estadístiques.

6.Llista de funcionalitats

1. Comprimir i descomprimir fitxers
    - Compressió manual: Compressió d'un fitxer txt escollin l'algorisme desitjat entre les següents opcions: LZSS,LZT8,LZW.
    - Compressió automàtica: Compressió d'un fitxer txt usant un algorisme escollit de forma automàtica
    - Descompressió: Descompressió automàtica d'un fitxer usant el mateix algorisme que s'ha usat per a comprimir-lo.
2. Comprimir i descomprimir carpetes
    - Compressió manual: Compressió d'un fitxer txt escollin l'algorisme desitjat entre les següents opcions: LZSS,LZT8,LZW.
    - Compressió automàtica: Compressió d'una carpeta amb un algorisme escollit de forma automàtica
    - Descompressió: Descompressió automàtica d'una carpeta usant el mateix algorisme que s'ha usat per a comprimir-la.
3. Generació d'estadístiques de compressió/descompressió

    Generació de les següents dades:
        - Elapsed Time: Temps que triga en comprimir.
        - Mida Inicial: Mida inicial de l'arxiu/carpeta.
        - Mida final: Mida final de l'arxiu/carpeta.
        - Rati de compressió.
4. Visualització de la seva corresponent descompressió després d'aplicar el procés de descompressió.

    Al descomprimir un fitxer, s'obre el resultat en el editor de text/visualitzador d'imatges predeterminats del sistema