A programozás alapjai 3. – HF dokumentáció

Ignácz Tamás, 2025

1. Feladat

Mini táblázatkezelő program. Adatok bevitele, módosítása, rajtuk értelmezett függvények végrehajtása, a cellaértékek megjelenítése, illetve táblázatok mentése, betöltése. Grafikus felületen keresztül való kezelés.

2. Felhasználói dokumentáció

- Cellába írás: dupla klikk/kiválasztás és írás, véglegesítés: *enter*.
- Cellákat kijelölve: másolás: *ctrl+c* (egy a programon belüli vágólapra); beillesztés: *ctrl+v* (mindig bal felső cellához illeszt); törlés: *delete*. Sorokra/oszlopokra az összes oszlop/sor kijelölhető az adott sorban/oszlopban, *shift* nyomvatartásával egy kiválasztás bővíthető.
- Az oszlop/sor fejlécben két sor/oszlop határán egérgombot lenyomva tartva változtatható a szélesség/magasság.
- Formulák: *„{ }”* között, a formulán belül: cella referenciák: *„:x:y:”,* ahol x az oszlop, y a sorszám; függvények: *„SUM(x1,y1,x2, y2)”* ahol (x1, y1) a bal felső, (x2, y2) a jobb alsó sarok, rosszul formázott/rekurzív formulát a program nem fog kiértékelni. Értelmezett függvények: *SUM(), AVG(), COUNT(), MIN(), MAX().*
- File kezelés: bal felső sarokban lévő gomb, a megjelenő menüben bevihető a file elérési útvonal majd a megfelelő gomb véglegesíti, a *read* funkciónál az első mező az értékeket elválasztó karakter. A file formátumát a program nem teszteli.

3. Kód dokumentáció

class App

- *public App()*
  Konstruktor, a táblázat és a frame létrehozása.
- *static private void createControls()*
  Létrehozza a file io-hoz szükséges swing komponenseket és a kiválasztott cellát megjelenítő szöveget.
- *public static void main(String[] args)*
  Main

class Table

- *public Table(JFrame f, DefaultTableModel m)*
  A táblázatot inicializáló függvények meghívása, fejlécek, editor és renderer beállítása.
- *public JScrollPane getScrollPane()*
- *public JTextField getSelectedValue()*
- *public void reset()*
  A formulák által használt adatszerkezetek és a program vágólapjának visszaállítása.

- *public void setValueAt(Object aValue, int row, int column)*
  Invalid sorra vagy oszlopra nem dob kivételt és a táblázat méreténél nagyobb sor vagy oszlop kérésére újakat hoz létre.
- *public void fill()*
  Feltölti a táblázat megjelenített részét sorokkal/oszlopokkal.
- *private void addListeners()*
  Listenerek hozzáadása:
  A *fill()*-hez szükséges *ComponentListener* és *MouseWheelListener*,
  A kiválasztott érték megjelenítéséhez szükséges *ListSelectionListener*,
  A táblázat frissítésére reagáló *TableModelListener*, ez hozza létre és értékelteti ki a formulákat.
- *private void updateCellsRecursive(Point p, Table t)*
  Formulák rekurzív kiértékelése a paraméterben megadott koordinátájú cella változására.
- *private void mapInputs()*
  Ctrl + c/v és *delete* parancsok megvalósítása.

class CellRenderer

- *public CellRenderer()*
  Konstruktor, a cellában a szöveg több sorba írását valósítja meg.
- *public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column)*

class CellEditor

- *public CellEditor()*
- *public Component getTableCellEditorComponent(JTable table, Object value, boolean isSelected, int row, int column)*
  Ha egy cella formulát tartalmaz, akkor magát a formulát jeleníti meg az értéke helyett.

class ColumnHeader

- *public ColumnHeader(Table table)*
  Konstruktor, egy *MouseListener*-el valósítja meg az oszlopok kiválasztását.

class RowHeader

- *public RowHeader(Table table)*
  MouseMotionListener-t, *MouseListener*-t és egy egyedi renderer-t ad hozzá, megvalósítja a sorok kiválasztását és átméretezését.

class TableIO

- *public static void load(Table t, String s)*
  Beolvas egy elmentett táblázatot.
- *public static void save(Table t, String s*
  Elment egy táblázatot

- *public static void read(Table t, String file, char delimiter)*
  Beolvas egy file-ból, az adott delimiter-rel elválasztott értékekből álló táblázatot.

class TableData

- *public TableData(Table t)*
  Egy adott táblázatból létrehoz egy könnyen menthető/beolvasható adatstruktúrát.
- *public TableData(ArrayList<String[]> input)*
  Egy adott *String* mátrixból létrehoz egy könnyen menthető/beolvasható adatstruktúrát.
- *public void readData(Table t)*
  A táblázatba írja az adatokat.

class FilePopUp

- *public FilePopUp(Table table, JFrame frame)*
  A file io-t vezérő popup menü.

class CellFormula

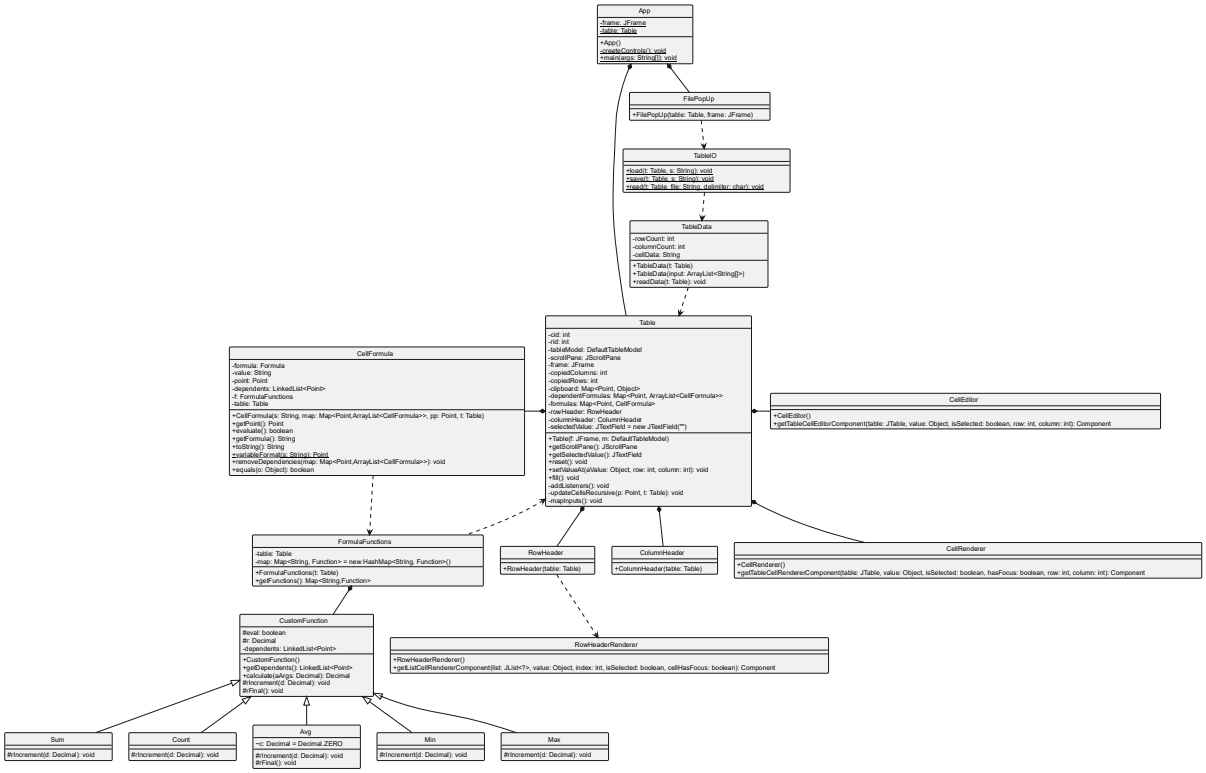
- *public CellFormula(String s, Map<Point, ArrayList<CellFormula>> map, Point pp, Table t)*
  Konstruktor, kiértékeli a formulát, beállítja a cella referenciákból közetlenül származó függőségeket, majd beállítja a függvényekből következő függőségeket is.
- *public Point getPoint() public boolean evaluate()*
  A formula koordinátája.
- *public String getFormula()*
  A cellába beírt formulát adja vissza.
- *public String toString()*
  A kiértékelés eredményét adja vissza.
- *public static Point variableFormat(String s)*
  Cella koordinátaként értelmez egy string-et.
- *public void removeDependencies(Map<Point, ArrayList<CellFormula>> map)*
  Kitörli az erre a cellára mutató függőségeket.
- *public boolean equals(Object o)*
  Összehasonlítás.

class FormulaFunctions

- *public FormulaFunctions(Table t)*
  Létrehozza a függvényeket.
- *public Map<String, Function> getFunctions()*
- *class CustomFunction*
  - public CustomFunction()
    Változó incializálás.
  - public LinkedList<Point> getDependents()
    A függvény által elért cellák koordinátájának visszaadása.

- public Decimal calculate(Decimal... aArgs)
  A paraméterekben megadott két pont közötti változókat hozzáadja a *Dependents* listához és meghívja az *rIncrement* függvényt rájuk, a végeredmény visszaadása előtt meghívja az *rFinal* függvényt.
- protected void rIncrement(Decimal d) {}
  A leszármazottaknak szánt, minden cellára elvégzett függvény.
- protected void rFinal() {}
  A leszármazottaknak szánt, csak egyszer elvégzett függvény.

4. UML



5. Egyéb

- Felhasznált könyvtár: https://johanley.github.io/formula4j/index.html