Cerință

Pentru această temă, vă propunem implementarea unui shell simplificat care să permită execuția unor comenzi de modificare/interogare asupra unui sistem de fișiere.

Sistemul de fișiere trebuie imaginat sub forma unei structuri arborescente, cu noduri de tip fișier sau director, fiecare nod director putând avea 0 sau mai mulți fii. Nodul rădăcină are calea "/". Recomandăm folosirea design pattern-ului **Composite** pentru modelarea acestei structuri ierarhice.

Comenzile pe care va trebui sa le suporte shell-ul vostru sunt următoarele:

- **ls** [<*path*>]

  Listează fișierele și directoarele din folderul <*path*> sau din folderul curent dacă nu este dat niciun folder ca argument.

  E1. Dacă <*path*> nu reprezintă un folder valid, se va afișa eroarea "ls: <*path*>: No such directory".

  Dacă este urmată de argumentul -R, comanda listează întreg subarborele cu rădăcina în directorul curent (sau folderul <*path*>, după caz). În acest caz, se va realiza o parcurgere depth-first a subarborelui, alegând la fiecare pas nodurile în ordine lexicografică.

  Format de afișare:

  <*cale_absolută_folder*>:
  fișiere și foldere în ordine lexicografică (pe aceeasi linie)
  linie goală

- **pwd**

  Afișează calea absolută (pornind de la folderul "/") a folderului curent.

- **cd** <*path*>
  Setează folderul curent la <*path*>.
  E1. Dacă acest folder nu există se va afișa eroarea “cd: <*path*>: No such directory”.

- **cp** <*source*> <*dest_folder*>
  Copiază fișierul/folderul dat prin calea <*source*> în folderul <*dest_folder*>.
  Notă: Dacă <*source*> reprezintă un folder, atunci se va realiza o copiere recursivă (se va copia și conținutul folderului <*source*>). Se garantează că al 2-lea argument va fi întotdeauna un folder.
  E1. Dacă path-ul <*source*> nu există, se va afișa eroarea “cp: cannot copy <*source*>: No such file or directory”.
  E2. Dacă path-ul <*dest_folder*> nu există, se va afișa eroarea “cp: cannot copy into <*dest_folder*>: No such directory”.
  E3. Dacă folderul destinație conține deja un fișier sau folder cu numele fișierului/ folderului ce se dorește copiat, se va afișa eroarea “cp: cannot copy <*source*>: Node exists at destination”, iar copierea nu va avea loc.

- **mv** <*source*> <*dest_folder*>
  Mută fișierul/folderul dat prin calea <*source*> la calea <*dest_folder*>. Nota de la **cp** se aplică și aici.
  E1. Dacă path-ul <*source*> nu există, se va afișa eroarea “mv: cannot move <*source*>: No such file or directory”.
  E2. Dacă path-ul <*dest_folder*> nu există, se va afișa eroarea “mv: cannot move into <*dest_folder*>: No such directory”.
  E3. Dacă folderul destinație conține deja un fișier sau folder cu numele fișierului/ folderului ce se dorește mutat, se va afișa eroarea “mv: cannot move <*source*>: Node exists at destination”, iar mutarea nu va avea loc.
  E4. Dacă se încearcă mutarea unui folder al cărui subarbore include *current working directory-ul,* noul *current working dir* se va “muta” și el, păstrându-și poziția relativă față de folderul care se mută.

- **rm** <*path*>
  Șterge fișierul/folderul (cu tot cu conținutul său) de la locația <*path*>.
  E1. Dacă fișierul/folderul nu există se va afișa eroarea “rm: cannot remove <*path*>: No such file or directory”. Se garantează că <*path*> nu va fi egal cu “/”.
  E2. Dacă se încearcă ștergerea unui folder al cărui subarbore include *current working directory-ul*, comanda nu va avea niciun efect.

- **touch** <*file_path*>
  Creează un nou fișier având calea <*file_path*>.

E1. Dacă folderul în care se dorește crearea fișierului nu există, se va afișa eroarea: "touch: <*parent_path*>: No such directory", unde <*parent_path*> se obține din <*file_path*> prin eliminarea token-ului ce reprezintă numele fișierului (Ex: /dir/file => /dir).

E2. Dacă un fișier/folder cu același nume există deja la calea dată de argument, se va afișa eroarea "touch: cannot create file <*file_absolute_path*>: Node exists".

- **mkdir** <*folder_path*>
  Creează un nou folder având calea <*folder_path*>.

E1. Dacă folderul în care se dorește crearea fișierului nu există, se va afișa eroarea: "mkdir: <*parent_path*>: No such directory", unde <*parent_path*> se obține din <*folder_path*> prin eliminarea token-ului ce reprezintă numele folderului ce se dorește creat.

E2. Dacă un fișier/folder cu același nume există deja la calea dată de argument, se va afișa eroarea "mkdir: cannot create directory <*folder_absolute_path*>: Node exists".

Notă: în cazul erorilor E2 de la comenzile touch și mkdir, se vor afișa căile absolute și complete (fără token-uri de forma ".", ".." sau care conțin '*' în cazul bonusului 2) ale fișierului, respectiv folderului care se doresc create.

Bonus

- **Primul bonus** constă în implementarea comenzii **grep**:
  grep "<*regex*>"

Ea va fi folosită în conjuncție cu **ls** printr-un pipe "|". Aceasta va limita conținutul folderelor listate de **ls** (sau **ls -R**) doar la acele elemente care fac match pe expresia regulată <*regex*>. Exemplu: ls / | grep "[a-z]*". **Indicație**: puteți folosi suportul pentru expresii regulate din Java.

- Deoarece pentru mulți studenți path-urile simple pot părea plictisitoare (duc de fiecare dată către un nod bine determinat din sistemul de fișiere), pentru **al doilea bonus** veți avea de implementat comenzile **ls** (fără argumentul -R), **rm**, **touch** și **mkdir** aplicate pe path-uri ce pot conține '*' (reprezentând *orice șir de caractere*).

Notă1:

Fiecare token al path-ului (adică fiecare substring izolat prin '/'-uri de restul path-ului) poate conține cel mult un caracter '*'. Exemple de path-uri valide sunt următoarele: "/root/*", "/*/sth*/", "file*", "*dir/img*". Path-uri precum "/*dir*/file", "**/x" nu vor apărea în fișierele de test.

Comenzile **ls** și **rm** pot conține '*' în toate token-urile căilor primite ca argumente. Pentru **ls**, se va realiza o listare similară cu cea din cazul comenzii **ls -R**, dar afișând numai fișierele/folderele care respectă calea primită ca argument. Comanda **rm** va elimina toate fișierele/folderele care fac match pe calea primită ca argument.

Path-urile date comenzilor **touch** și **mkdir** pot conține '*' în toate token-urile cu excepția ultimului (cel care reprezintă numele fișierului/folderului ce trebuie creat). În acest caz, se va

crea câte un fișier/folder cu numele corespunzător în fiecare locație ce respectă calea primită ca argument.

Notă2: Numai în cazul în care nu există niciun fișier/folder pe care comanda să se poată executa (adică nu fac match pe path-ul dat ca argument), se va afișa o eroare în acest sens (cu același mesaj ca în cazul fără '*' al comenzii).

Erorile E2 din cazul comenzilor **touch** si **mkdir** se vor afișa pentru fiecare path care face match cu cel dat ca argument și în care apare eroarea respectivă (Ordinea afișării trebuie să respecte ordinea parcurgerii din cadrul comenzii ls -R).

Observații

- ➔ Tema va fi rulată folosind comanda:
  java Main <*input_file_name*> <*output_file_name*> <*errors_file_name*>

- ➔ Comenzile vor trebui încapsulate și executate peste sistemul de fișiere folosind **Command** pattern. Se vor folosi pattern-urile **Singleton** si **Factory** pentru organizarea codului ce ține de crearea comenzilor.
- ➔ Pentru toate comenzile, căile pot fi absolute (pornind de la folderul "/", ex.: /dir/file.txt) sau relative (Ex.: file.txt, ./file.txt, ../file.txt). În path-urile date ca input pot apărea notațiile pentru folderul curent (".") sau folderul părinte (".."). Nu vor apărea notațiile "-" ("folderul anterior" de la comanda cd) sau '~'. Path-urile (atât cele de la input, cât și cele din output) nu vor conține '/' la final (excepție face path-ul "/", adică rădăcină).
- ➔ Pentru comenzile **cp** și **mv**, în cazul în care <*source*> este un folder, se garantează că folderul <*dest_folder*> nu va face parte din subarborele determinat de <*source*>.
- ➔ Dacă mai multe erori sunt aplicabile pentru o comandă simplă (fără '*'), se va afișa **prima** aplicabilă în ordinea din enunț (Ei cu i minim). **Obs**: Vezi și nota 2 de la bonusul 2.

Exemple

Pentru a realiza o separație a output-urilor comenzilor succesive, formatul de afișare va fi următorul:

...
index comandă
output/erori comandă
...

input	**output**	**errors**
mkdir /exemplu cd /exemplu pwd cd / mkdir /exemplu2	1 2 3 /exemplu 4	1 2 3 4 5

mkdir /exemplu/text mkdir /exemplu3/text ls ls -R	5 6 7 8 /: /exemplu /exemplu2 9 /: /exemplu /exemplu2 /exemplu: /exemplu/text /exemplu/text: /exemplu2:	6 7 mkdir: /exemplu3: No such directory 8 9

Explicație:
Prima comandă creează un folder, operație care nu implică un output și care se execută cu succes (deci nicio eroare). Cu toate acestea se afișează “1” în fiecare fișier indicând prima comandă.
A doua comandă este asemănătoare cu prima din punct de vedere al output-ului și al erorilor.
A treia comandă “pwd” afișează calea absolută curentă, drept urmare în fișierul de output se afișează “3” reprezentând indicele comenzii. Apoi, pe linia următoare, în fișierul de output se afișează rezultatul comenzii “pwd”. În fișierul de erori nu se afișează nimic în afară de indicele comenzii.
Comenzile 4, 5, 6 se execută cu succes și fără output.
Comanda 7 “mkdir /exemplu3/text” încearcă să creeze un folder la un path invalid, deci nu va avea niciun output, dar va avea text la a 7-a comandă din fișierul de erori sugerând problema cum este specificat în enunț.
Comanda 8 execută un “ls” în root și aceasta afișează în fișierul de output următoarele:
8 - indexul comenzii
/: - path-ul de unde se execută ls-ul + caracterul “:”
/exemplu /exemplu2 - conținutul path-ului **ordonat** în ordine lexicografică
\n (aka o linie goala) - conform cu specificațiile din enunț
Comanda 9 execută “ls” recursiv din root și se afișează:
9 - indexul comenzii
/: - primul ls este din root; la sfârșit se pun “:”
/exemplu /exemplu2 - conținutul root-ului ordonat lexicografic
\n (aka linie noua) - s-a “terminat” un ls, ls-ul din root
/exemplu: - se ia primul folder disponibil din ls-ul precedent și anume /exemplu și se execută

ls pe acesta
/exemplu/text - continutul folderului /exemplu cu path-uri absolute
\n (aka linie noua) - s-a terminat încă un ls
/exemplu/text: - se afișează conținutul acestui folder deoarece parcurgerea este **in-depth** (de aceea nu se intoarce la /exemplu2)
\n (aka linie noua) - se afișează conținutul directorului /exemplu/text care este gol
\n (aka linie noua) - s-a terminat încă un ls
/exemplu2: - ne întoarcem din dfs și explorăm(facem ls) alt nod, în cazul de față /exemplu2
\n (aka linie noua) - se afișează conținutul directorului /exemplu2 care este gol
\n (aka linie noua) - s-a terminat ls-ul lui /exemplu2

mkdir bonus mkdir bonus1 touch b0nusul_1 ls . | grep "b[0|[a-z]].*1" touch bonus/1fisier1 touch bonus/2file2 ls -R / | grep "[0-9]+.*"	1 2 3 4 /: /b0nusul_1 /bonus1 5 6 7 /: /bonus: /bonus/1fisier1 /bonus/2file2 /bonus1:	1 2 3 4 5 6 7

Primul grep va afișa numai elementele din / care fac match pe regex.
Al 2-lea grep va afișa pentru fiecare folder din subarborele / (parcurse în aceeași ordine ca la un ls -R normal) elementele care fac match pe regex.

mkdir ./././seasons cd seasons touch spring touch summer touch autumn touch winter cd .. ls seasons rm seas*/* ls -R / mkdir a	1 2 3 4 5 6 7 8 /seasons: /seasons/autumn /seasons/spring	1 2 3 4 5 6 7 8 9 10 11

mkdir b touch a/same_file touch */same_file mkdir /*/folder ls /*	/seasons/summer /seasons/winter 9 10 /: /seasons /seasons: 11 12 13 14 15 16 /a: /a/folder /a/same_file /b: /b/folder /b/same_file /seasons: /seasons/folder /seasons/same_file	12 13 14 touch: cannot create file /a/same_file: Node exists 15 16
Comanda 9 șterge din orice folder care începe cu “seas” toate elementele. Comanda 14 creează în fiecare folder din / un fișier “same_file”. Comanda 15 creează în fiecare folder din / un folder “folder”. Comanda 16 face ls pe toate folderele din /.		

Trimitere și notare

Tema va fi încărcată pe vmchecker sub forma unei arhive .zip ce va conține:

- fișierele sursă .java
- fișier Makefile (cu reguli de *build* - care creează fișierele .class în folderul cu Makefile-ul - și *clean*)
- README
- JavaDoc

Punctajul va fi acordat în felul următor:

- **90** de puncte pentru implementarea comenzilor de bază (mkdir, touch, ls, rm, mv, cp, cd, pwd) și a erorilor care pot apărea
    - Există 10 teste care verifică aceste condiții fiecare valorând **9** puncte
- **20** de puncte pentru implementarea **bonusului** distribuite astfel:
    - **5** puncte **primul bonus** (verificat printr-un singur test)
    - **15** puncte pentru **al doilea bonus** (verificat printr-un singur test)
- **10** puncte pentru coding style, README și JavaDoc
- **Notă:** Este **obligatorie** folosirea design pattern-urilor prezentate în enunț pentru rezolvarea temei. Nerespectarea acestei reguli poate duce la depunctări de până la 100%.