Full de càlcul senzill

Identificador de l'equip: 1.1

Joan Aluja Oraá: joan.aluja@estudiantat.upc.edu
Marc Clapés Marana: marc.clapes.marana@estudiantat.upc.edu
Marc Duch Buechler: marc.duch@estudiantat.upc.edu
Andreu Orensanz Bargalló: andreu.orensanz@estudiantat.upc.edu

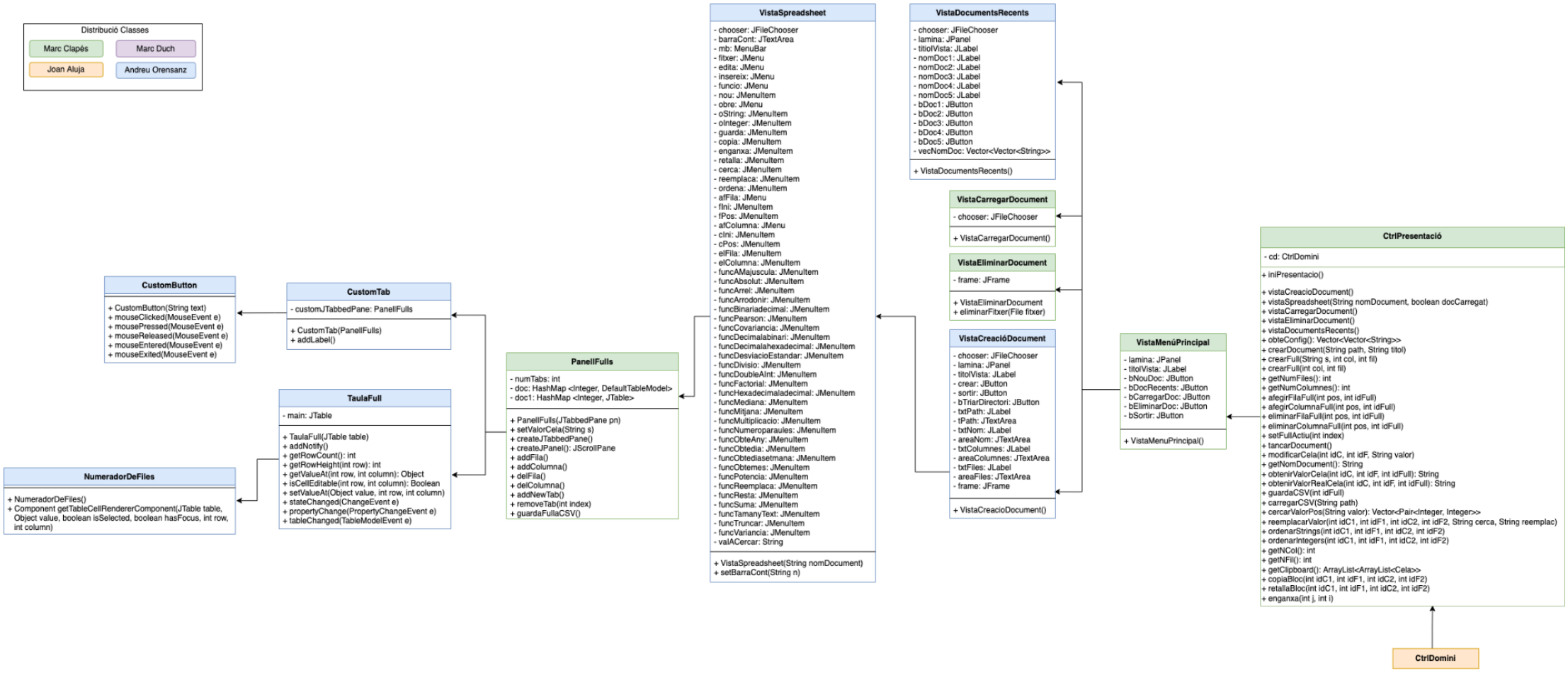
Versió del lliurament: 2.1

2ª entrega PROP

Índex

1. Diagrama de les classes i controladors de la capa de presentació

A continuació mostrem el disseny del diagrama de les classes i controladors de la capa de presentació. El diagrama es pot trobar detallat a la carpeta DOCS.



1.1. Explicació de les classes de presentació

Vistes:

1.1.1. VistaMenuPrincipal

Aquesta vista és l'encarregada de mostrar la primera pantalla que apareix a l'executar el nostre sistema que consisteix en un menú amb un text amb el títol del programa i 5 botons. 4 d'ells ens portaran a vistes diferents on es pot crear un nou document, carregar-ne un, eliminar-ne un o obrir-ne un recentment obert. L'altre botó servirà per sortir i tancar el programa.

- Atributs:
  - JPanel lamina: Panell on s'inclou tots els elements de la finestra
  - JLabel titolVista: Títol de la finestra
  - JButton bNouDoc: Botó per anar a la finestra de creació de document
  - JButton bDocRecentsBotó: per anar a la finestra de documents recentment oberts
  - JButton bCarregarDoc: Botó per anar a la finestra de carregar un document
  - JButton bEliminarDoc: Botó per anar a la finestra d'eliminar un document
  - JButton bSortir: Botó de sortir per a tancar el programa

- Mètodes:
  - VistaMenuPrincipal(): Constructora de la finestra del menú principal

1.1.2. VistaCreacioDocument

Aquesta vista és l'encarregada de crear un document. S'indica clarament 4 camps de text per indicar la localització del document (*path*), el nom del document, les files i les columnes que es vol tenir inicialment en els fulls. Només es pot crear document si no s'indica almenys la localització on estarà guardat, si no és així salta un missatge d'error conforme no s'ha indicat el *path*. La localització del document es pot indicar manualment escrita a la barra de text o es pot seleccionar el directori fent clic al botó *triar directori*, on s'obre l'explorador de documents i es pot seleccionar el directori a guardar el document. A sota de la vista s'hi proporciona també un botó per a tornar al menú principal.

- Atributs:
  - JFileChooser chooser: Finestra de selecció del directori on es vol crear el document creat
  - JPanel lamina: Panell on s'inclou tots els elements de la finestra
  - JLabel titolVista: Panell on s'inclou tots els elements de la pantalla
  - JButton crear: Botó de crear document
  - JButton sortir: Botó de tornar a la pantalla del menú principal

- JButton bTriarDirectori: Botó per a obrir l'explorador d'arxius i poder seleccionar el directori on guardar el document
- JLabel txtPath: Text indicant que la barra de text del costat és per a introduir la localització del directori on poder guardar el document
- JTextArea areaPath: Àrea de text per a introduir la localització del directori on poder guardar el document
- JLabel txtNom: Text indicant que la barra de text del costat és per a introduir el nom del document que es vol crear
- JTextArea areaNom: Àrea de text per a introduir el nom del document que es vol crear
- JLabel txtColumnes: Text indicant que la barra de text del costat és per a introduir el número de columnes del document que es vol crear
- JTextArea areaColumnes: Àrea de text per a introduir el número de columnes del document que es vol crear
- JLabel txtFiles: Text indicant que la barra de text del costat és per a introduir el número de files del document que es vol crear
- JTextArea areaFiles: Àrea de text per a introduir el número de files del document que es vol crear
- JFrame frame: Pantalla d'error que apareix quan es vol crear un document sense path

- Mètodes:
  - VistaCreacioDocument(): Constructora de la finestra de creació de document

1.1.3. VistaEliminarDocument

Aquesta vista s'encarrega d'obrir l'explorador de documents, on es pot seleccionar únicament fitxers amb els formats que suporta el nostre sistema que es poden eliminar. A sota a la dreta hi ha un botó de cancel·lar i un d'esborrar. El primer tornarà la VistaMenuPrincipal i el segon eliminarà el fitxer seleccionat i tornarà al menú altre cop.

- Atributs:
  - JFrame frame: Finestra de resultat conforme s'ha esborrat o bé no existeix el document seleccionat

- Mètodes:
  - VistaEliminarDocument(): Constructora de la finestra d'eliminar document
  - EliminarFitxer(File fitxer): S'esborra del sistema el fitxer passat per paràmetre i apareix un missatge conforme s'ha realitzat correctament. Surt un missatge d'error en cas que el fitxer no existeixi.

1.1.4. VistaCarregarDocument

Aquesta vista s'encarrega de carregar un fitxer guardat a l'ordinador i obrir-lo. La vista obre l'explorador d'arxius on es podran seleccionar únicament fitxers que el nostre sistema és capaç d'obrir. Es proporciona un botó de cancel·lar que tornarà a la VistaMenuPrincipal i un botó d'obrir que farà que s'obri a la VistaSpreadsheet el fitxer seleccionat.

- Atributs:
  - JFileChooser chooser: Finestra de selecció de l'arxiu que es vol carregar al nostre full de càlcul
- Mètodes:
  - VistaCarregarDocument(): Constructora de la finestra de carregar document

1.1.5. VistaDocumentsRecents

Aquesta vista és l'encarregada de mostrar els 5 fitxers més recentment utilitzats, per ordre de més recent a menys. Es mostra el path i el nom del fitxer i un botó per orbrir-lo. A sota de la vista s'hi proporciona també un botó per a tornar al menú principal.

- Atributs:
  - JPanel lamina: Panell on s'inclou tots els elements de la finestra
  - JLabel titolVista: Títol de la finestra
  - JLabel noRecents: Text quan no hi ha documents recents en pantalla
  - JLabel nomDoc1: Nom i data del document més recent
  - JLabel nomDoc2: Nom i data del segon document més recent
  - JLabel nomDoc3: Nom i data del tercer document més recent
  - JLabel nomDoc4: Nom i data del quart document més recent
  - JLabel nomDoc5: Nom i data del cinquè document més recent
  - JButton bDoc1: Botó d'obrir el document més recent
  - JButton bDoc2: Botó d'obrir el segon document més recent
  - JButton bDoc3: Botó d'obrir el tercer document més recent
  - JButton bDoc4: Botó d'obrir el quart document més recent
  - JButton bDoc5: Botó d'obrir el cinquè document més recent
  - Vector<Vector<String>> vecNomDoc: Vector on s'hi guarda el nom, la data i el path dels 5 documents més recentment oberts
  - JButton bSortir: Botó de retornar a la pantalla del menú principal

- Mètodes:
  - VistaCarregarDocument(): Constructora de la finestra de documents recents.

1.1.6. VistaSpreadsheet

Aquesta vista és l'encarregada de mostrar el document, amb els diversos fulls creats. Inicialment està format per un full de càlcul on les columnes van ordenades alfabèticament i les files ordenades numèricament. A sota es pot canviar o generar més fulls amb els panells que hi ha sota el full. A sobre el full hi trobem una barra de text on hi apareixerà el valor d'usuari de la cel·la seleccionada, mentre que a la cel·la hi apareixerà només el valor real. A dalt de tot d'aquesta vista hi trobem una barra de menú on s'hi pot seleccionar les funcionalitats principals que realitza el nostre full de càlcul, així com guardar el document actual i obrir-ne o crear-ne un de nou.

- Atributs:
  - JFileChooser chooser: Finestra de selecció de l'arxiu que es vol obrir
  - JTextArea barraCont: Àrea de text del contingut de la cel·la seleccionada
  - JMenuBar mb: Barra de menús que inclou la majoria de funcionalitats del full de càlcul classificades en menús desplegables
  - JMenu fitxer: Menú desplegable amb les opcions de crear o obrir un nou document, o guardar el document que s'està editant
  - JMenu edita: Menú desplegable amb les opcions de copiar, retallar o enganxar cel·les o blocs seleccionats
  - JMenu insereix: Menú desplegable amb les opcions de copiar, retallar o enganxar cel·les o blocs seleccionats
  - JMenu elimina: Menú desplegable amb les opcions d'eliminar columna i eliminar fila
  - JMenu funcio: Menú desplegable amb les opcions de totes les funcions que calcula el nostre full de càlcul
  - JMenuItem nou: Ítem del menú desplegable de Fitxer per a crear un nou document
  - JMenuItem obre: Ítem del menú desplegable de Fitxer per a obrir un nou document
  - JMenu guarda: Ítem del menú desplegable de Fitxer per a guardar el document que s'està editant. Fa de submenú desplegable amb les opcions de guardar a CSV o al format PROP
  - JMenuItem aCSV: Ítem del submenú Guarda per a guardar el document que s'està editant a CSV
  - JMenuItem aPROP: Ítem del submenú Guarda per a guardar el document que s'està editant a PROP
  - JMenuItem copia: Ítem del menú Edita per a copiar la cel·la o el bloc seleccionat
  - JMenuItem enganxa: Ítem del menú Edita per a enganxar la cel·la o el bloc seleccionat
  - JMenuItem retalla: Ítem del menú Edita per a tallar la cel·la o el bloc seleccionat

- JMenuItem cerca: Ítem del menú Edita per a cercar un valor al full seleccionat
- JMenuItem reemplaca: Ítem del menú Edita per a reemplaçar un valor al full seleccionat
- JMenu ordena: Submenú ítem del menú Edita per a ordenar el contingut de les cel·les d'un bloc seleccionat
- JMenuItem oString: Ítem del submenú Ordena per a ordenar exclusivament Strings
- JMenuItem oInteger: Ítem del submenú Ordena per a ordenar exclusivament Integers
- JMenu afFila: Submenú ítem del menú Insereix per a afegir files o bé a l'inici o a la posició de la cel·la seleccionada
- JMenuItem fIni: Ítem del submenú Fila per a afegir una fila a l'inici del full
- JMenuItem fPos: Ítem del submenú Fila per a afegir una fila a sota de la posició de la cel·la que està seleccionada
- JMenu afColumna: Submenú ítem del menú Insereix per a afegir columnes o bé a l'inici o a la posició de la cel·la seleccionada
- JMenuItem cIni: Ítem del submenú Columna per a afegir una columna a l'inici del full
- JMenuItem cPos: Ítem del submenú Columna per a afegir una columna a la dreta de la posició de la cel·la que està seleccionada
- JMenuItem elFila: Ítem del menú Elimina per a eliminar la fila de la cel·la seleccionada
- JMenuItem elColumna: Ítem del menú Elimina per a eliminar la columna de la cel·la seleccionada
- JMenuItem funcAMajuscula: Ítem del menú Funció per a introduir la funció A Majúscula
- JMenuItem funcAbsolut: Ítem del menú Funció per a introduir la funció Absolut
- JMenuItem funcArrel: Ítem del menú Funció per a introduir la funció Arrel
- JMenuItem funcArrodonir: Ítem del menú Funció per a introduir la funció Arrodonir
- JMenuItem funcBinariadecimal: Ítem del menú Funció per a introduir la funció Binari a Decimal
- JMenuItem funcPearson: Ítem del menú Funció per a introduir la funció Correlació Pearson
- JMenuItem funcCovariancia: Ítem del menú Funció per a introduir la funció Covariància
- JMenuItem funcDecimalabinari: Ítem del menú Funció per a introduir la funció Decimal a Binari

- JMenuItem funcDecimalahexadecimal: Ítem del menú Funció per a introduir la funció Decimal a Hexadecimal
- JMenuItem funcDesviacioestandar: Ítem del menú Funció per a introduir la funció Desviació estàndard
- JMenuItem funcDivisio: Ítem del menú Funció per a introduir la funció Divisió
- JMenuItem funcDivisio: Ítem del menú Funció per a introduir la funció Divisió
- JMenuItem funcFactorial: Ítem del menú Funció per a introduir la funció Factorial
- JMenuItem funcHexadecimaladecimal: Ítem del menú Funció per a introduir la funció Hexadecimal a Decimal
- JMenuItem funcMediana: Ítem del menú Funció per a introduir la funció Mediana
- JMenuItem funcMitjana: Ítem del menú Funció per a introduir la funció Mitjana
- JMenuItem funcMultiplicacio: Ítem del menú Funció per a introduir la funció Multiplicació
- JMenuItem funcNumeroparaules: Ítem del menú Funció per a introduir la funció Número paraules
- JMenuItem funcObteany: Ítem del menú Funció per a introduir la funció Obté any
- JMenuItem funcObtedia: Ítem del menú Funció per a introduir la funció Obté dia
- JMenuItem funcObtediasetmana: Ítem del menú Funció per a introduir la funció Obté dia setmana
- JMenuItem funcObtemes: Ítem del menú Funció per a introduir la funció Obté mes
- JMenuItem funcPotencia: Ítem del menú Funció per a introduir la funció Potència
- JMenuItem funcReemplaca: Ítem del menú Funció per a introduir la funció Reemplaça
- JMenuItem funcResta: Ítem del menú Funció per a introduir la funció Resta
- JMenuItem funcSuma: Ítem del menú Funció per a introduir la funció Suma
- JMenuItem funcTamanytext: Ítem del menú Funció per a introduir la funció Tamany text
- JMenuItem funcTruncar: Ítem del menú Funció per a introduir la funció Truncar
- JMenuItem funcVariancia: Ítem del menú Funció per a introduir la funció Variancia
- String valACercar: Guarda el valor a cercar a l'indicar l'opció de cercar

- Mètodes:
  - VistaSpreadsheet(String nomDocument, boolean DocCarregat): Constructora de la finestra del document amb els fulls de càlcul. S'entra per paràmetre el nom del document i un boolean conforme el document és nou o ha sigut carregat dels fitxers de l'ordinador
  - setBarraCont(String n): Posa a la barra de continguts el contingut del String passat per paràmetre

Altres classes:

1.1.7. CustomTab

La classe CustomTab es l'encarregada de reimplementar els botons de canvi de full. Aquesta reimplementació fa que es pugui tancar fulls afegint una creu al panell i obrir-ne i seleccionar-ne de nous.

- Atributs:
  - PanellFulls panellFulls: Panells de canvis de Full amb botons

- Mètodes:
  - CustomTab(PanellFulls customJTabbedPane): Constructora de CustomTab
  - addLabel(): Afegeix el nom al full

- Classes:
  class CustomButton extends JButton implements MouseListener: Classe per a reimplementar el botó de tancar un panell amb full

1.1.8. CustomButton

Classe per a reimplementar el botó de tancar un panell amb full

- Mètodes:
  - CustomButton(String text): Constructora de CustomButton
  - mouseClicked(MouseEvent e): Tanca el full quan es fa clic al botó amb la creu
  - mouseEntered(MouseEvent e): Pinta de color vermell la creu quan es passa el ratolí per sobre
  - mouseExited(MouseEvent e): Pinta de color negre la creu quan no es passa el ratolí per sobre

1.1.9. NumeradorDeFiles

La classe NumeradorDeFiles es tracta d'una reimplementació de l'objecte *DefaultTableCellRenderer* de la llibreria de *Swing*. És l'encarregada de dissenyar un sistema per a poder numerar les files i poder canviar de color un cop ha estat seleccionada una cel·la d'una fila.

- Mètode:
  - NumeradorDeFiles(): Constructora de NumeradorDeFiles
  - getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column): S'obté un Component de NumeradorDeFiles amb la capacitat de poder numerar files d'un JTable i pintar de color vermell i en negreta els números de les files de les cel·les seleccionades.

1.1.10. PanellFulls

La classe PanellFulls es tracta d'una reimplementació de l'objecte JTabbedPane que ens proporciona la llibreria *Swing* de Java. Aquesta classe és l'encarregada de crear els panells que permeten la creació i canvi de fulls a seleccionar. Dins d'aquests panells s'hi trobarà el full de càlcul determinat amb la capacitat de poder desplaçar-se cap a totes direccions gràcies a les dos barres de desplaçament que ens aporta l'objecte *JScrollPane* de *Swing*.

- Atributs:
  - int numTabs: Número de panells de Full
  - HashMap <Integer, DefaultTableModel> doc: Estructura HashMap de Integer i DefaultTableModel per a guardar totes DefaultTableModel dels fulls creats
  - HashMap <Integer, JTable> doc1: Estructura HashMap de Integer i JTable per a guardar tots els JTable dels fulls creats
  - String copia: Guarda el valor String copiat
  - String copiat: Determina si s'ha copiat una cel·la o un bloc
  - ArrayList<ArrayList<Cela>> c: Guarda el contingut de les cel·les copiades/retallades
  - int FullCopiat: Guarda l'índex del full on es copia/retalla contingut

- Mètodes:
  - PanellFulls(JTabbedPane pn, boolean docCarregat): Constuctora de PanellFulls
  - setValorCela(String s): Setter d'una cel·la de la taula amb el contingut String passat per paràmetre
  - createJTabbedPane(): Es creen dos JTabbedPane. Un que inclou el primer full que es crea predeterminadament al crear el document amb el nom de "Full 1" i un altre amb nom "+" que al fer-li clic crear un nou JTabbedPane amb un nou Full.
  - JScrollPane createJPanel(int i): Crea un panell JScrollPane que inclou un objecte TaulaFull. Es pot fer scroll en totes direccions per a veure el contingut complet del Full.
  - addFila(): S'afegeix una fila al full seleccionat
  - addFilaIni(): S'afegeix una fila a l'inici del full seleccionat
  - addColumna(): S'afegeix una columna al full seleccionat

- addColumnaIni(): S'afegeix una columna a l'inici del full seleccionat
- delFila(): S'elimina una fila del full seleccionat
- delCol(): S'elimina una columna del full seleccionat
- copiar(): Retalla el bloc/cel·la seleccionada
- enganxa(): Enganxa a partir de la cel·la seleccionada l'últim que s'ha copiat o retallat
- retalla(): Retalla el bloc/cel·la seleccionada
- addNewTab(): Afegeix un panell de full al document amb nom "Full n" on n és el número de l'últim full creat més 1
- removeTab(int index): S'esborra un panell de full amb l'índex passat per paràmetre
- guardaFullaCSV(): Guarda el Full seleccionat a format .csv
- carregaValorCeles(): Afegeix valor a totes les cel·les al Full amb els valors de la capa de domini
- reemplacaValor(String cercat, String reempl): Reemplaça el valor cercat per reempl dins de les cel·les seleccionades
- ordenaPerColumnes(): Ordena per columnes el bloc de cel·les seleccionat

1.1.11. TaulaFull

La classe TaulaFull es tracta d'una reimplementació de l'objecte *JTable* de *Swing*. S'encarrega de crear una taula amb la capacitat de que les files vagin ordenades numèricament, les columnes alfabèticament i que es pugui redimensionar les columnes sense problemes. També permet poder seleccionar únicament una cel·la i no una fila sencera com és en cas de l'objecte original *JTable*.

- Atributs:
  - JTable main: Taula JTable que representa el full
- Mètodes:
  - TaulaFull(JTable table): Constructora de TaulaFull on reimplementa el JTable passat per paràmetre
  - addNotify(): S'actualitza el JViewport al fer scroll per veure els continguts de TaulaFull
  - getRowCount(): Retorna el número de files de la taula
  - getRowHeight(int row) Retorna l'alçada (mida en pantalla) d'una fila. Es delega aquest mètode a la taula principal
  - getValueAt(int row, int column): Aquest mètode obté el valor que se li posarà a les cel·les de la primera columna que seran un valor numèric fixe definint la numeració de files
  - isCellEditable(int row, int column): No permet editar les cel·les de la primera columna, que és on hi haurà la numeració de les cel·les
  - setValueAt(Object value, int row, int column): No fa res ja que la nova taula ignora el model

- stateChanged(ChangeEvent e): Sincronitza el scrolling de la fila de la taula sincronitzada amb la taula principal
- propertyChange(PropertyChangeEvent e): Implementa el PropertyChangeListener
- tableChanged(TableModelEvent e): Implementa el TableModelListener

1.2. Explicació del controlador de presentació

1.2.1. CtrlPresentació

El controlador de Presentació s'encarrega de fer de comunicador entre les vistes de la capa de presentació. També és el que transmet a la capa de presentació les dades de les capes inferiors.

- Atributs:
  - CtrlDomini cd: Instància del controlador de domini

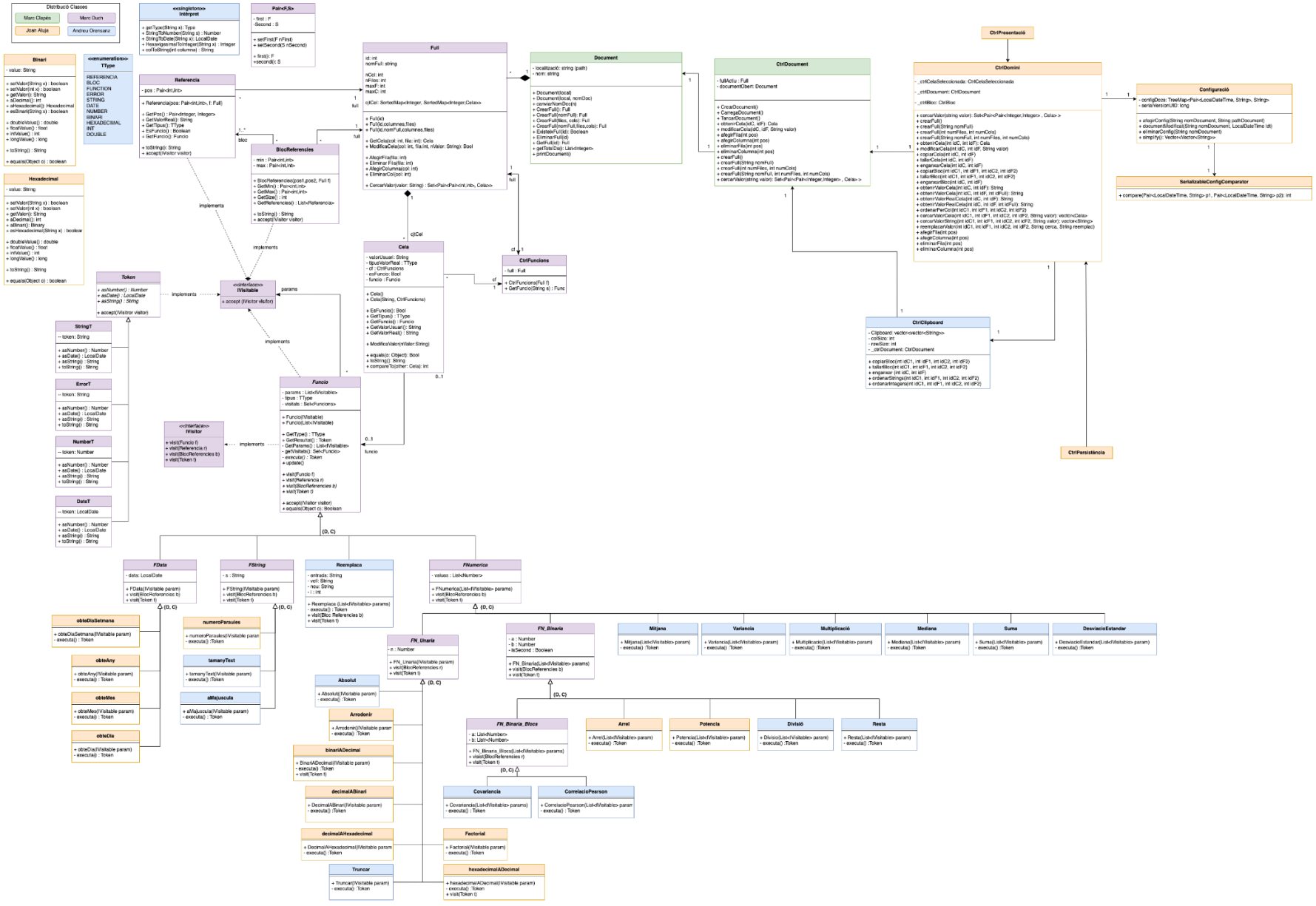
- Mètodes:
  - iniPresentacio(): Mostra en pantalla la finestra del menú principal
  - vistaCreacioDocument(): Mostra en pantalla la finestra de creació del Document.
  - vistaSpreadsheet(String nomDocument, boolean docCarregat): Mostra en pantalla la finestra del Document amb els Fulls
  - vistaCarregarDocument(): Mostra en pantalla la finestra per a seleccionar un arxiu per a carregar
  - vistaEliminarDocument(): Mostra en pantalla la finestra per a seleccionar un arxiu per a eliminar
  - vistaDocumentsRecents(): Mostra en pantalla la finestra que mostra els documents recentment oberts
  - obteConfig(): Crida al mètode obtenirConfig de CtrlDomini i en retorna el valor en Vector<Vector<String>> que en calcula
  - crearDocument(String path, String titol): Crida al mètode crearDocument de CtrlDomini
  - crearFull(String s, int col, int fil): Crida al mètode crearFull amb títol del full i el número de files i columnes com a paràmetre de CtrlDomini
  - crearFull(int col, int fil): Crida al mètode crearFull amb el número de files i columnes com a paràmetre de CtrlDomini
  - getNumFiles(): Crida al mètode getNumFiles de CtrlDomini i en retorna el número de files
  - getNumColumnes(): Crida al mètode getNumColumnes de CtrlDomini i en retorna el número de columnes
  - afegirFilaFull(int pos, int idFull): Crida al mètode afegirFilaFull de CtrlDomini on afegeix una fila al full idFull sota de la posició de cel·la pos

- afegirColumnaFull(int pos, int idFull): Crida al mètode afegirColumnaFull de CtrlDomini on afegeix una columna al full idFull a la dreta de la posició de cel·la pos
- eliminarFilaFull(int pos, int idFull): Crida al mètode eliminarFilaFull de CtrlDomini on elimina una fila al full idFull a la posició pos
- eliminarColumnaFull(int pos, int idFull): Crida al mètode eliminarColumnaFull de CtrlDomini on elimina una columna al full idFull a la posició pos
- setFullActiu(int index): Crida al mètode setFullActiu de CtrlDomini on posa a actiu el full amb l'index passat per paràmetre
- tancarDocument(): Crida al mètode tancarDocument de CtrlDomini que tanca el document
- modificarCela(int idC, int idF, String valor): Crida al mètode modificarCela de CtrlDomini que modifica la cel·la a idC i idF amb el valor passat per paràmetre
- getNomDocument(): Crida al mètode getNomDocument de CtrlDomini que retorna el nom del Document
- obtenirValorCela(int idC, int idF, int idFull): Crida al mètode obtenirValorCela de CtrlDomini que retorna el valor de la cel·la a idC i idF del full idFull
- obtenirValorRealCela(int idC, int idF, int idFull): Crida al mètode obtenirValorRealCela de CtrlDomini que retorna el valor real de la cel·la a idC i idF del full idFull
- guardaCSV(int idFull): Crida al mètode guardaCSV de CtrlDomini que guarda a format .csv el full amb l'índex idFull
- carregarCSV(String path): Crida al mètode carregarCSV de CtrlDomini que carrega al full de càlcul el fitxer en format .csv al path passat per paràmetre
- cercarValorPos(String valor): Crida al mètode cercarValorPos de CtrlDomini que retorna les posicions de les cel·les que contenen el valor passat per paràmetre
- reemplacarValor(int idC1, int idF1, int idC2, int idF2, String cerca, String reemplac): Crida al mètode reemplacarValor de CtrlDomini que reemplaça el valor cerca pel valor reemplac d'entre el bloc de cel·les que hi ha entre idC1,idF1 i idC2 i idF2
- ordenarStrings(int idC1, int idF1, int idC2, int idF2): Crida al mètode ordenarStrings de CtrlDomini que ordena per columnes el bloc de cel·les seleccionat entre les posicions passades per paràmetre
- ordenarIntegers(int idC1, int idF1, int idC2, int idF2): Crida al mètode ordenarIntegers de CtrlDomini que ordena per columnes el bloc de cel·les seleccionat entre les posicions passades per paràmetre

- getNCol(): Crida al mètode getNCol de CtrlDomini que retorna el número de columnes que conté l'ArrayList de dues dimensions del clipboard
- getNFil(): Crida al mètode getNFil de CtrlDomini que retorna el número de files que conté l'ArrayList de dues dimensions del clipboard
- getClipboard(): Retorna l'array de dues dimensions que conté el contingut copiat o retallat
- copiaBloc(int idC1, int idF1, int idC2, int idF2): Crida al mètode copiarBloc del CtrlDomini que copia al clipboard intern a CtrlDomini el contingut de les cel·les entre idC1, idF1, i idC2, idF2
- retallaBloc(int idC1, int idF1, int idC2, int idF2): Crida al mètode retallarBloc del CtrlDomini que copia al clipboard intern a CtrlDomini el contingut de les cel·les entre idC1, idF1, i idC2, idF2 i esborra el contingut en aquell bloc de cel·les.
- enganxa(int j, int i): Crida al mètode enganxar del CtrlDomini que enganxa al bloc començant per j,i el contingut del clipboard

2. Diagrama de les classes i controladors de la capa de domini

A continuació mostrem el disseny del diagrama de les classes i controladors de la capa de domini. El diagrama es pot trobar detallat a la carpeta DOCS.



2.1. Explicació de les classes de domini

2.1.1. Document

La classe Document és la classe que conté tots els elements que formen el full de càlcul, és a dir, fulls i aquests contenen cel·les. S'encarrega principalment de la gestió dels fulls, així com crea-los, eliminar-los i modificar-los.

- Atributs:
  -String localització: Localització on es guarda el Document.
  -String nom: nom amb el qual es guarda el Document.
  -int id: Identificador intern del Document.
  -TreeMap<Integer, Full> Doc: Estructura de dades per guardar el conjunt de Fulls amb valor no-null. És un Treemap de Fulls on cada un d'ells té els seu propi identificador.
- Mètodes:
  -Document(String local): Constructora
  -Document (String local, String nom): Constructora
  -GetLocal(): Retorna la localització del Document
  -GetNom(): Retorna el nom del Document
  -GetID(): Retorna el ID del últim Full creat.
  -canviarNomDoc(String n): Es canvia l'atribut nom de Document per el String passat per paràmetre.
  -GetNumFulls(): Retorna el número de Fulls que té el Document.
  -CrearFull(): Es crea un Full sense passar-li atributs per paràmetre i es retorna el Full creat.
  -CrearFull(String nomFull): Donat un nom es crea un Full i se li atribueixen els atributs restants i es retorna el Full creat.
  -CrearFull(int numCols, int numFiles): Donat un número de Columnes i de Files, es crea un Full i se li atribueixen els atributs restants i es retorna el Full creat.
  -CrearFull(String nomFull, int numCols, int numFiles): Donat un nom, un número de Columnes i de Files, es crea un Full i es retorna el Full creat.
  -ExisteixFull(int id): Retorna True si l'índex del Full passat per paràmetre existeix en el TreeMap, retorna Fals en cas contrari
  -GetFull(int id): Retorna el Full identificat per índex, aquest existeix en el TreeMap.
  -EliminarFull(int id): Si existeix el Full identificar per l'índex passat per paràmetre l'elimina del TreeMap.
  -GetTotsIds(): Retorna una Llista de tots els identificadors dels Fulls que existeixin en el TreeMap

2.1.2. Full

La classe Full representa una sola fulla de càlcul. Aquesta classe és l'encarregada de gestionar les seves cel·les (les que pertanyen al full en si). S'encarrega de modificar les cel·les existents o de afegir-les si no estaven ja al full, ordenar-les, o cercar les cel·les per valors.

- Atributs:
  - int _id: Identificador intern del Full
  - String _nomFull: Nom del Full
  - int _nCol: Número de columnes del Full
  - int _nFil: Número de files del Full
  - int _maxFila: Última fila amb una cel·la
  - int _maxColumna: Última columna amb una cel·la
  - TreeMap<Pair<Integer,Integer>, Referencia> _refs: TreeMap que conté totes les referències a les cel·les del full, TreeMap no permet nulls com a claus o valors
  - CtrlFuncions _cf: Controlador de funcions del full
  - SortedMap<Integer, SortedMap<Integer,Cela> > _cjtCel: Estructura de dades per guardar el conjunt de cel·les amb un valor no-null. Es un diccionari aniuat en un altre diccionari, on les claus del primer diccionari representen les columnes i les claus del diccionari aniuat representen les files per la columna a la qual pertanyen. Només es guarden les cel·les que contenen un string no null

- Mètodes:
  - Full(int id): Constructora amb un paràmetre. Crea un full amb la id indicada de 10c x 15f amb nom "Full id"
  - Full(int id, int columnes, int files) { this(id,"Full " + id, columnes, files): Constructora amb un tres paràmetres. Crea un full amb la id indicada, les dimensions passades com a paràmetres i amb nom "Full id"
  - Full(int id, String nom, int columnes, int files): Constructora amb un quatre paràmetres. Crea una instancia de Full
  - GetCela(int columna, int fila): Donades una columna i una fila, retorna la cela corresponent o null si no n'hi ha cap
  - GetReferencia(int columna, int fila): Retorna la referecia a una cel·la del full. Si no exsisteix, la crea
  - ModificaCela(int col, int fila, String valor): Metode principal per accedir a les cel·les del Full
  - AfegirFila(int pos): Afegir una fila buida en una posició donada, si ja hi havia una fila, aquesta i les seguents seran desplaçades

- EliminarFila(int pos): Elmina la fila especificada, les files posteriors es mouen per tal de mantenir la seva posició relativa
- AfegirCol(int pos): Afegeix una columna a la posició indicada, les columnes posterios es mouran per fer espai
- EliminarColumna(int pos): Elimina la columna indicada, les columnes posteriors es mouen per ocupar el la columna eliminada
- indexosCorrectes(int columna, int fila): Comprova si una columna i una fila estan incloses al full
- indexFilaCorrecte (int fila): Donat l'index d'una fila, llença una excepcio si aquesta no esta inclosa al full
- indexColsCorrecte (int colu): Donat l'index d'una columna, llença una excepcio si aquesta no esta inclosa al full
- shiftCols (int pos, boolean eliminar): Desplaça les columnes posteriors a la posició indicada segons si dita columna s'afegeix o s'elimina
- shiftFiles (SortedMap<Integer,Cela> col, int pos, boolean eliminar): Donat un diccionari de files (representa una sola columna) i una posicio, les files es mouen segons si s'ha afegit una fila, o s'ha eliminat

2.1.3. Cel·la (Cela)

La classe Cel·la (Cela.java) és la classe més granular del sistema. La seva funció és guardar el valor que introdueix l'usuari, sigui un funció o un altre tipus, i retornar, en el cas de les funcions, el seu resultat.

- Atributs:
  - String _valorUsuari: Guarda el valor introduït per l'usuari
  - String _valorReal: Guarda el valor real de la cel·la. Si l'usuari ha introduït una funció el valor real serà el resultat d'aquesta
  - TType _tipusValorReal: Enum que determina el tipus del contingut real
  - CtrlFuncions _cf: Controlador de Funcions, necessari per poder crear referències
  - boolean _esFuncio: Boolean que determina si el contingut de la cel·la és una funció
  - Funcio _funcio: Funció continguda a la cel·la, si _esFuncio es fals, _funcio = NULL

- Mètodes:
  - Cela(): Constructora per defecte de cel·la. Construeix una cel·la buida, no associada a cap full
  - Cela(String valor, CtrlFuncions cf): Constructora per defecte de cel·la
  - ModificaValor(String nValor): Modifica el contingut de la cel·la, si es una funció, la crea
  - delete(): Esborra les referències que te la Cela per facilitar el GC
  - equals(Object o): Override del metode equals() de la classe Object

- toString(): Retorna la representació toSring() de cel·la, el Valor de l'Usuari
- compareTo(Cela other): Compara dues cel·les de forma descendent caracter a caracter. Primer ordena alfabèticament i després numèricament

2.1.4. Interpret

La classe Interpret fa d'analitzador sintàctic de Strings, en determina el seu tipus i transforma els tipus String a Number, Data.

- Atributs:
  - Interpret INSTANCE: Instància única d'Interpret ja que és singleton

- Mètodes:
  - TType getType(String x): Retorna el tipus del String passat com a paràmetre
  - Number StringToNumber(String s): Passa a Number el String passat per paràmetre
  - StringToDate(String s): Donat un string, el converteix a LocalDate amb format (dd/MM/yyyy)
  - HexavigesimalToInteger(String x): Retorna string en Hexavigesimal (base26 en caràcters) a integer
  - colToString(int col): Tradueix un index de columna a la lletra corresponent

2.1.5. Funció

La classe funció és una classe abstracta que conté tots els mètodes necessaris per realitzar totes les operacions del sistema. Aquests mètodes estan implementats a les subclasses que van classificades segons els paràmetres que necessitin les funcions.

- Atributs:
  - List<IVisitable> _params: Llista de parametres de la funció, guardada com a IVisitables(Funció/Referència/BlocReferencies/Token)
  - TType _tipus: Determina el tipus de la dada que retorna
  - Set<Funcio> _visitats = new HashSet<>(): Conjunt de funcions de les qual depèn (visita)

- Mètodes:
  - Funcio(IVisitable parametre): Constructora amb un sol paràmetre
  - Funcio(List<IVisitable> params): Constructora amb llista de paràmetres
  - Token executa(): Mètode abstracte que calcula la funció en sí
  - Token GetResultat(): Recalcula el valor dels parametres i crida a la funció polimorfica executa()

- boolean has2Params(): Comprova que només tingui 2 params
- paramsToValues(): Aplica el patró visitor sobre tots els paràmetres y obté els valors finals de cada parametre
- ini(): Inicialització previa de cada subtipus de funció, es crida abans de fer la conversió de paràmetres a valors
- visit(Funcio f): Comprova que la funció no depèn de si mateixa per calcular el resultat. Executa visit() sobre el resultat de la funció
- visit(Referencia r): Delega la responsabilitat a visit(Funcio) o visit(Token) segons si l'element de la referencia es o no una funció
- boolean equals(Object o): Override de la funció equals d'Object per tal de poder fer ús del HashSet

2.1.6. Binari

La classe Binari gestiona la conversió de tipus entre diferents bases a binari i serveix com a tipus.

- Atributs:
  - String _value: Valor en String de l'objecte Binari
- Mètodes:
  - Binari(): Constructora sense valor. Crea un objecte Binari amb valor 0 -> 0b0
  - Binari(String x): Constructora amb valor. Emmagatzema el valor String x en l'objecte que es crea
  - Binari(int x): Constructora amb valor. Emmagatzema el valor int x en l'objecte que es crea<p>. Si x < 0 el valor es guarda en complement a 2 de 32 bits
  - setValor(String x): Guarda el valor entrat String al valor del objecte Binari. Retorna (TRUE) Si el valor s'ha pogut enmagatzemar (esBinari), si no, FALSE
  - setValor(int x): Guarda el valor entrat int x a un String amb codificacio Binaria<p>. Si x < 0 el valor es guarda en complement a 2 de 32 bits
  - getValor(): Retorna un String amb la codificacio en Binari
  - aDecimal(): Retorna el valor de l'objecte convertit a enter
  - aHexadecimal(): Retorna el valor de l'objecte convertit a Hexadecimal
  - esBinari(String x): Determina si un string es un binari o no. Retorna (TRUE) Si el String es pot interpretar com a binari i comença per 0b/0B, si no, FALSE
  - value(): Retorna _value sense el 0b del principi
  - doubleValue(): Retorna el valor enter del binari com double
  - floatValue(): Retorna el valor enter del binari com float
  - intValue(): Retorna el valor enter del binari
  - longValue(): Retorna el valor enter del binari com long
  - toString(): Retorna la representacio del Binari com a String

- equals(Object other): Comparador entre Binaris. Retorna (TRUE) Si other és Binari i el seu valor és el mateix, si no FALSE

2.1.7. Hexadecimal

La classe Hexadecimal gestiona la conversió de tipus entre diferents bases a hexadecimal i serveix com a tipus.

- Atributs:
  - String _value: Valor en String de l'objecte Binari
- Mètodes:
  - Hexadecimal(): Constructora sense valor. Crea un objecte Hexadecimal amb valor 0 - 0x0
  - Hexadecimal(String x): Constructora amb valor. Emmagatzema el valor String x en l'objecte que es crea
  - Hexadecimal(int x): Constructora amb valor. Emmagatzema el valor x en l'objecte que es crea<p>. Si x < 0 el valor es guarda en complement a 2 de 32 bits
  - setValor(String x): Guarda el valor entrat String x en l'objecte Hexadecimal. Retorna si el valor s'ha pogut guardar.
  - setValor(int x): Guarda el valor entrat int x en l'objecte Hexadecimal<p>. Si x < 0 el valor es guarda en complement a 2 de 32 bits. Retorna si el valor s'ha pogut guardar
  - getValor(): Retorna un String amb la codificacio en Hexadecimal del objecte
  - aDecimal(): Retorna el valor de l'objecte convertit a Decimal
  - aBinari(): Retorna el valor de l'objecte convertit a Binari
  - esHexadecimal(String x): Donat un String, determina si es Hexadecimal. Retorna (TRUE) Si comença en 0x/0X i es pot expressar com a hexadecimal, si no, FALSE.
  - value(): Retorna _value sense el 0x del principi
  - doubleValue(): Retorna el valor enter del hexadecimal com double
  - intValue(): Retorna el valor enter del hexadecimal
  - longValue(): Retorna el valor enter del hexadecimal com long
  - floatValue(): Retorna el valor enter del hexadecimal com float
  - toString(): Retorna la representació en String del Hexadecimal
  - equals(Object other): Comparador entre Hexadecimals. Retorna (TRUE) Si other es Hexadecimal i el seu valor es el mateix, si no FALSE

2.1.8. Pair<F, S>

La classe Pair<F, S> és una classe genèrica per crear tuples de dos elements, utilitzada principalment per simplificar l'ús dels index columna / fila o retornar un Pair(columna, fila) amb la seva cel·la.

- Atributs:

- F _first: Primer valor del Pair
- S _second: Segon valor del Pair

- Mètodes:
  - Pair(F first, S second): Constructora de Pair
  - equals(Object o): Comparador entre pairs
  - toString(): Representació d'un parell com a String

2.1.9. Configuració

La classe Configuració s'encarrega de mantenir dades persistents globals del programa, en el nostre cas s'encarrega de guardar la data d'última modificació, el nom i la localització de cada document que s'ha creat.

- Atributs:
  - configDocs: Conté la estructura de dades on es guarden les dades respectives de cada document. (Nom, Data d'última modificació i path)
  - serialVersionUID: Conté una identificacio unica de la versió de la classe serialitzable.
- Mètodes:
  - Configuracio(): Constructora per defecte que crea una nova instancia buida de la classe.
  - afegirConfig(String nomDocument, String pathDocument): Afegeix un nou document amb nom = nomDocument, path = pathDocument i amb la data i hora actuals.
  - documentModificat(String nomDocument, LocalDateTime ldt): Es canvia la data d'última modificació del document amb nom = nomDocument amb la data i hora ldt.
  - eliminarConfig(String nomDocument): Elimina el document amb nom = nomDocument de la configuració.
  - simplify(): Retorna una matriu d'Strings on cada entrada del primer index es un document i cada document té la data d'última modificació, el nom del document i el path del document. Aquesta matriu esta ordenada per Data i hora de manera descendent (Més recents primers), i si la data coincideix per nom del document.

2.1.10. SerializableConfigComparator

La classe SerializableConfigComparator és una implementació Serialitzable del comparador necessari per a poder ordenar l'estructura de la classe Configuració i poder serialitzar la classe per guardar-la a disc.

- Mètodes:
  compare(Pair<LocalDateTime, String> p1, Pair<LocalDateTime, String> p2): Retorna -1 si p1.first() > p2.first(), 1 si p1.first() < p2.first(). I si p1.first() == p2.first() retorna 1 si p1.second() > p2.second(), -1 si p1.second() < p2.second() i 0 si son iguals.

2.1.11. Referencia

La classe Referencia representa una referencia a una cel·la (Cela) d'un full

- Atributs:
  - pos: Una parella d'enters que representa la posició de la cel·la a la que fa referència
  - full: Full al que pertany la referència
- Mètodes:
  - Referencia(Pair<int,int> pos, Full f): Constructora, crea una referencia de la posició indicada que pertany al full passat com a paràmetre.
  - GetValorReal(): Retorna el valor real representat com un String de la cel·la a la que fa referència
  - GetTipus(): Retorna un enumerador amb el tipus de la cel·la a la que fa referencia
  - EsFuncio(): Retorna un boolea, indicant si la cel·la referenciada conté una funció o no
  - GetFuncio(): Retorna l'instancia de la Funcio que te associada la cel·la referenciada, si no en te cap, retorna NULL
  - accept(IVisitor visitor): Implementació de la interfície IVisitable, paràmetre visitor que vol afegir a Referencia
  - toString(): Retorna la representació en String de la referencia

2.1.12. BlocReferencies

La classe BlocReferencies representa un conjunt de referències a un full

- Atributs:
  - min: Una parella d'enters que representa la primera cantonada d'un area quadrada de cel·les, els índexs seran els mínims
  - max: Una parella d'enters que representa la primera cantonada d'un area quadrada de cel·les, els índexs seran els màxims
  - full: Full al que pertany el bloc
  - bloc: Llista de referències (Referencia)
- Mètodes:
  - BlocReferencies(Pair<int,int> pos1,Pair<int,int> pos2, Full f): Constructora, crea un bloc de referències a partir de dos punts, calcula la cantonada mínima, la màxima i obté les referencies del bloc.
  - GetSize(): Retorna el nombre de referencies del bloc
  - GetReferencies(): Retorna la llista de referències que formen el bloc
  - accept(IVisitor visitor): Implementació de la interfície IVisitable, parametre visitor que vol afegir al BlocReferencies
  - toString(): Retorna la representació en String del BlocReferencies

2.1.13. Token

La classe abstracta Token representa l'abstracció dels diferents tipus primitius que poden passar-se com a paràmetres a les funcions

- Mètodes:

- asNumber(): Retorna la representació del token com a Number, si no pot, genera una excepció
- asDate(): Retorna la representació del token com a LocalDate, si no pot, genera una excepció
- asString(): Retorna la representació del token com a String
- accept(IVisitor visitor): Implementació de la interfície IVisitable, paràmetre visitor que vol afegir al Token

2.1.14. ErrorT

La classe ErrorT implementa Token, representa un error a la hora de calcular el resultat d'una Funcio

- Atributs:
  - token: String que conté el missatge d'error que es vol transmetre
- Mètodes:
  - ErrorT(String s): Constructora amb l'error que es vol guardar
  - asNumber(): Genera una excepció
  - asDate(): Genera una excepció
  - asString(): Retorna el token
  - accept(IVisitor visitor): Implementació de la interfície IVisitable, paràmetre visitor que vol afegir al Token
  - toString(): Retorna el token

2.1.15. StringT

La classe StringT implementa Token, representa un String

- Atributs:
  - token: String
- Mètodes:
  - StringT(String s): Constructora amb el valor del String que es vol guardar
  - asNumber(): Intenta parsejar el String com a Number, si no pot, genera una excepció
  - asDate(): Intenta parsejar el String com a LocalDate, si no pot, genera una excepció
  - asString(): Retorna el token
  - accept(IVisitor visitor): Implementació de la interfície IVisitable, paràmetre visitor que vol afegir al Token
  - toString(): Retorna el token

2.1.16. DataT

La classe DataT implementa Token, representa un LocalDate

- Atributs:
  - token: LocalDate
- Mètodes:
  - DateT(): Constructora buida, construeix un token amb la data actual
  - DateT(LocalDate s): Constructora amb la data que es vol guardar

- DateT(String s): Constructora amb String, parseja el String en format de data (dd/MM/yyyy)
- asNumber(): Genera una excepció
- asDate(): Retorna la data guardada al crear-se
- asString(): Retorna la data en format String (dd/MM/yyyy)
- accept(IVisitor visitor): Implementació de la interfície IVisitable, paràmetre visitor que vol afegir al Token
- toString(): Retorna la data en format String (dd/MM/yyyy)

2.1.17. NumberT

La classe NumberT implementa Token, representa un Number

- Atributs:
  - token: Number
- Mètodes:
  - NumberT(Number n): Constructora amb un Number, inicialitza el token amb el numero especificat
  - NumberT(int i): Constructora amb un enter, inicialitza el token com a Integer amb el valor de i
  - NumberT(double d): Constructora amb un double, inicialitza el token com a Double amb el valor de d
  - asNumber(): Retorna el objecte Number que te guardat
  - asDate(): Genera una excepció
  - asString(): Retorna la representació en String del numero
  - accept(IVisitor visitor): Implementació de la interfície IVisitable, paràmetre visitor que vol afegir al Token
  - toString(): Retorna la representació en String del numero

2.1.18. IVisitor

Interfície del patró Visitor: Les classes que implementen la interficie decideixen com afegir/processar els diferents IVisitables

Els mètodes permeten processar els IVisitable de manera diferent segons el seu tipus.

- Metodes
  - visit(Funcio f): Implementació per al IVistable de tipus Funcio
  - visit(Referencia r): Implementació per al IVistable de tipus Referencia
  - visit(BlocReferencies b): Implementació per al IVistable de tipus BlocReferencies
  - visit(Token t): Implementació per al IVistable de tipus Token

2.1.19. IVisitable

La interfície IVisitable determina si una classe és visitable per un IVisitor

- Metodes
  - accept(IVisitor visitor): L'implementació de cada classe sempre serà la mateixa: { visitor.visit(this); }

2.1.20. FuncioError

La classe FuncioError exten de la classe abstracta Funcio, representa una funció mal declarada que ha generat una excepció a la hora de crear-se.

- Atributs:
  - token: ErrorT
- Mètodes:
  - FuncioError(String s): Constructora amb un String amb l'error que ha causat la creació de la funcio de tipus error
  - Token executa(): Implementació del metode abstracte executa(), retorna un ErrorT amb el missatge de la constructora
  - visit(Token t): Implementació de visit Token, només accepta un sol token de tipus ErrorT que es crea a la constructora
  - visit(BlocReferencies b): Implementació buida de visit BlocReferencies

2.1.21. FString

La classe FString es una classe abstracta que exten de la classe Funcio, representa les funcions que només admeten un sol paràmetre de tipus String

- Atributs:
  - s: String
- Mètodes:
  - FString(IVisitable param): Constructora amb un sol paràmetre
  - visit(Token t): Implementació de visit Token, només accepta un sol paràmetre i inicialitza el String s amb el valor del token com a string
  - visit(BlocReferencies b): Implementació de visit BlocReferencies, com que només s'accepta un parametre, aquest no pot ser un bloc

2.1.22. Reemplaca

La classe Reemplaça (Reemplaca.java) exten Funcio, representa una funció de reemplaçament, accepta 3 parametres, el String d'entrada, el valor que es vol reemplaçar i el valor per el qual es vol reemplaçar

- Atributs:
  - entrada: String
  - vell: String
  - nou: String
  - i: int, comptador de parametres
  - Nom: String, nom de la funció, REPLACE
- Mètodes:
  - Reemplaca(List<IVisitable> params): Constructora amb una llista de parametres
  - Token executa(): Retorna el String d'entrada amb els valors reemplaçats
  - ini(): Inicialitza el valor de i a 0

- visit(Token t): Implementació de visit Token, segons el valor de i, inicialitza entrada, vell o nou
- visit(BlocReferencies b): Implementació de visit BlocReferencies, cap dels tres paràmetres poden ser un bloc

2.1.23. aMajuscula

La classe aMajuscula exten FString, representa una funció de capitalització, accepta 1 paràmetre

- Atributs:
  - Nom: String, nom de la funció, MAJUS
- Mètodes:
  - aMajuscula(param): Constructora amb un sol paràmetre
  - Token executa(): Retorna el String passat com a paràmetre amb majúscules

2.1.24. numeroParaules

La classe numeroParaules exten FString, representa una funció que compta el nombre de paraules que té un String, accepta 1 paràmetre

- Atributs:
  - Nom: String, nom de la funció, NUMPARAULES
- Mètodes:
  - numeroParaules(param): Constructora amb un sol paràmetre
  - Token executa(): Retorna el nombre de paraules en el text passat com a paràmetre (considerant que l'apòstrof forma part de la paraula)

2.1.25. tamanyText

La classe tamanyText exten FString, representa una funció que compta el nombre de caracters que té un String, accepta 1 paràmetre

- Atributs:
  - Nom: String, nom de la funció, TAMANY
- Mètodes:
  - tamanyText(param): Constructora amb un sol paràmetre
  - Token executa(): Retorna el nombre de paraules en el nombre de caracters en el text s

2.1.26. FData

La classe FData es una classe abstracta que exten de la classe Funcio, representa les funcions que només admeten un sol paràmetre de tipus Data

- Atributs:
  - data: LocalDate
- Mètodes:
  - FData(IVisitable param): Constructora amb un sol paràmetre
  - visit(Token t): Implementació de visit Token, només accepta un sol paràmetre i inicialitza la Data data amb el valor del token com a LocalDate
  - visit(BlocReferencies b): Implementació de visit BlocReferencies, com que només s'accepta un parametre, aquest no pot ser un bloc

2.1.27. obteAny

La classe obteAny es una classe ue exten de la classe FData, retorna el any de la data pasada com a paràmetre

- Atributs:
  - Nom: String, nom de la funció, OBTEANY
- Mètodes:
  - obteAny(IVisitable param): Constructora amb un sol paràmetre
  - Token executa(): Retorna l'any de la data

2.1.28. obteDia

La classe obteDia es una classe ue exten de la classe FData, retorna el dia de la data pasada com a paràmetre

- Atributs:
  - Nom: String, nom de la funció, OBTEDIA
- Mètodes:
  - obteDia(IVisitable param): Constructora amb un sol paràmetre
  - Token executa(): Retorna el dia del mes de la data

2.1.29. obteDiaSetmana

La classe obteDiaSetmana es una classe ue exten de la classe FData, retorna el dia de la setmana de la data pasada com a paràmetre

- Atributs:
  - Nom: String, nom de la funció, DIASETMANA
- Mètodes:
  - obteDia(IVisitable param): Constructora amb un sol paràmetre
  - Token executa(): Retorna el dia de la setmana de la data

2.1.30. obteMes

La classe obteMes es una classe ue exten de la classe FData, retorna el número de mes de la data pasada com a paràmetre

- Atributs:
  - Nom: String, nom de la funció, OBTEMES
- Mètodes:
  - obteMes(IVisitable param): Constructora amb un sol paràmetre
  - Token executa(): Retorna el número de mes de la data

2.1.31. FNumerica

La classe FNumerica es una classe abstracta que exten de la classe Funcio, representa les funcions que només admeten paràmetres de numerics

- Atributs:
  – values: List<Number> parametres com a numeros
- Mètodes:
  - FNumerica(List<IVisitable> params): Constructora amb llista de paràmetres
  - ini(): Inicialització previa d'una funció
  - visit(Token t): Implementació de visit Token, comprova que el token es de tipus numero, en cas contrari, l'ignora.

- visit(BlocReferencies b): Implementació de visit BlocReferencies, implementació per a funcions N-àries

2.1.32. Suma

La classe Suma es una classe que exten de la classe FNumerica, retorna la suma dels paràmetres

- Atributs:
  - Nom: String, nom de la funció, SUMA
- Mètodes:
  - Suma(List<IVisitable> param): Constructora amb una llista paràmetres
  - Token executa(): Retorna el sumatoria dels paràmetres

2.1.33. Multiplicacio

La classe Multiplicacio es una classe que exten de la classe FNumerica, retorna la multiplicacio dels paràmetres

- Atributs:
  - Nom: String, nom de la funció, MULT
- Mètodes:
  - Multiplicacio(List<IVisitable> param): Constructora amb una llista de paràmetres
  - Token executa(): Retorna el sumatoria dels paràmetres

2.1.34. Mitjana

La classe Mitjana es una classe que exten de la classe FNumerica, retorna la mitjana dels paràmetres

- Atributs:
  - Nom: String, nom de la funció, MITJANA
- Mètodes:
  - Mitjana(List<IVisitable> param): Constructora amb una llista de paràmetres
  - Token executa(): Retorna la mitjana dels paràmetres

2.1.35. Mediana

La classe Mediana es una classe que exten de la classe FNumerica, retorna la mediana dels paràmetres

- Atributs:
  - Nom: String, nom de la funció, MEDIANA
- Mètodes:
  - Mediana(List<IVisitable> param): Constructora amb una llista de paràmetres
  - Token executa(): Retorna la mediana dels paràmetres

2.1.36. Variancia

La classe Variancia es una classe que exten de la classe FNumerica, retorna la variància dels paràmetres

- Atributs:
  - Nom: String, nom de la funció, VARIANCIA
- Mètodes:

- Variancia(List<IVisitable> param): Constructora amb una llista de paràmetres
- Token executa(): Retorna la variància dels paràmetres

2.1.37. DesviacioEstandard

La classe DesviacioEstandard es una classe que exten de la classe FNumerica, retorna la desviació estàndard dels paràmetres

- Atributs:
  - Nom: String, nom de la funció, DESVEST
- Mètodes:
  - DesviacioEstandard(List<IVisitable> param): Constructora amb una llista de paràmetres
  - Token executa(): Retorna la desviació estàndard de la llista de paràmetres

2.1.38. FN_Unaria

La classe FN_Unaria es una classe abstracta que exten de la classe FNumerica, representa les funcions que només admeten un sol paràmetre numeric

- Atributs:
  – n: Number, paràmetre com a número
- Mètodes:
  - FN_Unaria(IVisitable param): Constructora amb un sol paràmetre
  - ini(): Inicialització previa d'una funció
  - visit(Token t): Implementació de visit Token, comprova que el token es un número
  - visit(BlocReferencies b): Implementació de visit BlocReferencies, com que només s'accepta un paràmetre, aquest no pot ser un bloc

2.1.39. Absolut

La classe Absolut es una classe que exten de la classe FN_Unaria, retorna el valor absolut del paràmetre

- Atributs:
  - Nom: String, nom de la funció, ABS
- Mètodes:
  - Absolut(IVisitable param): Constructora amb un sol paràmetre
  - Token executa(): Retorna el valor absolut del paràmetre

2.1.40. Arrodonir

La classe Arrodonir es una classe que exten de la classe FN_Unaria, retorna el valor arrodonit del paràmetre

- Atributs:
  - Nom: String, nom de la funció, ARRODONIR
- Mètodes:
  - Arrodonir(IVisitable param): Constructora amb un sol paràmetre
  - Token executa(): Retorna el valor arrodonit del paràmetre

2.1.41. Truncar

La classe Truncar es una classe que exten de la classe FN_Unaria, retorna el valor enter truncat del paràmetre

- Atributs:
  - Nom: String, nom de la funció, TRUNCAR
- Mètodes:
  - Truncar(IVisitable param): Constructora amb un sol paràmetre
  - Token executa(): Retorna el valor enter truncat del paràmetre

2.1.42. Factorial

La classe Factorial es una classe que exten de la classe FN_Unaria, retorna el Factorial del paràmetre interpretat com a enter

- Atributs:
  - Nom: String, nom de la funció, FACTORIAL
- Mètodes:
  - Factorial(IVisitable param): Constructora amb un sol paràmetre
  - Token executa(): Retorna el factorial del paràmetre interpretat com a enter

2.1.43. decimalABinari

La classe decimalABinari es una classe que exten de la classe FN_Unaria, retorna la interpretació en binari d'un enter

- Atributs:
  - Nom: String, nom de la funció, DEC2BIN
- Mètodes:
  - decimalABinari(IVisitable param): Constructora amb un sol paràmetre
  - Token executa(): Retorna la interpretació en binari del paràmetre interpretat com a enter

2.1.44. decimalAHexadecimal

La classe decimalAHexadecimal es una classe que exten de la classe FN_Unaria, retorna la interpretació en hexadcimal d'un enter

- Atributs:
  - Nom: String, nom de la funció, DEC2HEX
- Mètodes:
  - decimalAHexadecimal(IVisitable param): Constructora amb un sol paràmetre
  - Token executa(): Retorna la interpretació en hexadecimal del paràmetre interpretat com a enter

2.1.45. hexadecimalADecimal

La classe hexadecimalADecimal es una classe que exten de la classe FN_Unaria, retorna la el valor d'un numero hexadcimal com a enter

- Atributs:
  - Nom: String, nom de la funció, HEX2DEC
- Mètodes:

- hexadecimalADecimal(IVisitable param): Constructora amb un sol paràmetre

– Token executa(): Retorna el valor d'un parametre hexadecimal com a enter

– visit(Token t): Override de visit Token, comprova que el token es un número Hexadecimal

2.1.46. binariADecimal

La classe binariADecimal es una classe que exten de la classe FN_Unaria, retorna la el valor d'un numero binari com a enter

- Atributs:
  – Nom: String, nom de la funció, BIN2DEC
- Mètodes:
  – binariADecimal(IVisitable param): Constructora amb un sol paràmetre
  – Token executa(): Retorna el valor d'un parametre binari com a enter
  – visit(Token t): Override de visit Token, comprova que el token es un número Binari

2.1.47. FN_Binaria

La classe FN_Binaria es una classe abstracta que exten de la classe FNumerica, representa les funcions que només admeten dos paràmetres numerics

- Atributs:
  – a: Number, primer paràmetre com a número
  – b: Number, segon paràmetre com a número
  – isSecond: boolea que determina si el paràmetre es el segon
- Mètodes:
  – FN_Binaria(List<IVisitable> params): Constructora amb una llista de parametres, només s'espera dos
  – ini(): Inicialització previa d'una funció
  – visit(Token t): Implementació de visit Token, comprova que el token es un número, inicialitza els paràmetres adequadament
  – visit(BlocReferencies b): Override de visit BlocReferencies, com que només s'accepten dos paràmetres, aquests no poden ser blocs

2.1.48. Arrel

La classe Arrel es una classe que exten de la classe FN_Binaria, retorna l'arrel del primer paràmetre del grau del segon paràmetre

- Atributs:
  - Nom: String, nom de la funció, SQRT
- Mètodes:
  – Arrel(List<IVisitable> params): Constructora amb una llista de paràmetres, només n'espera dos
  – Token executa(): Retorna l'arrel del primer paràmetre del grau del segon paràmetre o una excepció si s'intenta fer una arrel negativa

2.1.49. Divisio

La classe Divisio es una classe que exten de la classe FN_Binaria, retorna la divisió del primer paràmetre dividit per el segon paràmetre

- Atributs:
  - Nom: String, nom de la funció, DIVISIO
- Mètodes:
  – Divisio(List<IVisitable> params): Constructora amb una llista de paràmetres, només n'espera dos
  – Token executa(): Retorna la divisió del primer paràmetre entre el segon, llença una excepció si el segon paràmetre es 0

2.1.50. Potencia

La classe Potencia es una classe que exten de la classe FN_Binaria, retorna la el primer paràmetre elevat al segon paràmetre

- Atributs:
  - Nom: String, nom de la funció, POW
- Mètodes:
  – Potencia(List<IVisitable> params): Constructora amb una llista de paràmetres, només n'espera dos
  – Token executa(): Retorna el valor de a^b

2.1.51. Resta

La classe Resta es una classe que exten de la classe FN_Binaria, retorna la resta del primer paràmetre amb el segon

- Atributs:
  - Nom: String, nom de la funció, RESTA
- Mètodes:
  – Resta(List<IVisitable> params): Constructora amb una llista de paràmetres, només n'espera dos
  – Token executa(): Retorna el valor de (a - b)

2.1.52. FN_Binaria_Blocs

La classe FN_Binaria_Blocs es una classe abstracta que exten de la classe FN_Binaria, representa les funcions que només admeten dos blocs de paràmetres de tipus numèric

- Atributs:
  – a: List<Number>, primer bloc de paràmetres com a números
  – b: List<Number>, segon bloc de paràmetres com a números
- Mètodes:
  – FN_Binaria_Blocs(List<IVisitable> params): Constructora amb una llista de paràmetres, només s'espera dos paràmetres de tipus bloc
  – ini(): Inicialització previa d'una funció
  – visit(Token t): Implementació de visit Token, comprova que el token es un número i l'afegeix a la llista corresponent

– visit(BlocReferencies b): Override de visit BlocReferencies, com que només s'accepten paràmetres de tipus bloc, aqui determinem a quina llista de valors estem omplint

2.1.53. CorrelacioPearson

La classe CorrelacioPearson es una classe que exten de la classe FN_Binaria_Blocs, retorna la correlacio de pearson entre dos blocs

- Atributs:
  - Nom: String, nom de la funció, PEARSON
- Mètodes:
  – CorrelacioPearson(List<IVisitable> params): Constructora amb una llista de paràmetres, només s'espera dos paràmetres de tipus bloc
  – Token executa(): Retorna el valor de calcular la correlacio de pearson entre els elements del dos blocs

2.1.54. Covariancia

La classe Covariancia es una classe que exten de la classe FN_Binaria_Blocs, retorna la covariancia entre dos blocs

- Atributs:
  - Nom: String, nom de la funció, COVARIANCIA
- Mètodes:
  – Covariancia(List<IVisitable> params): Constructora amb una llista de paràmetres, només s'espera dos paràmetres de tipus bloc
  – Token executa(): Retorna el valor de calcular la covariancia entre els elements del dos blocs

2.2. Explicació dels controladors de domini

2.2.1. CtrlDomini

El controlador de Domini s'encarrega d'instanciar la resta de controladors de la capa de domini i proporciona una interfície més pròxima a l'usuari dels mètodes de la resta de controladors com el CtrlDocument o el CtrlClipboard.
També s'encarrega de comunicar–se amb l'intèrpret per realitzar els càlculs de funcions quan es modifica una cel·la.

- Atributs:
  – _ctrlDocument: Conté la instancia del CtrlDocument que proporciona accés a Document i les seves relacions
  – _ctrlClipboard: Conté la instancia del CtrlClipboard que manté el *Clipboard* i proporciona les funcionalitats copiar/tallar/enganxar
  – _ctrlPersistencia: Conté la instancia del CtrlPersistencia que permet accedir i guardar les dades persistents de les altres capes.
  – _config: Conté la instancia de la Configuracio que proporciona accés a documents recents creats per l'usuari

- Mètodes:

– CtrlDomini(): Creadora per defecte que instancia les diferents relacions del CtrlDomini amb la resta de controladors/classes i carrega les dades de configuració persistens, si existeixen.
– cercarValorPos(String valor): Retorna un vector de pairs amb les posicions de totes les ocurrències de Valor en el Full actiu del document obert.
– reemplacarValor(int idC1, int idF1, int idC2, int idF2,
String cerca, String reemplac): Reemplaça cada ocurrencia de cerca (En el bloc format per les cel·les idC1, idF1 i idC2, idF2 del Full actiu actual) i la substitueix per reemplac
– copiarCela(int idC, int idF): Copia el valor (En aquell instant) de la cel·la identificada per idC i idF en el Full actiu actual
– tallarCela(int idC, int idF): Copia el valor (En aquell instant) de la cel·la identificada per idC i idF en el Full actiu actual i esborra el seu contingut
– enganxar(int idC, int idF): Enganxa el/els valor/valors actual/actuals del *Clipboard* a partir del punt indicat per la cel·la idC, idF en el Full actiu actual (Pot ser un bloc si s'havia copiat anteriorment)
– copiarBloc(int idC1, int idF1, int idC2, int idF2): Copia els valors (En aquell instant) de les Cel·les que es troben en el rang idC1, idF1 i idC2, idF2 (Inclusiu) en el Full actiu actual.
– tallarBloc(int idC1, int idF1, int idC2, int idF2): Copia els valors (En aquell instant) de les Cel·les que es troben en el rang idC1, idF1 i idC2, idF2 (Inclusiu) en el Full actiu actual. I després elimina el contingut de les Cel·les.
– obtenirValorCela(int idC, int idF): Retorna el valor (String) entrat per l'usuari (Per exemple "=SUMA(A1:B5)") de la Cel·la identificada per idC, idF en el Full actiu actual.
– obtenirValorCela(int idC, int idF, int idFull): Retorna el valor (String) entrat per l'usuari (Per exemple "=SUMA(A1:B5)") de la Cel·la identificada per idC, idF en el Full idFull i canvia el full actiu a idFull.
– obtenirValorRealCela(int idC, int idF): Retorna el valor (String) calculat per el sistema (Per exemple "65.7") de la Cel·la identificada per idC, idF en el Full actiu actual.
– obtenirValorRealCela(int idC, int idF, int idFull): Retorna el valor (String) callculat per el sistema (Per exemple "65.7") de la Cel·la identificada per idC, idF en el Full idFull i canvia el full actiu a idFull.
– ordenarStrings(int idC1, int idF1, int idC2, int idF2): Ordena, per columnes, de manera descendent el valor String de les Cel·les que es troben en el bloc format per idC1, idF1 i idC2, idF2 en el Full actiu actual.
– ordenarIntegers(int idC1, int idF1, int idC2, int idF2): Ordena, per columnes, de manera descendent el valor Integer de les Cel·les que es troben en el bloc format per idC1, idF1 i idC2, idF2 en el Full actiu actual.

– crearFull(): Crea un nou full de 1000*1000 amb nom “Full.(id+1)” s’afegeix al document obert actualment i es posa com a full actiu.
– crearFull(String nomFull): Crea un nou full de 1000*1000 amb nom = nomFull s’afegeix al document obert actualment i es posa com a full actiu.
– crearFull(int numCols, int numFiles): Crea un nou full de numCols*numFiles amb nom “Full.(id+1)” s’afegeix al document obert actualment i es posa com a full actiu.
– crearFull(String nomFull, int numCols, int numFiles): Crea un nou full de numCols*numFiles amb nom = numFiles, s’afegeix al document obert actualment i es posa com a full actiu.
– setFullActiu(int idf): Canvia el Full actiu al Full amd id = idf
– getNomDocument(): Retorna el nom (String) del document obert actualment
– getPathDocument(): Retorna la localització (String) del document obert actualment
– getIDsFulls(): Retorna una llista d’enters amb el les IDs dels fulls que conté el document obert actualment.
– getNumFiles(): Retorna el número de files (int) que te el Full actual (Aquestes files no tenen perquè tenir contingut)
– getNumColumnes(): Retorna el número de columnes (int) que te el Full actual (Aquestes columnes no tenen perquè tenir contingut)
– getNumFulls(): Retorna el número de fulls (int) que conté el document obert.
– crearDocument(String path): Crea un document (Sense fulls) amb localització = path, nom = “Sense_Títol”, si no n’hi ha cap d’obert aquest es posa com a document obert. La informació relevant a aquest document es guarda a Configuració i s’actualitza el fitxer de persistencia de configuració.
– crearDocument(String path, String nomDoc): Crea un document (Sense fulls) amb localització = path, nom = nomDoc, si no n’hi ha cap d’obert aquest es posa com a document obert. La informació relevant a aquest document es guarda a Configuració i s’actualitza el fitxer de persistencia de configuració.
– tancarDocument(): Es tanca el document obert actualment.
– afegirFila(int pos): Afegeix una fila justament a la posició indicada per pos en el Full actiu.
– afegirColumna(int pos): Afegeix una columna justament a la posició indicada per pos en el Full actiu.
– eliminarFila(int pos): S’elimina la fila identificada per pos del full actiu
– eliminarColumna(int pos): S’elimina la columna identificada per pos del Full actiu
– afegirFilaFull(int pos, int idFull): Afegeix una fila justament a la posició indicada per pos en el Full idFull. Es canvia el full actiu a idFull.

– afegirColumnaFull(int pos, int idFull): Afegeix una columna justament a la posició indicada per pos en el Full idFull. Es canvia el full actiu a idFull.
– eliminarFilaFull(int pos, int idFull): S'elimina la fila identificada per pos del Full idFull. Es canvia el full actiu a idFull.
– eliminarColumnaFull(int pos, int idFull): S'elimina la columna identificada per pos del Full idFull. Es canvia el full actiu a idFull.
– eliminarFullActiu(): S'elimina el full actiu. Si és l'únic full del document obert se'n crea un altre de 1000*1000 amb nom = "Full.(id+1)"
– canviarNomFullActiu(String nom): Es canvia el nom del Full actiu per nom
– canviarNomDocument(String nom): Es canvia el nom del document obert actualment per nom
– modificarCela(int idC, int idF, String valor): Es modifica el valor usuari de la Cel·la idC, idF del Full actiu per valor i es fan els càlculs del valor "real" del input del usuari. També s'actualitza la data d'última modificació del document obert a Configuració i es guarda de manera persistent.
– modificarCelaFull(int idC, int idF,int idFull, String valor): Es modifica el valor usuari de la Cel·la idC, idF del Full idFull per valor i es fan els càlculs del valor "real" del input del usuari. També s'actualitza la data d'última modificació del document obert a Configuració i es guarda de manera persistent. Es canvia el full actiu a idFull.
– eliminarDocument(String nomDoc): Elimina el document amb nom = nomDoc de Configuració i es guarda de manera persistent.
– guardaConfig(): Serialitza la instancia de Configuració _config i envia les dades a CtrlPersistencia per guardar les dades a disc.
– obtenirConfig(): Converteix i retorna les dades de Configuració en una matriu d'String per poder ser passat entre les diferents capes.
– guardaCSV(int idFull): Es canvia full actiu a idFull i es guarda el seu contingut "real" (Valor calculat per el sistema) en un fitxer de format CSV.
– carregaCSV(String path): Es carrega el fitxer en format CSV indicat per path. Es crea un document amb path = path i nom = "Sense_Títol" i se l'hi afegeix un full del tamany del CSV carregat. Es canvia el document obert al nou document i el full actiu al nou full. (No pot haver–hi cap instancia oberta de cap document)

2.2.2. CtrlDocument

El controlador de document s'encarregarà de la gestió dels mètodes de les classes de Document i Full. Es comunica amb Document per tal de realitzar totes les funcionalitats referent a aquest.

- Atributs:

- Full FullActiu: Identificador intern del Full que està Actiu.
-Document DocumentObert: Identificador intern del Document Obert

- Mètodes:
-CtrlDocument(): Constructora del controlador de Document.
-getDocumentObert(): Retorna l'atribut DocumentObert.
-obtenirFullActiu(): Retorna l'atribut FullActiu.
-setFullActiu(int idf): A partir de un índex d'un Full, passa aquest a FullActiu.
-crearDocument(String path): Crida a la creadora de Document i li passa els paràmetres rebuts. Deixa com a DocumentObert el Document creat.
-crearDocument(String path, String nomDoc): Crida a la creadora de Document i li passa els paràmetres rebuts. Deixa com a DocumentObert el Document creat.
-crearFull(): Crida a la funció de Document crearFull sense paràmetres. Deixa com a FullActiu el Full creat.
-CrearFull(String nomF): Crida a la funció de Document crearFull amb el paràmetre rebut. Deixa com a FullActiu el Full creat.
-crearFull(int nC, int nF): Crida a la funció de Document crearFull amb el paràmetre rebut. Deixa com a FullActiu el Full creat.
-crearFull(String nomF, int nC, int nF): Crida a la funció de Document crearFull amb el paràmetre rebut.
-canviarNomDoc(String nom): Crida a la funció de Document canviarNomDoc amb el paràmetre rebut.
-canviarNomFull(String nom): Crida a la funció de Document canviarNomFull amb el paràmetre rebut.
-tancarDocument(): Es tanca el DocumentObert i es posen a Null l'atribut DocumentObert i FullActiu
-EliminarFullActiu(): S'elimina el FullActiu, sí no hi ha més fulls se'n crea un de nou automàticament i passa a ser el FullActiu. En cas que hi hagi Fulls amb identificadors més petits que l'eliminat passaria el primer en ordre descendent a ser el FullActiu. En el cas que no hi hagi fulls amb identificadors més petits el fullActiu passaria a ser el següent amb identificador major.
-obtenirCela(int idC, int idF): Donades una columna i una fila, crida a la funció GetCela de Full i retorna la cel·la corresponent al FullActiu o null si no n'hi ha cap.
-modificarCela(int i, int j, String s): Crida a la funció ModificaCela de Full.
-CercarValor(String valor): Crida a la Funció CercarValor de Full
-AfegirFila(int pos): Crida a la funció AfegirFila de Full.
-AfegirColumna(int pos):Crida a la funció AfegirCol de Full.
-EliminarFila(int pos):Crida a la funció EliminarFila de Full.
-EliminarColumna(int pos):Crida a la funció EliminarColumna de Full.

2.2.3. CtrlClipboard

El controlador de clipboard s'encarrega de gestionar els mètodes copiar, tallar i enganxar cel·les i blocs al full corresponent. Les cel·les copiades queden dins una matriu d'ArrayList de Cela que fa la funció de porta–retalls (*clipboard*).

- Atributs:
  – ArrayList<ArrayList<Cela>> Clipboard: Conté totes les cel·les del bloc que s'ha copiat o tallat.
  – CtrlDocument _ctrlDocument: Instància de controlador de document per a poder cridar al mètode per a modificar cel·la.
- Mètodes:
  – CtrlClipboard(CtrlDocument cd): Constructora del controlador de Clipboard que fa de setter del controlador de document que té com a atribut privat
  – copiarBloc(int idC1, int idF1, int idC2, int idF2): Copiar al clipboard totes les cel·les del bloc.
  – tallarBloc(int idC1, int idF1, int idC2, int idF2): Copiar al clipboard totes les cel·les del bloc i esborra el bloc
  – enganxar(int idC, int idF): Enganxa al full les cel·les que té guardades al clipboard
  – ordenarStrings(int idC1, int idF1, int idC2, int idF2): Ordena els valors String que hi ha dins del bloc de cel·les idC1, idF1 i idC2, idF2
  – ordenarIntegers(int idC1, int idF1, int idC2, int idF2): Ordena els valors Integer que hi ha dins del bloc de cel·les idC1, idF1 i idC2, idF2

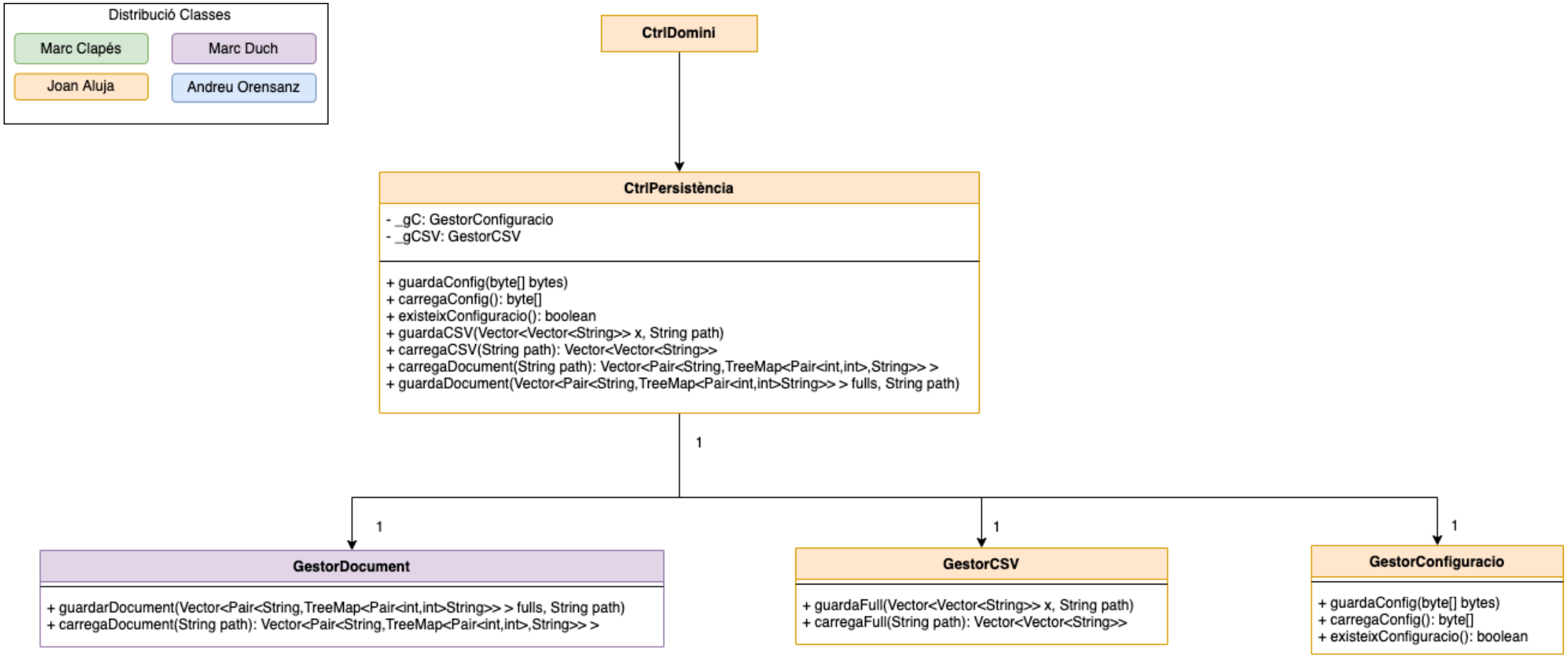
2.2.4. CtrlFuncions

El controlador de Funcions s'encarrega de parsejar un String que representa una funció i els seus paràmetres

- Atributs:
  – Full full: Full al que pertany el controlador
- Mètodes:
  – CtrlFuncions(): Constructora per defecte, només utilitzada per el driver
  – CtrlFuncions(Full f): Constructora amb el full al que pertany
  – Funcio GetFuncio(String s): Retorna la funció representatda per l'String passat com a paràmetre

3. Diagrama de les classes i controladors de la capa de persistència

A continuació mostrem el disseny del diagrama de les classes i controladors de la capa de persistència. El diagrama es pot trobar detallat a la carpeta DOCS.



3.1. Explicació de les classes de persistència

3.1.1. GestorConfiguracio

El GestorConfiguracio s'encarrega de guardar/carregar les dades en format serialitzat a disc

- Atributs:
  – path: Conté el path complet on es guarda la configuració del programa
- Mètodes:
  – guardaConfig(byte[] bytes): Guarda les dades del byte array bytes a disc en el path marcat per l'atribut path.
  – carregaConfig(): Retorna un byte array amb les dades llegides del fitxer marcat per l'atribut path
  – existeixConfiguracio(): Retorna true si existeix el fitxer marcat per path, false altrament.
  – borraConfig(): Elimina el fitxer marcat per path, si s'ha pogut eliminar retorna true, altrament retorna false.

3.1.2. GestorCSV

El GestorCSV s'encarrega de guardar/carregar les dades en format CSV (El format CSV estandard només pot guardar un full i per norma guarda les dades ja calculades, per tant provoca pèrdua d'informació)

- Mètodes:
  – guardaFull(Vector<Vector<String>> x, String path): Guarda les dades rebudes en la matriu x en el fitxer marcat per path en format CSV (Separat per comes)
  – carregaFull(String path): Carrega el fitxer en format CSV marcat per path i retorna les dades en una matriu d'Strings.

3.1.3. GestorDocument

El GestorDocument s'encarrega de guardar/carregar les dades de les classes de domini en un format semblant al XML, les etiquetes indiquen els diferents atributs que es necessiten per instanciar les classes corresponents (ex valor real de cel.la o nom del document/full)

- Mètodes:
  – guardaDocument(Vector<Pair<String,TreeMap<Pair<int,int>,<String>>> celes, String path): Guarda les dades rebudes en el fitxer marcat per path en un format inventat (.prop). El vector representen els fulls, on el string de la parella és el nom del full, el TreeMap te per clau la posició de la cel·la i per valor, el valor de l'usuari
  – carregaFull(String path): Carrega el fitxer en el nostre format inventat (.prop) i retorna un vector de parelles on el primer elemnt es el nom del full, el segon es un diccionari amb les cel·les del full i el seu valor d'usuari

3.2. Explicació del controlador de persistència

3.2.1. CtrlPersistència

El controlador de Persistència s'encarrega de gestionar la entrada/sortida a disc de dades en diferents formats. Principalment el format .prop que manté totes les dades d'un document, el format CSV que manté nomes les dades d'un full (Sense fórmules, només el valor calculat) i la configuració, que manté les dades dels documents que s'han creat i la seva localització.

- Atributs:
  - _gC: Conté la instancia del GestorConfiguracio de capa de persistència
  - _gCSV: Conté la instancia del GestorCSV de capa de persistència
- Mètodes:
  - guardaConfig(byte[] bytes): Guarda les dades del byte array bytes a disc en el path marcat per l'atribut path de la classe GestorConfiguracio.
  - carregaConfig(): Retorna un byte array amb les dades llegides del fitxer marcat per l'atribut path de la classe GestorConfiguracio.
  - existeixConfiguracio(): Retorna true si existeix el fitxer marcat per path (Atribut de la classe GestorConfiguracio), false altrament.
  - borraConfig(): Elimina el fitxer marcat per path (Atribut de la classe GestorConfiguracio), si s'ha pogut eliminar retorna true, altrament retorna false.
  - guardaCSV(Vector<Vector<String>> x, String path): Guarda les dades rebudes en la matriu x en el fitxer marcat per path en format CSV (Separat per comes)
  - carregaCSV(String path): Carrega el fitxer en format CSV marcat per path i retorna les dades en una matriu d'Strings.
  - carregaDocument(String path): Vector<Pair<String,TreeMap<Pair<int,int>,String>> >: Crida al mètode carregaDocument de GestorDocument
  - guardaDocument(Vector<Pair<String,TreeMap<Pair<int,int>String>> > fulls, String path): Crida al mètode guardaDocument de GestorDocument.

4. Descripció dels algorismes i les estructures de dades

4.1. Estructures de dades

A continuació expliquem cada una de les estructures de dades que hem considerat per tal de crear el nostre full de càlcul i quina d'elles hem acabat fent servir.

4.1.2 Full

Tal i com hem especificat anteriorment, la classe full representa una agrupació de cel·les que, tal i com apareix al UML, pot ser buit, que simplement vol dir que només guardarem aquelles cel·les que tinguin un valor no–nul.

Abans d'entrar en detall sobre les estructures plantejades, definirem 2 variables, el número de cel·les que no estan buides seran representades per la x, i la segona variable n, representa el nombre de files, i per tal de simplificar alguns dels costos, el nombre de columnes.

D'aquesta manera, podem dir que el màxim nombre de cel·les (x) serà el nombre de files per el nombre de columnes. Amb això podem definir els límits de $x \mid 0 \leq x \leq n^2$.

A l'hora de decidir com implementar l'agrupació de cel·les d'un full, vam haver de decidir–nos entre varies opcions:

- **Matriu bidimensional:**

  Per a representar una taula d'elements com un full de càlcul, hi ha una estructura que és molt fàcil d'entendre i d'implementar, la matriu. Implementada mitjançant dos col·leccions d'objectes aniuats, tals que la primera col·lecció ("exterior") tindrà la segona col·lecció, i és aquesta la que guardarà els elements finals de la matriu.

  Depenent de com vulguem implementar les funcionalitats de Full, podem interpretar la col·lecció externa com les columnes o com les files sense cap problema.

  - **Matriu d'Arrays**

    L'estructura més evident per implementar una matriu és un Array d'Arrays, la base de les col·leccions d'elements a Java.

    La ventatja principal d'aquesta implementació es que a l'hora d'accedir a un element segons la seva fila i columna, el temps d'accés és instantani, $\Theta(1)$ independentment del nombre d'elements o el tamany.

    En quant als contres, en tenim uns quants:

    Començant per l'espai ocupat en memòria; independentment del nombre de cel·les (x), el espai ocupat en memòria sempre serà $\Theta(n^2)$. Tal i com es guarden els objectes a Java (en aquest cas Cel·la), aquest cost no es tan gran com en altres llenguatges, per tot aixi, el cost segueix sent quadratic i

a la pràctica, el nombre de cel·les que s'omplen en un full molt gran serà molt petit en comparació.
L'altre gran problema apareix a l'hora d'afegir o eliminar files/columnes. Els Arrays tenen un tamany fix, cosa que vol dir que a la hora de fer més gran o més petit un full, hauriem de clonar totes les cel·les a un altre matriu nova, que faria que el cost d'aquestes operacions seria sempre $\Omega(n^2)$, sense plantejar–nos la complexitat de moure tots els elements segons on s'ha afegit la fila/columna.
Una solució que ens vam plantejar per el problema anterior era inicialitzar la matriu amb un tamany superior al especificat per l'usuari/donat per defecte, pero això simplement empitjora el problema de l'espai ocupat.

- **Matriu de LinkedList**

  Per tal de millorar el problema d'afegir/eliminar files o columnes, vam pensar en implementar la matriu amb LinkedList o DoubleLinkedList.

  Amb aquesta implementació, milloràvem el problema d'afegir/eliminar files o columnes, pero només per un dels dos eixos.
  Si la primera LinkedList representa les columnes, afegir–les/eliminar–les era tan fàcil com buscar el node corresponent i afegir una nova LinkedList i actualitzar els punters o eliminar el node (amb la LinkedList anterior), cost temporal lineal $\Theta(n)$. Ara bé, si volem afegir o eliminar una fila (on els índexs de les files estan a la segona LinkedList), el cost augmenta dràsticament, ja que per cada columna, hauriem de trobar la fila corresponent i modificar els nodes segons si s'afegeix o s'elimina la fila. El millor dels casos, eliminem només la primera fila i el cost es $\Omega(n)$, i el pitjor, eliminem l'ultima, $O(n^2)$, la mitja, però, segueix sent $\Theta(n^2)$. Millor que amb Arrays estàtiques, pero no per molt.
  La gran desavantatge que té en comparació a la matriu d'Arrays és el seu cost d'accés. On l'anterior tenia un cost fix $\Theta(1)$, la implementació amb LinkedList trigaría en el pitjor dels casos $O(n + n)$, i de mitja $\Theta(n)$. No es horrible, pero tenint en compte que l'accés a les cel·les és un element fonamental d'un full de càlcul, havíem de buscar una estructura amb un cost d'accés millor que lineal.

- ## Diccionaris

  Les matrius de col·leccions, estàtiques o no, tenen el mateix problema: el seu cost espaial sempre serà $\Theta(n^2)$. Tot i així, l'accés instantani que permet la matriu d'Arrays es practicament impossible de millorar.

Aquí és on entren els diccionaris, amb aquestes estructures de dades ens desfem del concepte de files i columnes, ja que les claus que fan servir no estan restringides a cap tamany.
A més, l'espai que ocupen sempre serà $\Theta(\boldsymbol{x})$ si ens desfem de les cel·les que ja no es facin servir de forma adecuada. Java no permet construir objectes manualment, té una funció anomenada *GarbageCollector* que s'encarrega d'eliminar objectes que no estiguin sent referenciats per tal d'alliberar memoria.

- **HashMap**

  Per aquesta implementació, la idea és tenir un sol HashMap on les claus son strings formatejats tals que: "A1", "C3", etc; i els valors evidentment, com a objectes de tipus Cel·la.

  Com acabem de mencionar, la principal avantatge que te sobre les matrius es que només es guardaran les cel·les que haguem afegit explícitament, ergo l'espai ocupat serà $\Theta(\boldsymbol{x})$.

  Un altre punt positiu es que la cerca de cel·les dins d'un full sempre tindrà cost lineal $\Theta(\boldsymbol{x})$ on les implementacions amb matrius havien de comprovar totes les posicions, tinguin cel·les o no, per tant, tenia un cost quadratic $\Theta(\boldsymbol{n^2})$.

  A la hora d'afegir o eliminar files/columnes, la idea era recorrer tot el HashMap, comprovant si les claus havien estat desplaçades o no, i reinsertant els elements amb les claus actualitzades. Tot això amb un cost lineal en relació a les cel·les: $\Theta(\boldsymbol{x})$.

  La tenir un diccionari amb una sola clau té dos grans inconvenients.
  El primer, i potser més menor, vé per l'implementació del HashMap, la qual fa servir un *Binary Search Tree*, ordena els elements en un arbre binari, cosa que fa que el temps d'accés de mitja sigui $\Theta(\mathbf{1})$, pero en el pitjor dels casos (un cas que realisticament és molt improbable), podria ser lineal amb els objectes del diccionari $\Theta(\boldsymbol{x})$.

  L'altre problema es que a la hora d'ordenar per columnes, trobar els elements a ordenar seria molt difícil, partint de que tenim dues cantonades, és a dir, podem ordenar més d'una columna per crida, si intentem trobar valor associats a les claus incloses en l'àrea seleccionada per les cantonades anteriors, i després, fem un *Insertion Sort* dels elements trobats i els tornem a afegir al seu lloc segons l'ordre. Al final la ordenació te cost quadratic en relació als elements seleccionats per columna que en el pitjor dels casos seria $\Theta(\boldsymbol{n^2})$, pero la abstractesa i

llibertat del HashMap feia més complicat la implementació de certes operacions.

- **SortedMaps aniuats**

  Finalment arribem a la implementació actual, dos SortedMaps aniuats, on el diccionari "exterior" representa les columnes i l'interior les files, d'aquesta manera les operacions sobre columnes, principalment l'ordenació són més fàcils de implementar e interpretar.

  El SortedMap, com el seu nom indica, té l'avantatge de que els seus elements s'ordenen segons el valor de les seves claus, el que fa que sigui més fàcil optimitzar aquelles funcions que requereixin buscar els elements anteriors o posteriors a una certa posició, com es el cas de afegir/eliminar files i columnes.

  L'altre gran avantatge del SortedMap, o més pròpiament dit, el TreeMap, la classe de Java que l'implementa, és que fa servir un *Black–Red Binary Tree*, un tipus d'arbre binari balancejat que fa que les cerques, tant de columnes com de les files siguin $\Theta(\log(\boldsymbol{n})$) en el cas de columnes i $\Theta(\log(\boldsymbol{n}))$ per les files, tenint en compte que el cas promig, tant les files com les columnes seran més petits que el nombre d'elements ($\boldsymbol{x} \leq \boldsymbol{n}$), aleshores, el temps asimptòtic d'accés quedaria tal que $\Theta(\log(\boldsymbol{n})$) < $\Theta(\log(\boldsymbol{x})$)

4.1.3 Interpret

La classe Interpret, com ja hem esmentat anteriorment, només es tracta d'un parser de tipus de dades i també gestiona totes les funcions entrades per l'usuari i s'encarrega d'anar a buscar el resultat, per tant, les estructures de dades emprades han sigut molt senzilles:

- **Vectors**

  Hem utilitzat vectors de String i de Double per guardar la coordenada de les cel·les referenciades i els valors numèrics de la cel·la, respectivament. Hem emprat vector en comptes d'un array, ja que volem crear una estructura buida i anar afegint els valors a mida que es van trobant.

4.1.4 Document

En aquesta classe cal utilitzar una estructura per a guardar els fulls que conté el document. Per tenir un accés més ràpid i eficient hem obtat per un diccionari de manera que podem accedir per clau al full que volem. Més concretament hem emprat un TreeMap<Integer, Full>:

- **TreeMap**
  Tot i que al principi vam considerar emprar un set, vam acabar agafant un TreeMap amb clau Integer, que representa la *id* del full, i un valor Full, que és el full amb aquesta *id* que té per clau.

4.1.5 Configuracio

En aquesta classe necessitem una estructura de dades per poder guardar la informació rellevant de cada document que crea l'usuari. En si, volem guardar la data última modificació, el nom del document i la seva localització (path).

- **TreeMap**
  Vam decidir utilitzar un TreeMap, ja que hem de poder ampliar el nombre de dades de l'estructura fàcilment i poder accedir de manera ràpida i ens proporciona una manera facil de poder tenir les dades ordenades segons la clau.
  La clau en aquest cas està formada per un Pair<LocalDateTime,String>.
  Per poder mantenir les dades ordenades i poder serialitzar l'objecte hem hagut de crear un comparador propi que es troba dins de la classe SerializableConfigComparator al fitxer Configuracio.java

4.1.6 CtrlClipboard

En aquest controlador hem hagut de crear una estructura bidimensional que guardés Cel·les. Per fer això hem emprat una matriu d'arrays de cel·les:

- **Matriu d'arrays de cel·les**
  Hem escollit aquesta estructura, ja que a part de copiar cel·les individualment també copiem blocs, de manera que amb una matriu podem copiar al porta-retalls les cel·les tal i com estan col·locades al full. Hem triat aquesta estructura ja que és molt fàcil anar recorrent les cel·les seleccionades al full i anar-les afegint a la matriu que s'inicialitza buida.
  Hem triat aquesta estructura en comptes d'un array de 2 dimensions, ja que un ArrayList no fa falta definir la mida de l'estructura inicialment i es pot anar afegint valors.

4.1.7 VistaDocumentsRecents

En aquesta vista ens ha calgut una estructura on guardar tots els documents recentment oberts amb certa informació de cada un d'ells. És per això que hem emprat una matriu de strings:

- **Matriu de Strings**

Hem utilitzat una matriu de strings (vector de vectors de strings) per a guardar els documents que hem obtingut de la capa de persistència que són els més recentment oberts. Com a màxim aquesta matriu tindrà 5 posicions (5 documents recents) i per cada una hi haurà només 3 posicions ja que només volem el nom del document, la data quan es va obrir per últim cop i el *path*.

4.1.8 VistaSpreadsheet

Aquesta vista com que és la que mostra els fulls, hem hagut de crear una estructura pròpia combinant JTabbedPane, JScrollPane i JTable, que hem anomentat PanellFulls:

- **PanellFulls**

  Aquesta estructura, declarada com a una classe a part de la vista, es tracta d'una reimplementació de JTabbedPane on cada un dels panells a seleccionar és tracta d'un JScrollPane que conté un JTable reimplementat que hem anomenat TaulaFull. Aquest es tracta d'una taula on les files estan numerades i les columnes ordenades alfabèticament. En aquest es permet, a més, afegir i eliminar columnes de qualsevol part de la taula, com també es poden moure les columnes de lloc. Quan es selecciona una cel·la o un conjunt de cel·les el número de la/les fila/es quedarà de color vermell.

  Pel que fa al JTabbedPane de l'estructura, aquest s'ha implementat també que les pestanyes de canvi de full continguin una petita creu (que es posa de color vermell quan es passa per sobre el ratolí) per a poder tancar el full. A més, a la dreta hi ha una pestanya que conté un símbol de "+" que quan es premut es crea una nova pestanya i, amb ella, un nou full.

4.1.9 PanellFulls

En aquesta classe ens ha calgut una estructura de dades per a guardar de cada Full el seus Default Table Model i el JTable. Per poder cambiar de Default Table Model i de JTable depenent del full on ens trobem, vàrem decidir utilitzar dos HashMaps amb identificador idFull. HashMap <Integer, DefaultTableModel> i HashMap<Integer, Jtable>.

També hem fet servir un ArrayList<ArrayList<Cel·la>>.

- **HashMap**

  El HashMap, ens ha estat molt útil a l'hora de poder cambiar de Defaut Table Model i JTable a la vegada que canviàvem de Full, també ens ha servit per tindre tots els Default Table Model i JTable de tots els fulls

guardats, i en el cas de si s'elimina algún Full, eliminar també el seu Default Table Model i el seu JTable associats.

- **ArrayList**
  En aquest cas, hem utilitzat un ArrayList<ArrayList<Cel·la>>, per anar guardant els valors en cas de copiar/retallar els valors d'un Bloc.
  Vàrem decidir utilitzar un ArrayList ja que en aquest cas no fa falta declarar al inici la dimensió d'aquesta, i es poden afegir i eliminar valors sense preocupació.