

Bevezető, ALAP ISMERTETŐ

A program Java nyelven íródott, a grafikus megjelenítéshez a Swing-et használja. Az alkalmazás egy XML fájlt kezel, melybe új bejegyzéseket tudunk felvinni, valamint tudunk belőle bejegyzést törölni, a bejegyzés adatait módosítani és „tartalmazó jelleggel” keresni köztük. A bejegyzések az úgymond „tranzakciók”, egy tranzakció 5 adatot hordoz, ezek a következők: a tranzakció azonosító száma (erről a későbbiekben még részletesen is lesz szó), pénzforgalom típusa (bevetel/kiadás), dátum, összeg és végül a leírás.

Nos, azoknak az azonosítóknak a funkciója, hogy tudjunk módosításnál, törlésnél pontosan hivatkozni az adott bejegyzésre. Illetve amikor a típus, dátum, összeg, leírás között tartalmazó jelleggel (nem szükséges a pontos egyezés) keresünk, ezeket az értékeket jeleníti meg a program.

Itt fontos megemlíteni még, hogy ezeknek az azonosítóknak a törlésnél „csak” közvetett szerepük van... Ugyanis az XML fájlból törölni technikailag index/sorszám szerint lehet. Ezért erre az alábbi megoldási elvet dolgoztam ki:

- Van egy *c_variable.txt* fájlunk, amiben kezdetben van egy „0” (bár igazság szerint bármilyen szám lehet...).
 - Amikor új elemet veszünk fel az XML fájlba, akkor ennek a változónak az értékét növeljük 1-gyel.
 - Ezt az új értéket adjuk az új elemünk (tranzakciónk) azonosítójának.
 - Ekkor a tranzakciók azonosító száma 1-től n-ig tartó sorozatot fog alkotni, ahol n az n.-ik tranzakciót jelöli.
 - Ha most 1 és n között kitörlünk egy értéket, akkor ez a sorozat már hiányos lesz, de az még továbbra is igaz, hogy az egymást követő értékek növekvőek.
 - A következő törlésnél már nem törölhetünk azonosító szerint, de azt tudjuk hogy eddig az azonosítóig kell keresni az új indexet/sorszámot. Így eddig az azonosítónak az értékéig összeszámoljuk az elemeket, majd ez a szám lesz az új index/sorszám, ami szerint már ismét tudunk törölni az adatbázisból.
- Megjegyzés: Az előző algoritmuská nagyon nagy bejegyzésmennyiségnél nem túl hatékony. Másik megoldási ötlet lehetne az, hogy a törlések darabszámát naplózzuk (minden törlésnél növelünk egyet egy változón) mindig, és ezt kivonjuk az azonosító számból...

A *db_to_scr.txt* és *search.txt* fájlok nem az elemek tartós tárolására, hanem az XML fájl és a grafikus panel közti ún. bufferelésre lettek kitalálva (szertintem struktúráltabb így megoldani...).

A program main osztálya és metódusa az APPLICATION.java fájlban található. Itt a GUI_Main metódust hívjuk meg, mely az azonos nevű JFrame típusú fájlban található.

Ez utóbbi fájl felelős a főablak megjelenéséért, label-ekért, gombokért... A gombok eseményeit kezelő metódusok is itt vannak meghívva, ezek a következők:

Element_printing_GUI, Element_modify_GUI, Element_remove_GUI, Ki, Main_screen_function, New_element

Ezen esetekben a további metódushívásokat az jellemzi, hogy minden ilyen „GUI” végződésű metódust tartalmazó osztálynak van egy „GUI” végződés nélküli megfelelője, ezeket a már meghívott „GUI” végződésű metódusok hívják meg. Értelem szerűen az egyik fajta a grafikus felület megjelenését, a másik pedig a háttérműveleteket (pl. fájl írások-olvasások...) végzi.

A további, egyéb metódusokról a következő oldalon talál részletesebb infót:

OSZTÁLYOK, osztályok METÓDUSAI, + megjegyzések

A programomban használt osztály és változónevek listája az osztályok nevének alfabetikus sorrendje szerint:

```
APPLICATION.java           //main osztály
(main)                     //main metódus, amely meghívja a Main_GUI-t
All_element_toTXTprinting.java
(All_element_toTXTprinting_function)    //itt kiírjuk az összes elemet egy txt-be... (buffer létrehozás)
Element_modify.java
(Element_modify_funciton, Element_modify_funciton2 ... Element_modify_funciton4)
Element_modify_GUI.java
(Element_modify_GUI, actionPerformed)
Element_printing.java      //ennek a metódusa txt-be írja a keresett elemek azonosító számait (buffer készítés)
(Element_printing_function)    //meghívjuk a Ki2 metódust
Element_printing_GUI.java
(Element_printing_GUI, actionPerformed)
Element_printing_out_GUI.java, Kiír
(main, Ki2)                //a Ki2 visszaolvassa a txt-be kiírt keresett azonosítókat
Element_remove.java
(Element_remove_function, prettyPrint)
Element_remove_GUI.java
(Element_remove_GUI, actionPerformed)
Full_db_to_screen.java, Kiír
(main, Ki)                  //itt visszaolvassuk a txtből az elemeket, majd a képernyőre írjuk (bufferolvasás)
GUI_Main.java
(GUI_Main, actionPerformed)
Main_screen.java           //itt számolódik annak a speciális „egyensúly”-nak az értéke
(Main_screen_function)     //megszámolja hány bejegyzés van az XML-ben (ehhez kell a c_variable txt),
ill. mennyi az összeg tag-ek szummája... => Innen számolódik az „egyensúly” szám a fő ablakba...
New_element.java
(New_element, actionPerformed)    //ezek felelősek a GUI-ért, meghívódik a következő metódus...
New_element_to_node.java
(ERROR_nodetag_bovito)        //bejegyzést adunk hozzá az XML struktúránkhoz
```

Az XML fájl szerkezete

Most az előbbieken alapján felvázolnám az XML fájlom struktúráját (feltételezem, hogy a dokumentáció elejét már elolvasta), amelyet az alábbiakban olvashat, és amelyben értelemszerűen a tranzakciók „szintjéhez” fűzzük hozzá az új elemeket. Az XML fájlban mindez egy sorban jelenik meg, de az úgy van „jól”.

```
<header>
  <transaction>
    <azon>
    <type>
    <date>
```

