Programozói dokumentáció

Total Commander

Total Commander

Rövid Leírás

Programom egy Total Commander másolat, melyben grafikus interfészen keresztül lehet elérni a különböző menüpontokat, pontosan, mint az eredeti Total Commanderben. A kinézete megszólalásig hasonlítani fog az eredetire (vagy nem☹).

A program tartalmazni fog:

- 2 db táblázatot, melyben külön-külön lehet dolgozni:
  - Lépkedni a mappák között(Ha egy mappához nincs jogosultsága a felhasználónak, a program egy figyelmeztető ablakkal jelez).
  - Egyikből a másikba fájlműveleteket végezni.
- Mindkét táblázathoz egy lenyíló menüt, melyben a kívánt tárolóeszközre lehet lépni.
- Teljes és szabad területet, az adott tárolóeszközön.
- Mindkét táblázathoz az aktuális elérési útvonalat.
- Gombokat a fájlműveleteknek.

A táblázat oszlopai:

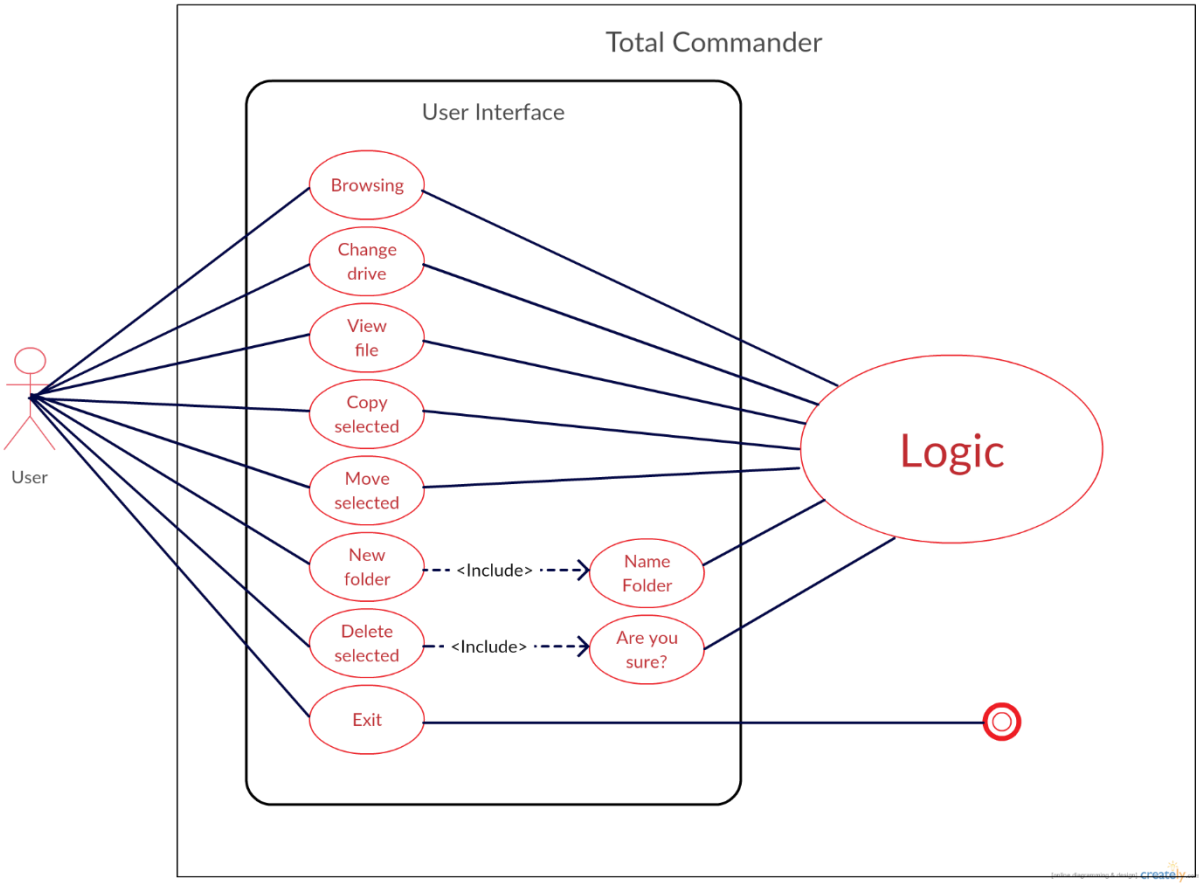
(Icon)	Name	Ext	Size	Date

A Total Commander alsó menüsora is meg lesz valósítva, ezek az előbb említett fileműveletek:

- F3 View
- F5 Copy
- F6 Move
- F7 NewFolder
- F8 Delete
- Alt + F4 Exit

Ezek a műveltek úgy fognak működni, ahogy az elvárt. Pl a Delete művelettel nem csak a kijelölt mappa, hanem minden benne lévő mappa és fájl is törlődik.

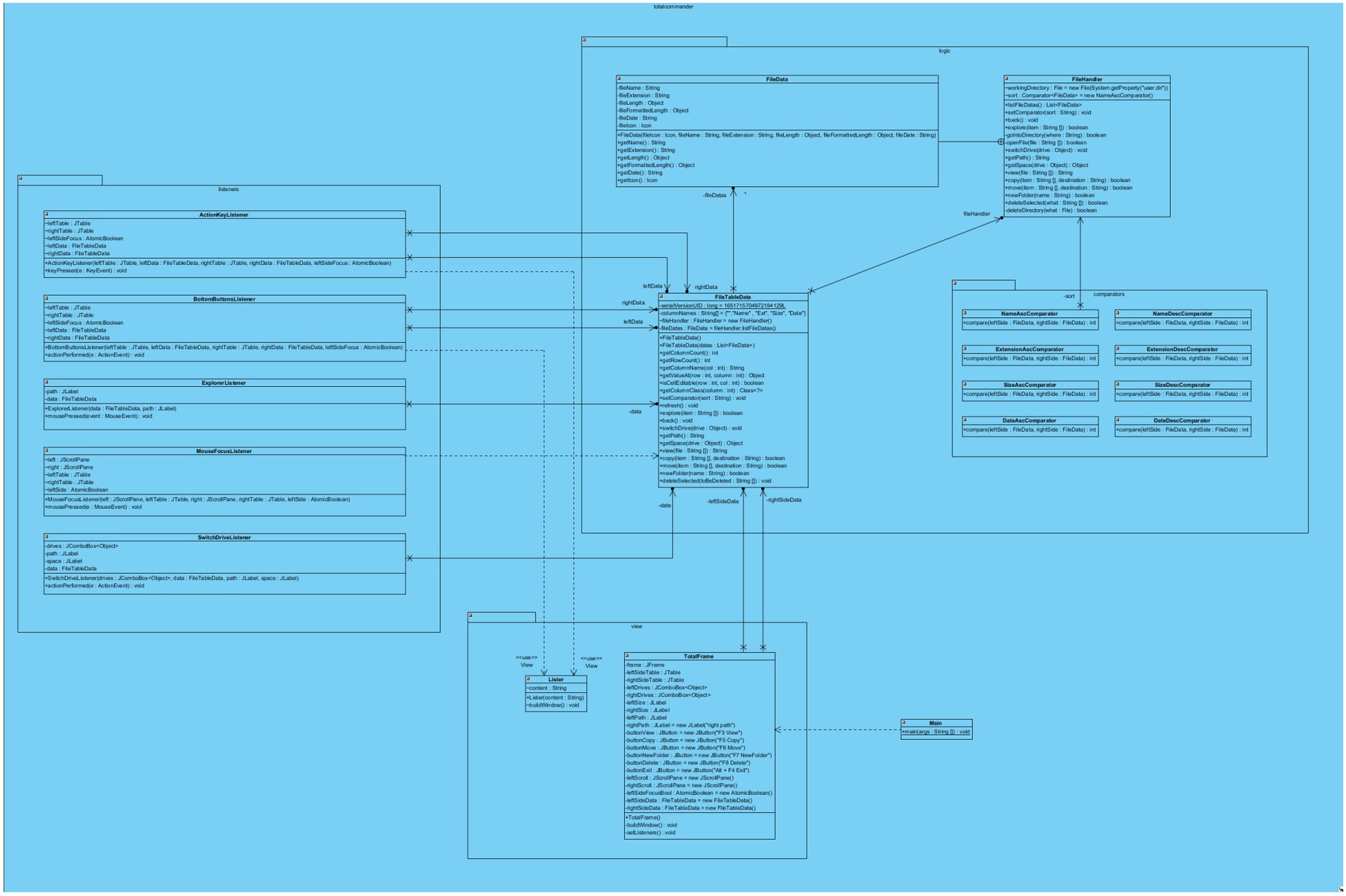
Use-case diagramm



Megvalósítás

A program Javaban lesz megírva. A grafikus interfészhez a Swing könyvtárat veszem segítségül. A fájlokat/mappákat tartalmazó táblázat egy-egy JTable. Az aktuális tárolóeszközt egy JComboBox-ból lehet kiválasztani. A fájlműveletekhez JButtonöket használok.

Osztálydiagram



Osztályok metódusai

FileData

A FileHandler class FileData objektumokkal foglalkozik. A FileData össze attribútumára van getter.

FileHandler

public void back()

Ha nem root directoryban vagyunk, fellép eggyel a mappaszerkezetben és az lesz az új working directory.

public boolean copy(String[] item, String destination)

A kapott paraméterekből felállítja forrás és cél File útvonalat, majd elvégzi a másolást. False-szal tér vissza, ha sikertelen a másolás, true-val ha sikeres.

public boolean deleteSelected(String[] what)

A kapott paraméterből felállítja a törölni kívánt File útvonalát. Ha az útvonal mappára mutat, meghívja a private deleteDirectory metódust, ha fájl, elvégzi a törlést. False-szal tér vissza, ha sikertelen a törlés, true-val ha sikeres.

private boolean deleteDirectory(File what)

A deleteSelected metódustól kapott File paramétrerre, ami egy mappára mutat, elvégzi a törlést. A függvény rekurzív, tehát a mappán belüli elemeket is kitörli. False-szal tér vissza, ha sikertelen a törlés, true-val ha sikeres.

public boolean explore(String[] item)

Ha a kapott paraméter egy File-ra mutat, a private openFile metódust hívja meg, ha mappára, a private goIntoDirectory metódust. Ha sikerült a fájl megnyitása/mappába való belépés, true-val tér vissza, egyébként false-szal.

public String getPath()

Visszaadja formázva a working directory útvonalát.

public String getSpace()

Visszaadja formázva, hogy az adott drivenak mekkora a tárolóképessége és ebből mennyi foglalt.

private boolean goIntoDirectory(String where)

A kapott paraméterből felépíti a mappa File útvonalát, majd belelép. Ha sikerült a belépés true-val tér vissza, egyébként false-szal.

public List<FileData> listFileDatas()

Felépíti egy listába a working directory-ban tárolt minden fájlt és mappát FileData objektumokban. Ezt a listát rendezi az attribútumként tárolt Comparator-ral majd visszatér vele.

public boolean move(String[] item, String destination)

A kapott paraméterekből felállítja forrás és cél File útvonalat, majd elvégzi az áthelyezést. False-szal tér vissza, ha sikertelen az áthelyezés, true-val ha sikeres.

public boolean newFolder(String name)

Létrehoz egy új mappát a working directoryba. A mappa neve a paraméterként kapott String lesz. Ha sikerül a mappa létrehozása, true-val tér vissza, egyébként false-szal.

private boolean openFile(String[] file)

A kapott paraméterből felépíti a fájl File útvonalát, majd a beépített jawa.awt.Desktop class segítségével megnyitja a programot az operációs rendszer alapértelmezett segédprogramjával. Ha sikerült a fájl megnyitása, true-val tér vissza, egyékbént false-szal.

public void setComparator(String sort)

A függvény a paraméterként kapott String szerint eldönti, mi lesz az új Comparator, majd az attribútumként tárolt Comparator-t felülírja az újjal.

public boolean swtichDrive (Object drive)

A kapott paraméterből, mely egy drive root útvonala, felépíti a File útvonalat, majd ha ez létezik a working directory-t felülírja az újra. Ha sikerült átlépni az új drive-ra, true-val tér vissza, egyékbént false-szal.

public String view(String[] file)

A kapott paraméterből felépíti a fájl File útvonalát, majd ezt megpróbálja a java.util.Scanner osztály segítségével beolvasni. A beolvasott String-gel tér vissza. Ha nem sikerült a beolvasás null-lal tér vissza.

FileTableData (extends AbstractTableModel)

A FileTableData osztály egy átmenetet képez a JTable és a FileHandler között. Minden egyes függvénye meghívja a FileHandler beli megfelelőjét, majd ha szükséges, frissíti a táblázatot.

Comparators (implements Comparator<FileData>)

A JTable 4 oszlopa szerint lehet rendezni. Ez 4 Comparator lenne, viszont a növekvő és csökkenő nem egymásnak az ellentéte ebben az esetben. A mappák mindig előrébb (feljebb) helyezkednek el, utánuk jönnek a fájlok. Ezért 8 db Comparator megírása volt szükséges. Ezek a nevük szerint működnek.

ActionKeyListener (extends KeyAdapter)

public void keyPressed(KeyEvent e)

A class egyetlen függvénye. A billentyűzetről beadott fájlműveleteket kezeli le. Ezen kívül foglalkozik figyelmeztető- és hibaablakok megjelenítésével is. A következő gombnyomás-fájlműveletek kezelésére képes:

- F3 – View
- F5 – Copy
- F6 – Move
- F7 – NewFolder
- F8 – Delete
- Alt + F4 – Exit

BottomButtonsListener (implements ActionListener)

public void actionPerformed(ActionEvent e)

A class egyetlen függvénye. Hasonlóképp működik, mint az előbb tárgyalt ActionKeyListener, viszont ez a class JButton-ok megnyomásának hatására hajtja végre a fájlműveleteket.

ExplorerListener (extends MouseAdapter)

public void mousePressed(MouseEvent event)

Ez az osztály kezeli le a fájlrendszerbeli mozgást. Egy JTable elemre duplakattintás segítségével tudunk böngészni. Ha az adott elem mappa, akkor belelépünk, ha fájl, megnyitjuk az operációs rendszer által alapértelmezett program segítségével.

MouseFocusListener (extends MouseAdapter)

public void mousePressed(MouseEvent e)

Az osztály dönti el, hogy a bal, vagy jobb oldali JTable volt utoljára aktív. Erre az osztályra épít a program egésze, hiszen nem mindegy melyik oldalra szeretnénk elvégezni az adott műveletet.

SwitchDriveListener (implements ActionListener)

public void actionPerformed(ActionEvent e)

A drive választó JComboBox eventjét kezeli le. Ha drive-ot váltunk, minden szükséges adat is frissül természetesen.

Lister

Az osztály a View fájlművelet segédosztálya. Segítségével létrehozunk egy új ablakot, melyre a megnyitott fájl tartalmát írjuk ki.

private void buildWindow()

Felépíti és megjeleníti az ablakot. A konstruktorban kerül meghívásra, nem kell külön meghívni.

TotalFrame

A program megjelenítéséért, elemek összekapcsolásáért felelős osztály.

private void buildWindow()

Felépíti és megjeleníti a TotalCommander programot.

private void setListeners()

A megfelelő elemeket összekapcsolja a megfelelő Listenerekkel.

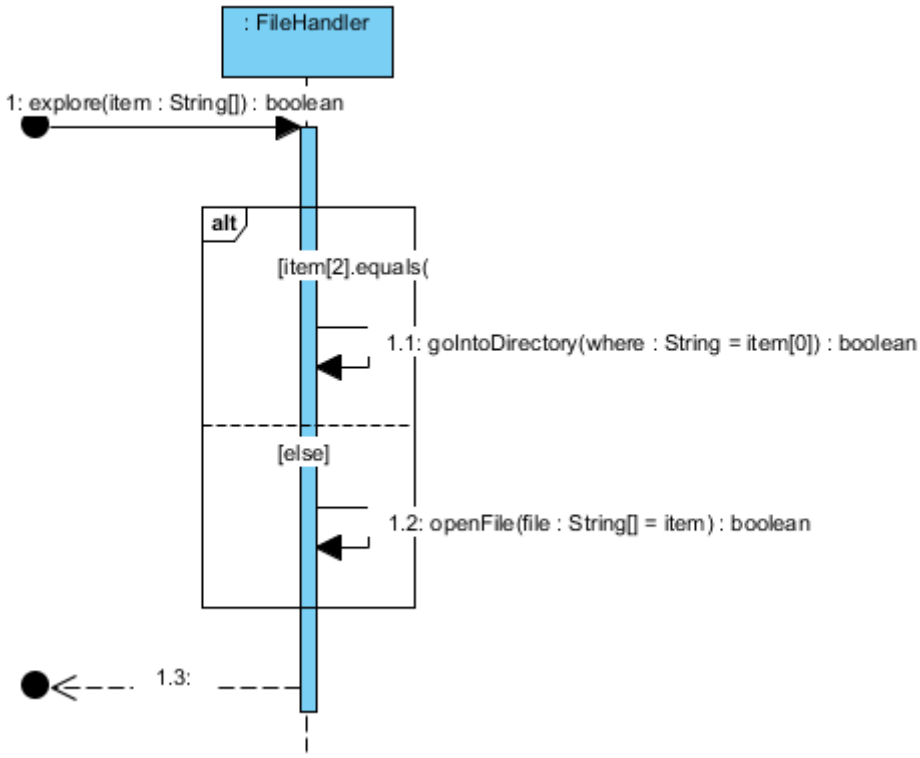
Main

public static void main(String[] args)

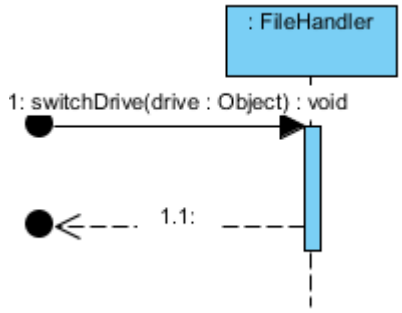
Létrehoz egy TotalFrame objektumot.

Szekvenciadiagramok

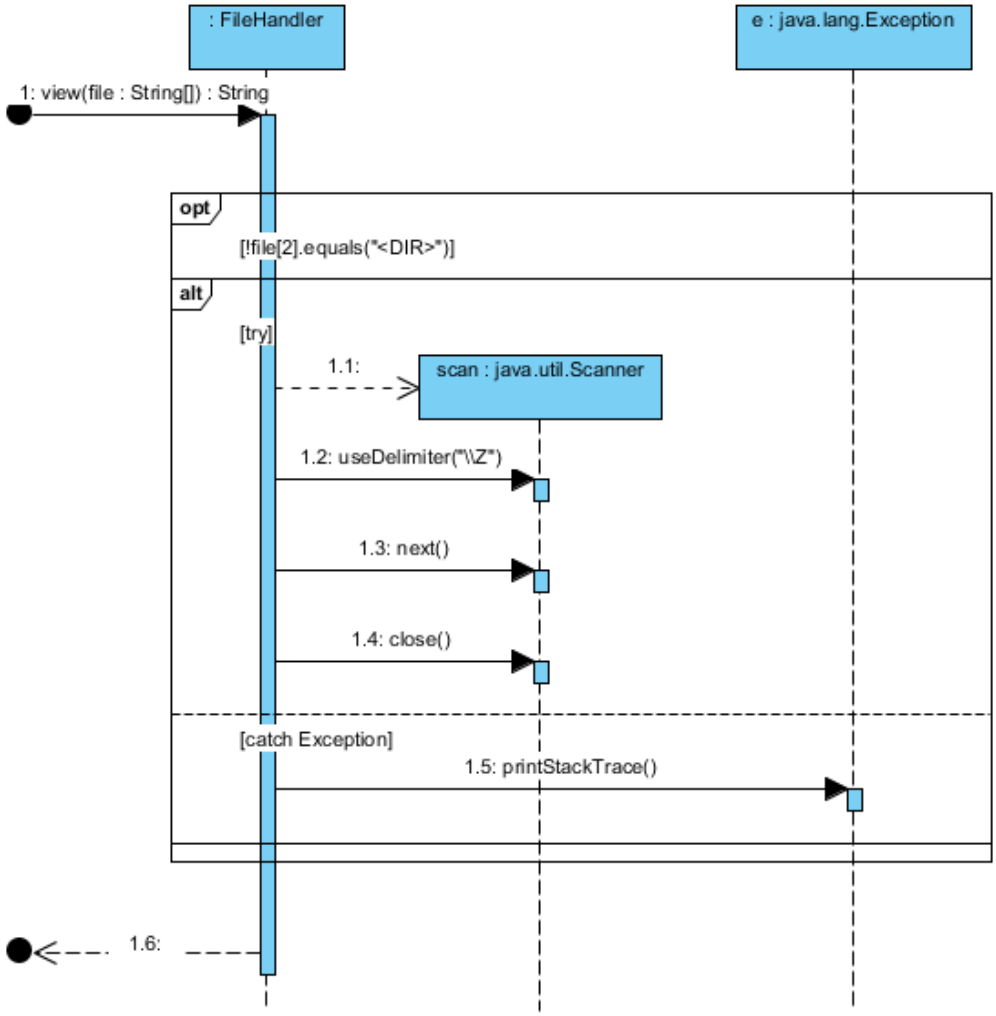
Browsing



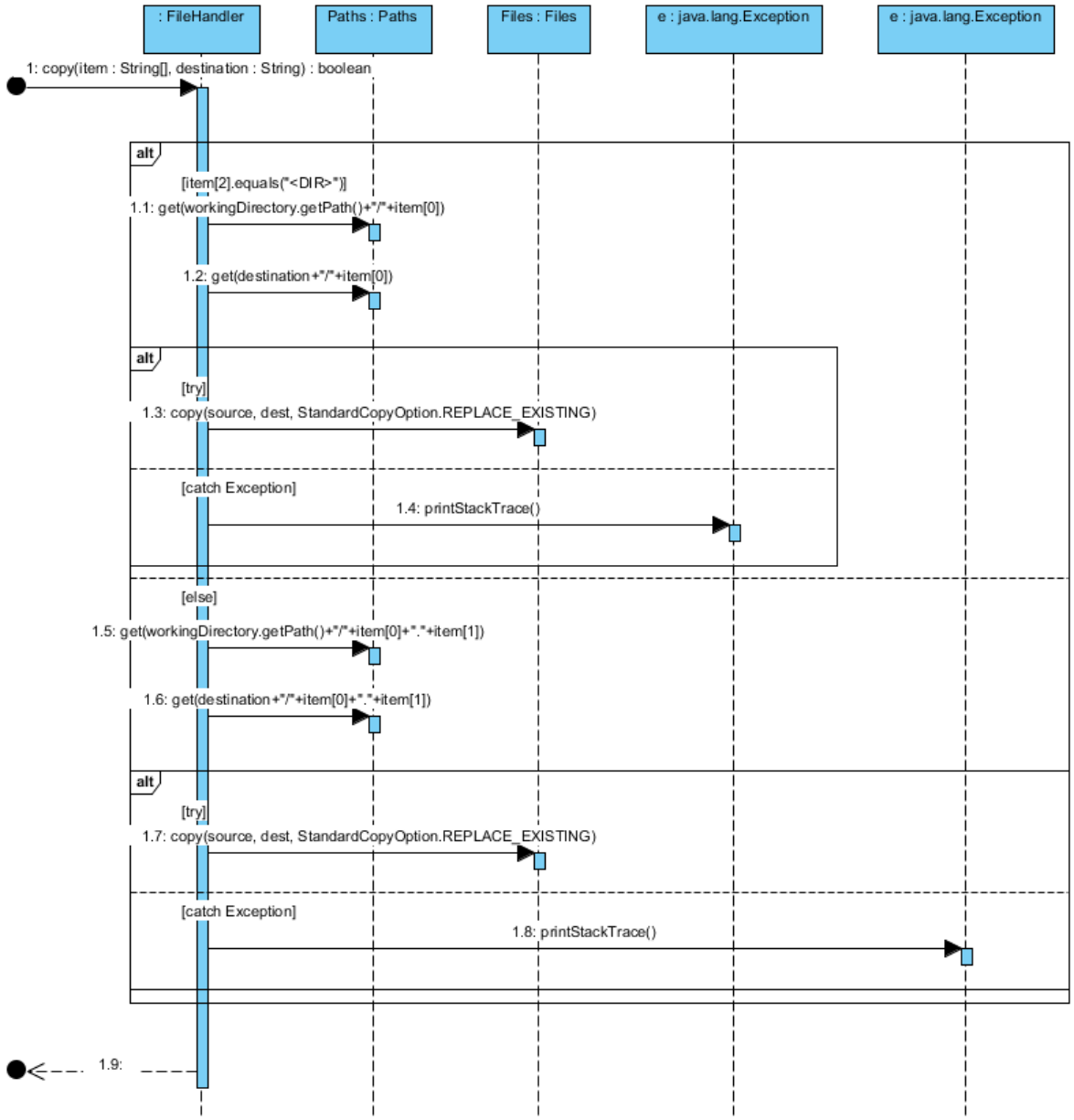
Change drive



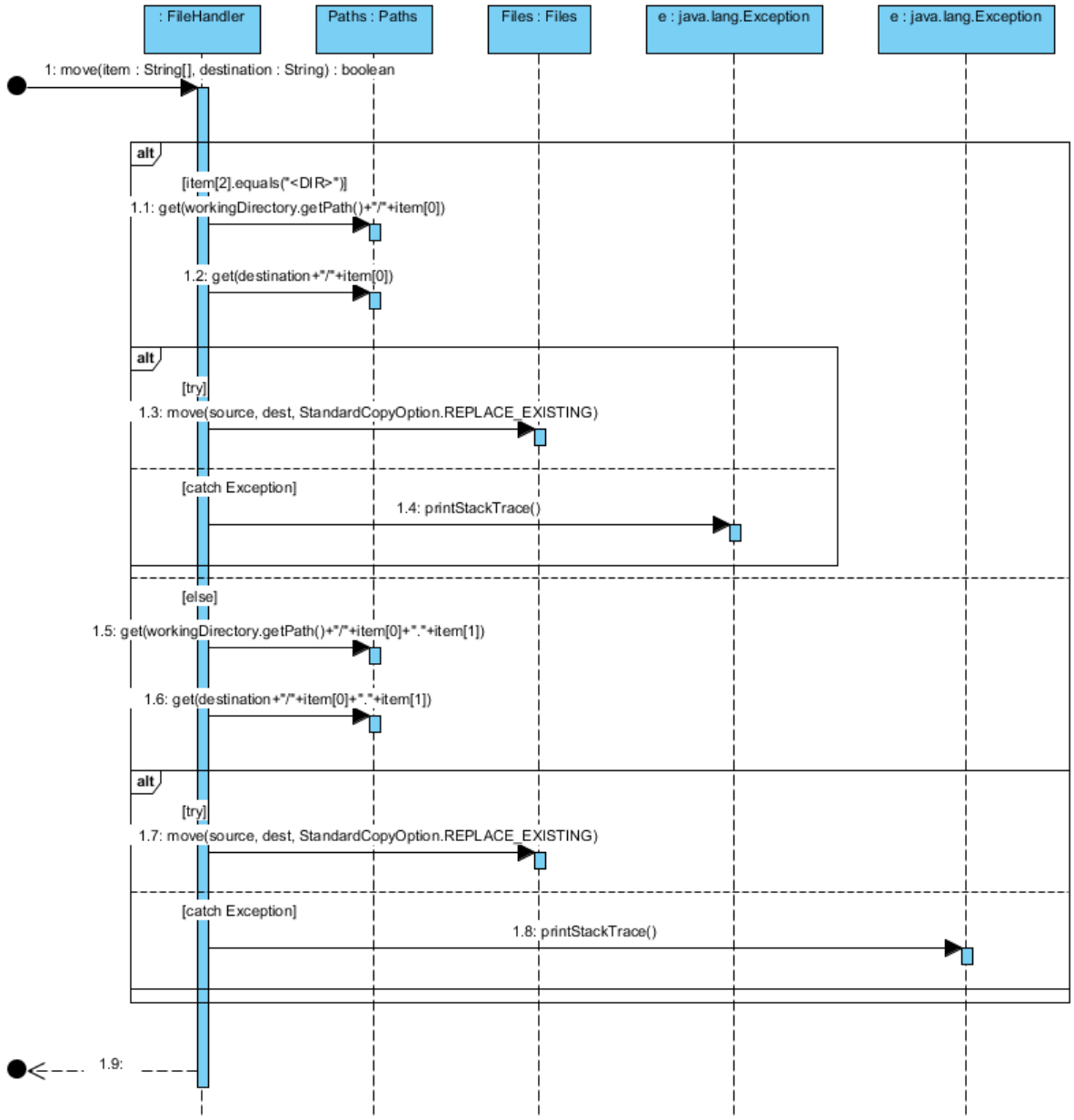
View file


: FileHandler
e : java.lang.Exception
1: view(file : String[]) : String
opt
[!file[2].equals("<DIR>")]
alt
[try]
1.1:
scan : java.util.Scanner
1.2: useDelimiter("\\Z")
1.3: next()
1.4: close()
[catch Exception]
1.5: printStackTrace()
1.6:

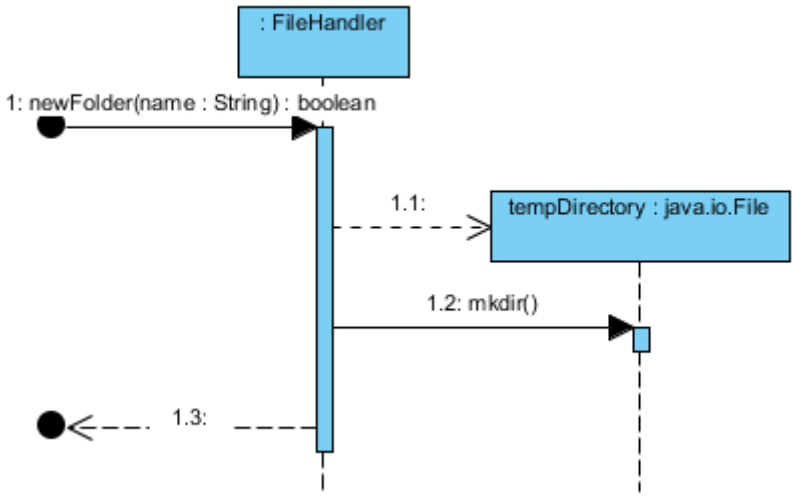
Copy selected



Move selected



New folder



Delete selected