

Java IO labor

Készítette: Goldschmidt Balázs, BME IIT, 2015.

A. A fájlrendszer elérése

A *java.io.File* osztály felhasználásával írjon a fájlrendszert bejárni képes Java alkalmazást, amely a standard bemenetről fogadja a felhasználói parancsokat!

A File osztály metódusai elérhetők az alábbi URL-en:

<http://docs.oracle.com/javase/8/docs/api/java/io/File.html>

1. Parancsok fogadása

Az alkalmazás sorokat olvas, majd a sorokat a whitespace-ek mentén stringekké töri (*String.split*). A kapott tömb első stringje a parancs, a többi a parancs argumentuma.

A *split* metódus paramétere legyen egy szóközt tartalmazó *String* (" ").

2. Parancsfeldolgozás

Minden parancshoz egy saját függvényt kell implementálni. A függvények fejléce a következő mintát kövesse (ahol a "fun" helyett az adott parancs neve áll):

```
protected void fun(String[] cmd)
```

A parancsok feldolgozása a következő módon történjen. Készíteni kell egy *if-else if...* sorozatot, ahol az egyes *if*-ek azt ellenőrzik, hogy a parancs (az előző pontban előálló tömb első eleme) egy adott parancssal egyeznek-e. Ha igen, akkor meghívja a parancsot megvalósító függvényt. Pl:

```
if (cmd[0].equals("ls")) {  
    ls(cmd);  
}
```

3. Fájlrendszer bejárása

A fájlrendszerben való mozgáshoz szükség lesz egy *File* típusú attribútumra (*wd*, *working directory*), amely tárolja, hogy éppen melyik az aktuális munkakönyvtár (*directory*, *mappa*, *folder*).

Az alkalmazás indulásakor a kezdő könyvtár nevét a "user.dir" rendszerbeállításból vegyük (*System.getProperty()*).

Amikor könyvtárat váltunk, ezt az értéket kell felülírunk. Amikor egy adott nevű fájl el akarunk érni, ebből a könyvtárból indulunk ki. Pl.:

```
File f = new File(wd, filename);
```

4. Parancsok

Az egyes megvalósítandó parancsok a következők:

- *exit*: kilép a programból.
- *recList*: rekurzívan kilistázza a könyvtár tartalmát (lásd 2. Java diasor 44. dia).

- `pwd`: kiírja az aktuális könyvtár elérési útját (*getCanonicalPath()*)
- `cd <dir>`: az aktuális könyvtárból átlép a benne levő, `<dir>` nevű alkönyvtárba. Ha `<dir>` értéke `".."`, akkor egy szinttel feljebb lép (*getParentFile()*). Ha a `<dir>` nem létező könyvtár, akkor írjon ki hibaüzenetet!
- `ls`: kilistázza az aktuális könyvtárban levő fájlok és könyvtárak neveit.
lehetséges paraméterek:
 - `-l`: mint a fenti, de a listában megjeleníti a fájlok méretét és típusát (d - könyvtár, f - sima fájl)
- `rm <file>`: törli a `<file>` nevű fájlt.
Ha probléma merül fel, akkor adjon hibajelzést.
- `mkdir <dir>`: létrehozza az aktuális könyvtárban a `<dir>` nevű könyvtárat (*mkdir()*)
Ha `<dir>` már létezik, írjon ki hibaüzenetet!
- `cp <file1> <file2>`: `<file1>`-et átmásolja `<file2>`-be. Használja a *FileInputStream* és *FileOutputStream* osztályokat, és bájtanként másolja át a tartalmat. Ha ideje engedi, próbáljon blokkos másolást¹.
Ha a fájl nem létezik, adjon hibajelzést!
- `head -n <n> <file>`: kiírja a `<file>` nevű fájl első `<n>` sorát. Ha az opcionális `-n` paraméter hiányzik, `<n>` értéke legyen 10. Használjon *FileReadert* és *BufferedReadert*!
Ha a fájl nem létezik, adjon hibajelzést!
- `mv <file1> <file2>`: `<file1>` fájlt átnevezi `<file2>`-re.
Ha hiba történt, jelezze!
- `cat <file>`: kiírja a `<file>` nevű fájl tartalmát soronként a szabványos kimenetre.
Használjon *FileReadert* és *BufferedReadert*!
Ha a fájl nem létezik, adjon hibajelzést!
- `wc <file>`: kiírja a `<file>` nevű fájl statisztikai adatait: sorok száma, szavak száma, betűk száma. Használjon *FileReadert*, *BufferedReadert* és *StringTokenizert*!
Ha a fájl nem létezik, adjon hibajelzést!
- `length <file>`: kiírja a `<file>` nevű fájl hosszát.
Ha a fájl nem létezik, adjon hibajelzést!
- `tail -n <n> <file>`: kiírja a `<file>` nevű fájl utolsó `<n>` sorát. Ha az opcionális `-n` paraméter hiányzik, `<n>` értéke legyen 10. Használjon *FileReadert*, *BufferedReadert* és a *List* interfészt megvalósító *LinkedListet*!
Ha a fájl nem létezik, adjon hibajelzést!
- `grep <pattern> <file>`: kiírja a `<file>` nevű fájl tartalmából a `<pattern>`-re illeszkedő sorokat. Használjon *FileReadert* és *BufferedReadert*, valamint a *String.matches* metódust!
Ha a fájl nem létezik, adjon hibajelzést!

¹ Blokkos másolás esetén egyszerre nem csak egy-egy, hanem nagyobb mennyiségű bájtot másolunk (pl. 1024-et), egészen addig, amíg van beolvasható bájt.

A `System.setProperty()` metódus használata tilos. Ne felejtse el bezárni a megnyitott fájlokat!

Alternatív megoldás: objektum-orientált parancsok

Az egyes parancsokat külön-külön osztályokban is implementálhatja, ebben az esetben az osztályok valósítsák meg az alábbi interfészt. Az `execute` függvény visszatérési értéke a parancs végrehajtása után előálló aktuális könyvtár.

```
public interface Command {  
    File execute(File wd, String[] cmd);  
}
```

B. Sorszűrő alkalmazás

5. Egyszerű sorszűrő

Készítsen sorszűrő Java alkalmazást!

Az alkalmazás a standard bementről olvas sorokat, és a standard kimenetre kiírja azokat, amelyek egy adott szövegmintának megfelelnek (`String.matches()`). A mintát az első parancssori opcióként vegye át!

6. Parancssori opciók

Írja át az 5. feladat alkalmazását úgy, hogy parancssori opcióként megadott fájlokra is működjön! A programnak három opciója legyen: `-p <minta>`, `-i <file1>` és `-o <file2>`, amiket tetszőleges sorrendben meg lehet adni. Az opciók feldolgozáshoz alkalmazhatjuk a következő programrészletet:

```
String input = null;  
String output = null;  
String pattern = "";  
for (int i = 0; i < args.length; i++) {  
    if ((i+1 < args.length) && args[i].equals("-i")) {  
        i++;  
        input = args[i];  
    } else if ((i+1 < args.length) && args[i].equals("-o")) {  
        i++;  
        output = args[i];  
    } else if ((i+1 < args.length) && args[i].equals("-p")) {  
        i++;  
        pattern = args[i];  
    }  
}
```

7. Tömörített fájlok

Bővítse ki a fenti alkalmazást úgy, hogy ha `-gi` vagy `-go` opciót is kap, akkor gzip tömörítéssel tömörített fájlból olvas illetve ír (`java.util.zip` csomagban `GZIPInputStream` és `GZIPOutputStream`).

Pl.: `-i hello.txt -p java -o bello.txt.gz -go` opciók esetén a tömörítetlen `hello.txt`-ből olvas, a "java" tartalmú sorokat írja gzip tömörítéssel a `bello.txt.gz` fájlba.