

Programozói dokumentáció

Total Commander

| | |
|-----------------------------|----|
| Rövid leírás | 2 |
| Use-case diagram | 3 |
| Megvalósítás..... | 3 |
| Osztálydiagram..... | 4 |
| Osztályok metódusai | 5 |
| FileHandler..... | 5 |
| FileTableData | 6 |
| Comparators | 7 |
| ActionKeyListener..... | 7 |
| BottomButtonsListener | 7 |
| ExplorerListener..... | 7 |
| MouseFocusListener..... | 7 |
| SwitchDriveListener | 8 |
| Lister | 8 |
| TotalFrame..... | 8 |
| Main..... | 8 |
| Szekvenciadiagramok..... | 9 |
| Browsing | 9 |
| Change drive | 9 |
| View file | 10 |
| Copy selected..... | 11 |
| Move selected | 12 |
| New Folder | 13 |
| Delete Selected..... | 13 |

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 tárolóeszközt lehet lépni.
- Teljes és szabad területet, az adott tárolóeszközön.
- Mindkét táblázathoz az aktuális elérési útvonalat.
- Gombokat a fájlműveleteknek.

A táblázat oszlopai:

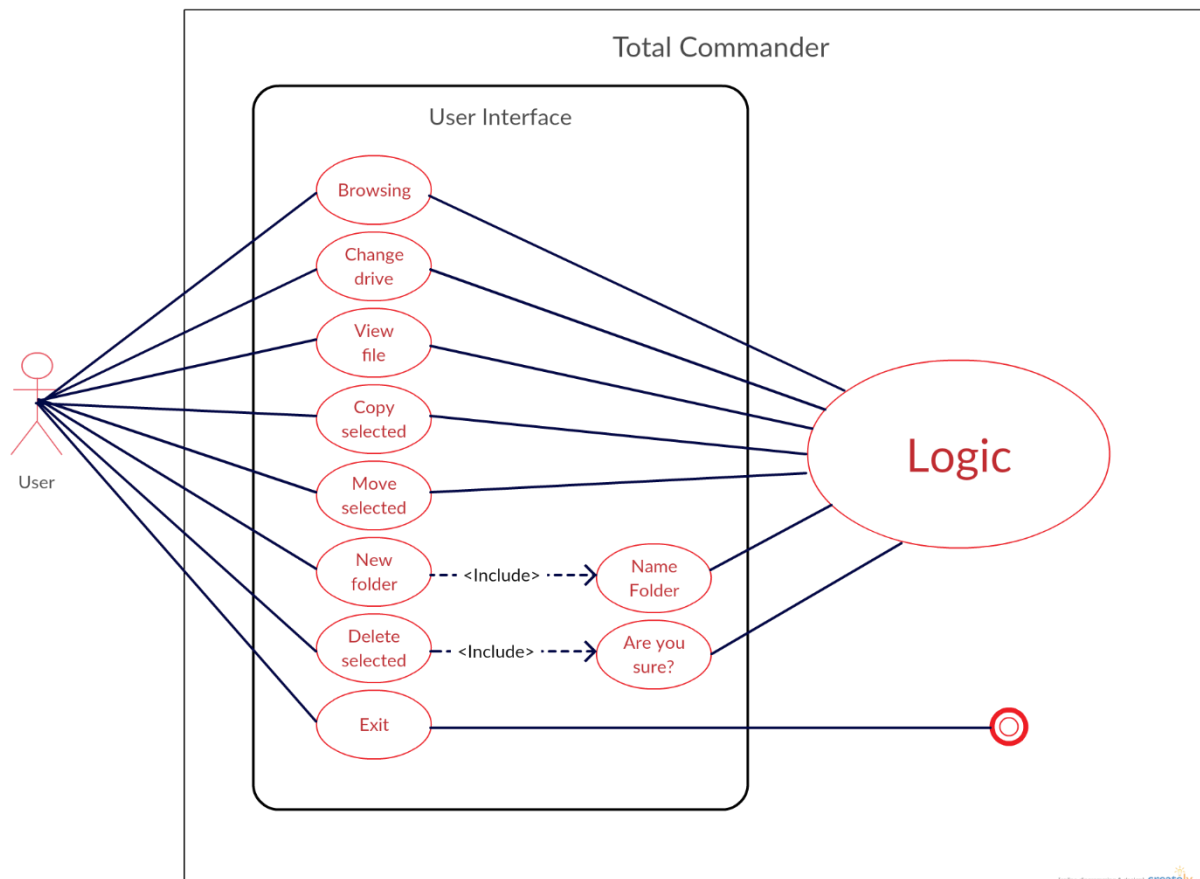
| (Icon) | Name | Ext | Size | Date |
|--------|------|-----|------|------|
|--------|------|-----|------|------|

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 műveletek 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 Javában 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.

```

classDiagram
    class FileData {
        -fileName : String
        -fileExtension : String
        -fileLength : Object
        -fileFormattedLength : Object
        -fileDate : String
        -fileIcon : Icon
        +FileData(fileIcon : Icon, fileName : String, fileExtension : String, fileLength : Object, fileFormattedLength : Object, fileDate : String)
        +getName() : String
        +getExtension() : String
        +getLength() : Object
        +getFormattedLength() : Object
        +getDate() : String
        +getIcon() : Icon
    }

    class FileHandler {
        -workingDirectory : File = new File(System.getProperty("user.dir"))
        -sort : Comparator<FileData> = new NameAscComparator()
        +File(FileData) : List<FileData>
        +setComparator(sort : String) : void
        +back() : void
        +explore(item : String[]) : boolean
        +getInfoDirectory(where : String) : boolean
        +getFiles(file : String[]) : boolean
        +switchDrive(drive : Object) : void
        +getPath() : String
        +getSpace(drive : Object) : Object
        +view(file : String[]) : String
        +copy(item : String[], destination : String) : boolean
        +move(item : String[], destination : String) : boolean
        +newFolder(name : String) : boolean
        +deleteSelected(what : String[]) : boolean
        +deleteDirectory(what : File) : boolean
    }

    class FileTableData {
        -serialVersionUID : long = 1651715704972194129L
        -columnNames : String[] = {"Name", "Ext", "Size", "Date"}
        -fileHandler : FileHandler = new FileHandler()
        -fileData : FileData = fileHandler.listFiles()
        +FileTableData()
        +FileTableData(datas : List<FileData>)
        +getColumnCount() : int
        +getRowCount() : int
        +getColumnNames(col : int) : String
        +getValueAt(row : int, column : int) : Object
        +isCellEditable(row : int, col : int) : boolean
        +getColumnClass(column : int) : Class<?>
        +setComparator(sort : String) : void
        +refresh() : void
        +explore(item : String[]) : boolean
        +back() : void
        +switchDrive(drive : Object) : void
        +getPath() : String
        +getSpace(drive : Object) : Object
        +view(file : String[]) : String
        +copy(item : String[], destination : String) : boolean
        +move(item : String[], destination : String) : boolean
        +newFolder(name : String) : boolean
        +deleteSelected(toBeDeleted : String[]) : void
    }

    class ActionKeyListener {
        -leftTable : JTable
        -rightTable : JTable
        -leftSideFocus : AtomicBoolean
        -leftData : FileTableData
        -rightData : FileTableData
        +ActionKeyListener(leftTable : JTable, leftData : FileTableData, rightTable : JTable, rightData : FileTableData, leftSideFocus : AtomicBoolean)
        +keyPressed(e : KeyEvent) : void
    }

    class BottomButtonsListener {
        -leftTable : JTable
        -rightTable : JTable
        -leftSideFocus : AtomicBoolean
        -leftData : FileTableData
        -rightData : FileTableData
        +BottomButtonsListener(leftTable : JTable, leftData : FileTableData, rightTable : JTable, rightData : FileTableData, leftSideFocus : AtomicBoolean)
        +actionPerformed(e : ActionEvent) : void
    }

    class ExplorerListener {
        -path : JLabel
        -data : FileTableData
        +ExplorerListener(data : FileTableData, path : JLabel)
        +mousePressed(event : MouseEvent) : void
    }

    class MouseFocusListener {
        -left : JScrollPane
        -right : JScrollPane
        -leftTable : JTable
        -rightTable : JTable
        -leftSide : AtomicBoolean
        +MouseFocusListener(left : JScrollPane, leftTable : JTable, right : JScrollPane, rightTable : JTable, leftSide : AtomicBoolean)
        +mousePressed(e : MouseEvent) : void
    }

    class SwitchDriveListener {
        -drives : JComboBox<Object>
        -path : JLabel
        -space : JLabel
        -data : FileTableData
        +SwitchDriveListener(drives : JComboBox<Object>, data : FileTableData, path : JLabel, space : JLabel)
        +actionPerformed(e : ActionEvent) : void
    }

    class NameAscComparator {
        +compare(leftSide : FileData, rightSide : FileData) : int
    }

    class NameDescComparator {
        +compare(leftSide : FileData, rightSide : FileData) : int
    }

    class ExtensionAscComparator {
        +compare(leftSide : FileData, rightSide : FileData) : int
    }

    class ExtensionDescComparator {
        +compare(leftSide : FileData, rightSide : FileData) : int
    }

    class SizeAscComparator {
        +compare(leftSide : FileData, rightSide : FileData) : int
    }

    class SizeDescComparator {
        +compare(leftSide : FileData, rightSide : FileData) : int
    }

    class DateAscComparator {
        +compare(leftSide : FileData, rightSide : FileData) : int
    }

    class DateDescComparator {
        +compare(leftSide : FileData, rightSide : FileData) : int
    }

    class TotalFrame {
        -frame : JFrame
        -leftSideTable : JTable
        -rightSideTable : JTable
        -leftDrives : JComboBox<Object>
        -rightDrives : JComboBox<Object>
        -leftSize : JLabel
        -rightSize : JLabel
        -leftPath : JLabel
        -rightPath : JLabel
        -buttonView : JButton = new JButton("F3 View")
        -buttonCopy : JButton = new JButton("F5 Copy")
        -buttonMove : JButton = new JButton("F6 Move")
        -buttonNewFolder : JButton = new JButton("F7 NewFolder")
        -buttonDelete : JButton = new JButton("F8 Delete")
        -buttonExit : JButton = new JButton("Alt + F4 Exit")
        -leftScroll : JScrollPane = new JScrollPane()
        -rightScroll : JScrollPane = new JScrollPane()
        -leftSideFocusBool : AtomicBoolean = new AtomicBoolean()
        -leftSideData : FileTableData = new FileTableData()
        -rightSideData : FileTableData = new FileTableData()
        +TotalFrame()
        +buildWindow() : void
        +setListeners() : void
    }

    class Main {
        +main(args : String[]) : void
    }

    FileData "1" -- "*" FileTableData : -fileData
    FileTableData "1" -- "*" FileHandler : -fileHandler
    FileTableData "1" -- "*" ActionKeyListener : -leftData, -rightData
    FileTableData "1" -- "*" BottomButtonsListener : -leftData, -rightData
    FileTableData "1" -- "*" ExplorerListener : -data
    FileTableData "1" -- "*" MouseFocusListener : -left, -right, -leftTable, -rightTable, -leftSide
    FileTableData "1" -- "*" SwitchDriveListener : -data
    FileTableData "1" -- "*" TotalFrame : -leftSideData, -rightSideData
    Main "1" -- "*" TotalFrame : -main
  
```

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 paraméterre, 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 rendezzi 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 java.awt.Desktop class segítségével megnyitja a programot az operációs rendszer alapértelmezett segédprogramjával. Ha sikerült a fájl megnyitása, true-val tér vissza, egyébként 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, egyébként 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 ActionListener, 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.

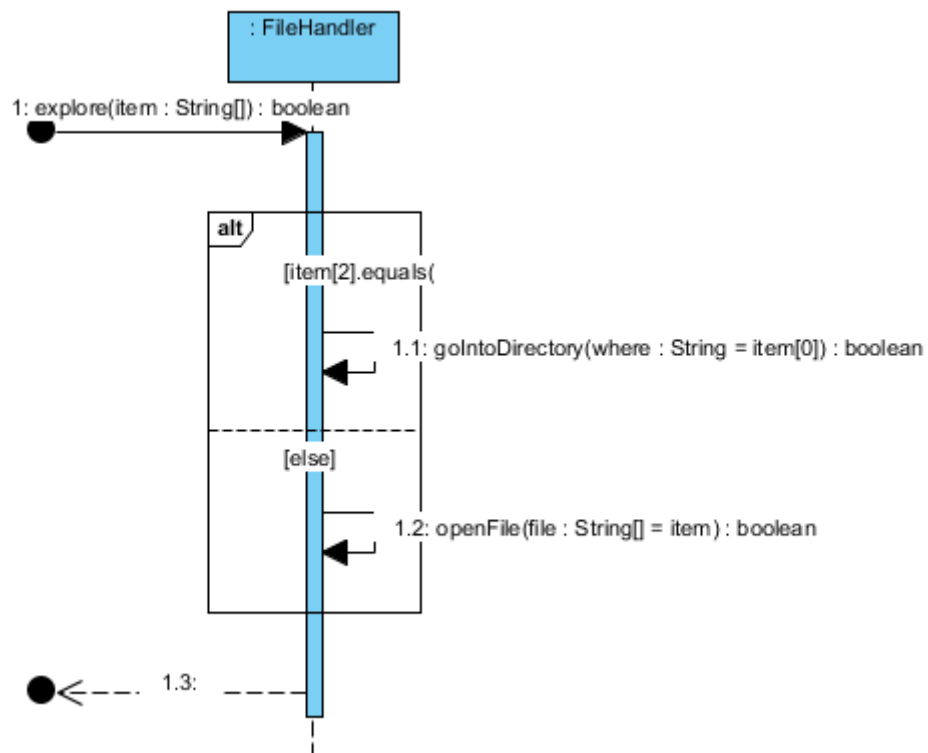
Main

```
public static void main(String[] args)
```

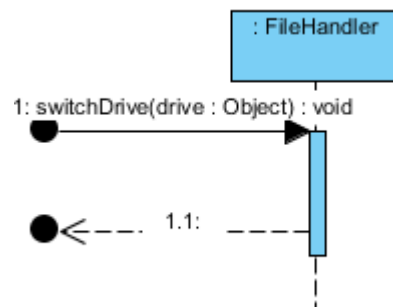
Létrehoz egy TotalFrame objektumot.

Szekvenciadiagramok

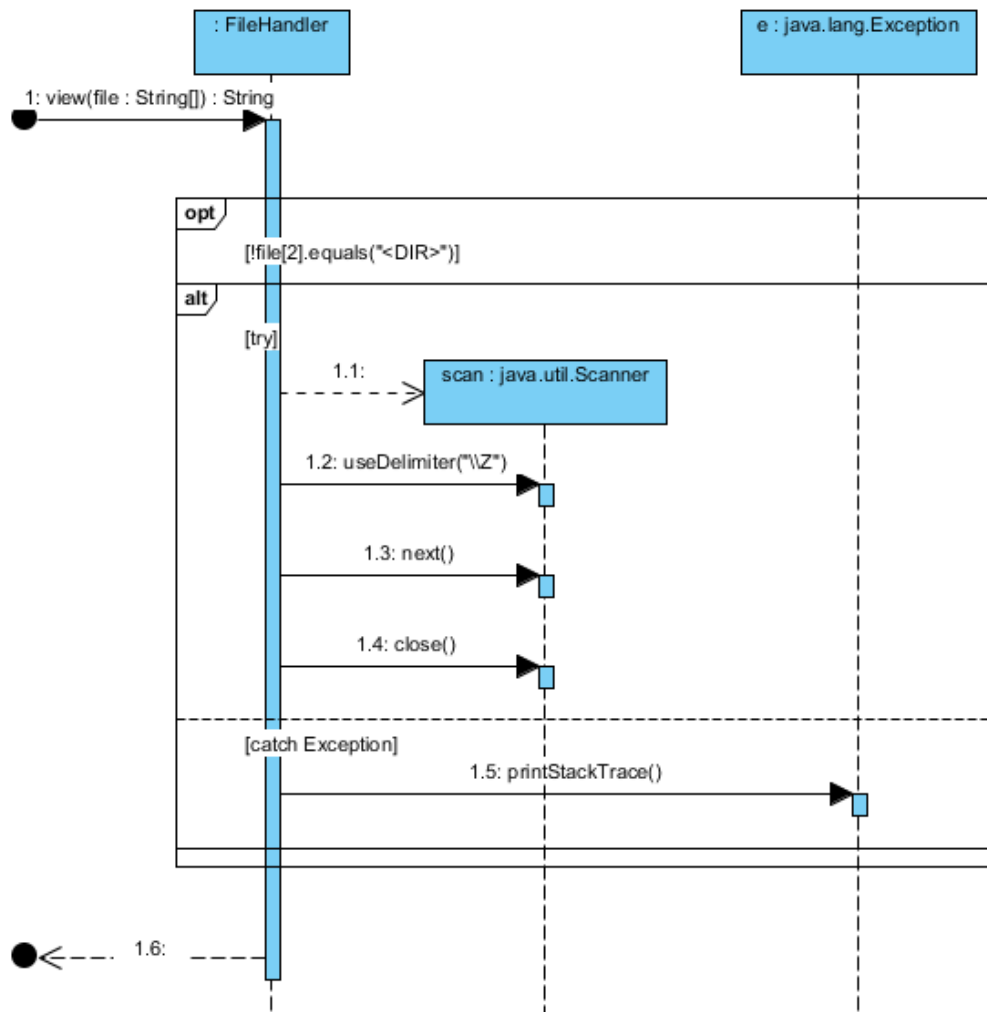
Browsing



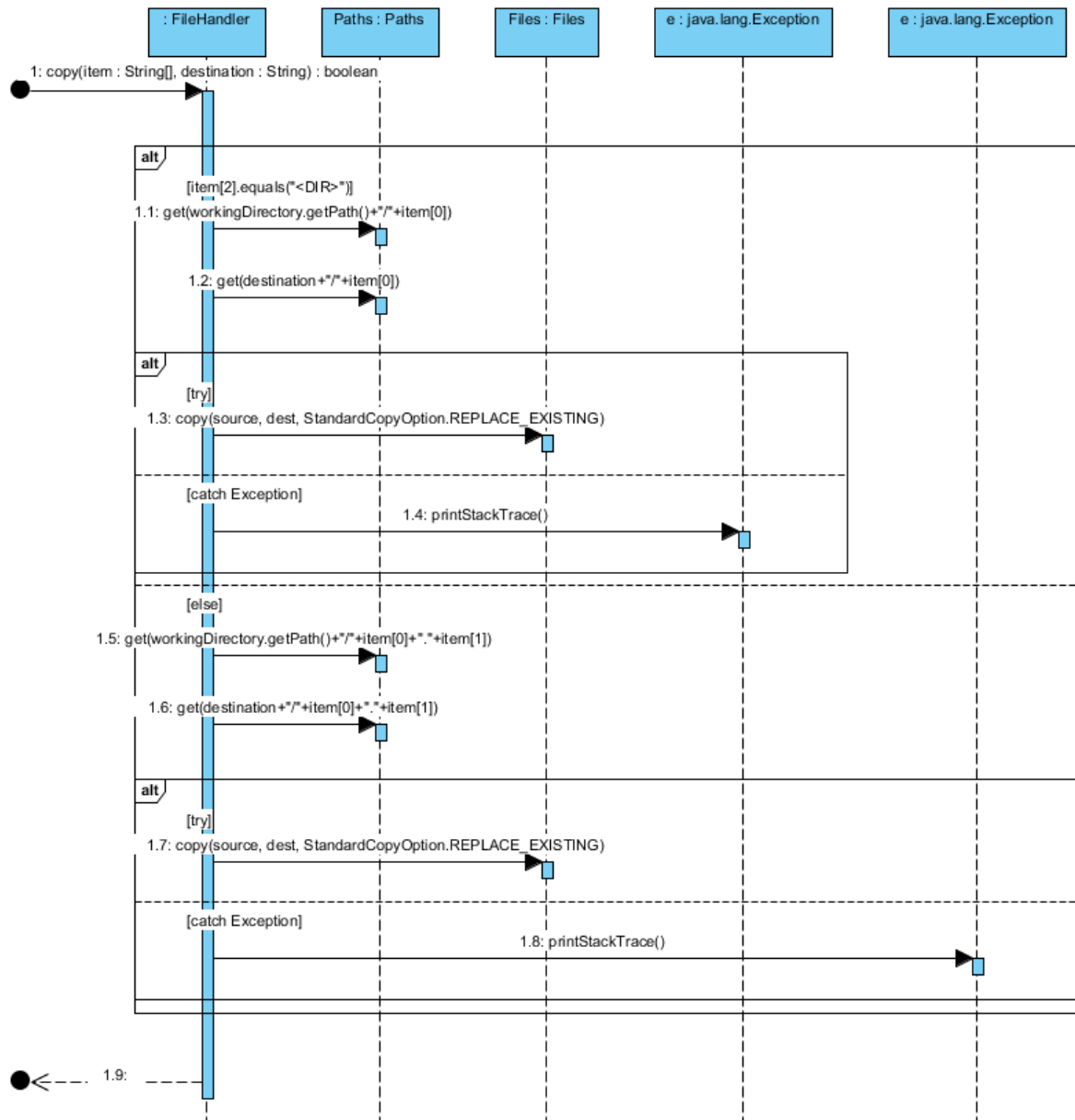
Change drive



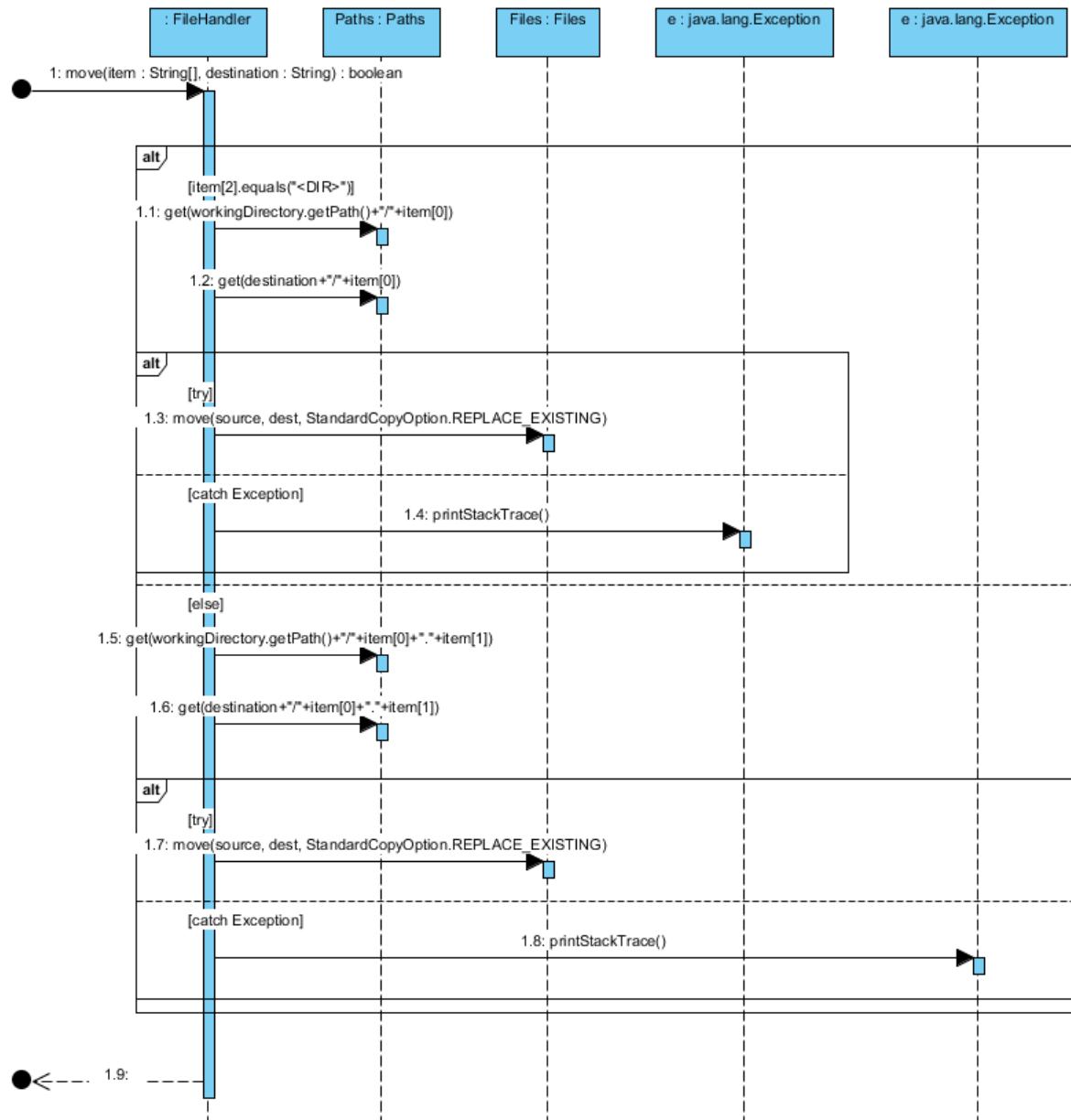
View file



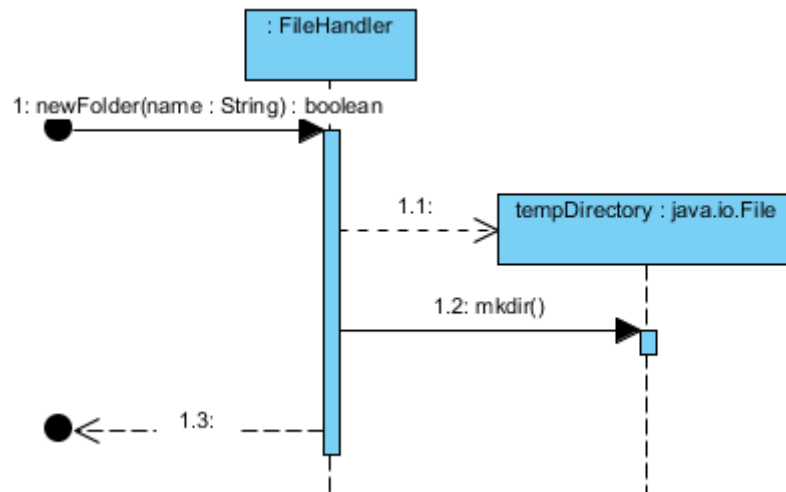
Copy selected



Move selected



New folder



Delete selected

