8. domaća zadaća

U ovoj zadaći dorađujemo ljusku koju ste pripremili za prethodnu domaću zadaću. Nastavite raditi u istom projektu u kojem ste radili prošlu domaću zadaću, te nastavite koristiti pakete kakve ste tamo imali. Kad ste gotovi, ponovno ćete zapakirati kompletnu domaću zadaću u jednu ZIP arhivu koja će time sadržavati uniju svega što ste napravili za prošlu i ovu novu domaću zadaću).

Proširite sučelje Environment sljedećim metodama:

```
Path getCurrentDirectory();

void setCurrentDirectory(Path path);

Object getSharedData(String key);

void setSharedData(String key, Object value);
```

Po pokretanju programa, poziv getCurrentDirectory() treba vraćati apsolutnu normaliziranu stazu koja odgovara trenutnom direktoriju pokrenutog java procesa (tj. tražite da se “.” prebaci u apsolutnu stazu pa normalizira). setCurrentDirectory(…) omogućava da se kao trenutni direktorij koji će koristiti Vaša ljuska koristi zadani direktorij, ako isti postoji; u suprotnom pokušaj postavljanja takvog direktorija treba baciti iznimku. Implementacija sučelja Environment trenutni direktorij pamti kao jednu člansku varijablu (iz Jave ne možete doista mijenjati trenutni direktorij procesa).

Metode getSharedData(…) i setSharedData(…) omogućavaju da naredbe

dijele/pamte određene podatke. Implementacija sučelja Environment za ovo koristi mapu. getSharedData(…), ako se traži nepostojeći ključ, vraća null.

Dodajte naredbe:

pwd - bez argumenata, printa u terminal trenutni direktorij kako je zapisan u Environment (ispis mora biti apsolutna staza jer se takva i čuva u Environmentu)

cd STAZA - prima jedan argument (STAZA): novi trenutni direktorij i njega zapisuje u Environment.

Argument naredbe cd može biti relativna staza koja se razriješava s obzirom na aktualni trenutni direktorij kakav je zapisan u Environmentu. Za ovo koristite metodu resolve koju nudi razred Path.

Prođite sada kroz sve vaše naredbe koje ste implementirali u prethodnoj zadaći, i doradite ih tako da svaki argument koji je bio staza sada bude razriješen s obzirom na trenutni direktorij zapisan u Environmentu.

Dodajte naredbe:

pushd STAZA - naredba trenutni direktorij pusha na stog i potom kao trenutni direktorij postavlja onaj zadan jedinim argumentom (STAZA). Naredba u dijeljenim podatcima pod ključem “cdstack” stvara stog (ako isti već ne postoji) i tamo zapisuje trenutni direktorij prije no što ga promijeni. Ako STAZA ne predstavlja postojeći direktorij, naredba ispisuje pogrešku i ne modificira stog

niti trenutni direktorij).

popd - naredba je bez argumenata, skida sa stoga vršnu stazu i nju postavlja kao trenutni direktorij (ako takav postoji - primjerice, moguće da je u međuvremenu obrisan; u tom slučaju staza se ipak miče sa stoga ali se trenutni direktorij ne mijenja). Ako je stog prazan, naredba javlja pogrešku.

listd - naredba ispisuje u terminal sve staze koje su na stogu počev od one koja je posljednje dodana; pod “ispisuje” podrazumijeva se ispis same staze, ne sadržaj direktorija ili slično. Ako su na stogu tri staze, ispis će imati tri retka. Ako je stog prazan, ispisuje se “Nema pohranjenih direktorija.”.

dropd - naredba sa stoga skida vršni direktorij (i odbacuje ga); trenutni direktorij se ne mijenja. Ako je stog prazan, naredba javlja pogrešku.

Dodajte naredbe:

rmtree STAZA - staza (razriješena uz trenutni direktorij) mora biti postojeći direktorij; naredba briše njega i njegov kompletan sadržaj (budite JAKO JAKO JAKO oprezni pri testiranju ove naredbe da ne ostanete bez vlastite domaće zadaće).

cptree STAZA1 STAZA2 - naredba kopira stablo zadano stazom 1. Neka je STAZA1 oblika nesto/ime1 i unutra postoji primjerice datoteka1.txt, a staza 2 oblika nestodrugo/ime2/ime3. Ako nestodrugo/ime2/ime3 postoji na disku, onda se u njemu stvara poddirektorij ime1 i dalje kopira sadržaj, pa će tako nastati nestodrugo/ime2/ime3/ime1/datoteka1.txt. Ako nestodrugo/ime2/ime3 ne postoji ali postoji nestodrugo/ime2, intrpretacija je da se direktorij ime1 želi iskopirati

pod imenom ime3 pa se tako kopira, odnosno nastat će nestodrugo/ime2/ime3/datoteka1.txt. Ako ne postoji niti nestodrugo/ime2/ime3 niti nestodrugo/ime2, naredba javlja grešku i ništa ne kopira. Zadane staze početno se razriješavaju s obzirom na trenutni direktorij.

Dodajte naredbu:

massrename DIR1 DIR2 CMD MASKA ostalo

Naredba služi masovnom preimenovanju/premještanju datoteka (ne direktorija!) koji su izravno u direktoriju DIR1. Datoteke će biti premještene u DIR2 (koji može biti isti kao i DIR1). MASKA je regularni izraz napisan u skladu sa sintaksom podržanom razredom Pattern (https://docs.oracle.com/javase/9/docs/api/java/util/regex/Pattern.html) a koji selektira datoteke iz DIR1 na koje će se uopće primijeniti postupak preimenovanja/premještanja. Prilikom uporabe regularnih izraza uvijek treba raditi uz postavljene zastavice UNICODE_CASE i CASE_INSENSITIVE.

Kako je ovaj zadatak dosta “opasan”, naredba podržava nekoliko podnaredbi određenih s CMD. Za potrebe ilustracije, neka je u direktoriju DIR1 smješteno sljedeće:

slika1-zagreb.jpg

slika2-zagreb.jpg

slika3-zagreb.jpg

slika4-zagreb.jpg

slika1-zadar.jpg

slika2-zadar.jpg

slika3-zadar.jpg

slika4-zadar.jpg

ljeto-2018-slika1.jpg

ljeto-2018-slika2.jpg

ljeto-2018-slika3.jpg

ljeto-2018-slika4.jpg

Ako je CMD jednak filter, naredba treba ispisati imena datoteka koje su selektirane maskom. Primjerice:

massrename DIR1 DIR2 filter “slika\d+-[^.]+.jpg”

slika1-zagreb.jpg

slika2-zagreb.jpg

slika3-zagreb.jpg

slika4-zagreb.jpg

slika1-zadar.jpg

slika2-zadar.jpg

slika3-zadar.jpg

slika4-zadar.jpg

(pri čemu redoslijed u ispisu nije bitan i može ovisiti o načinu na koji dohvaćate datoteke direktorija DIR1. Primijetite, MASKU ili pišemo pod navodnicima (pa joj pripada sve do sljedećih navodnika, i nema nikakvih escapeova za našu ljusku-sve što unutra piše doista se predaje patternu), ili je pišemo bez navodnika pa joj pripada sve do prvog razmaka/taba/… (i opet nema escapeova za našu ljusku). S obzirom na dani primjer, potpuno jednako ponašanje ćemo dobiti i s:

massrename DIR1 DIR2 filter slika\d+-[^.]+.jpg

slika1-zagreb.jpg

slika2-zagreb.jpg

slika3-zagreb.jpg

slika4-zagreb.jpg

slika1-zadar.jpg

slika2-zadar.jpg

slika3-zadar.jpg

slika4-zadar.jpg

Za izoliranje dijelova imena koristit ćemo mogućnost grupiranja koju nudi Pattern. Podnaredba groups treba ispisati sve grupe za sve selektirane datoteke:

massrename DIR1 DIR2 groups slika(\d+)-([^.]+).jpg

slika1-zagreb.jpg 0: slika1-zagreb.jpg 1: 1 2: zagreb

slika2-zagreb.jpg 0: slika2-zagreb.jpg 1: 2 2: zagreb

slika3-zagreb.jpg 0: slika3-zagreb.jpg 1: 3 2: zagreb

slika4-zagreb.jpg 0: slika4-zagreb.jpg 1: 4 2: zagreb

slika1-zadar.jpg 0: slika1-zadar.jpg 1: 1 2: zadar

slika2-zadar.jpg 0: slika2-zadar.jpg 1: 2 2: zadar

slika3-zadar.jpg 0: slika3-zadar.jpg 1: 3 2: zadar

slika4-zadar.jpg 0: slika4-zadar.jpg 1: 4 2: zadar

Kako u maski imamo dvije zagrade - definirane su dvije grupe (grupa 1 i grupa 2) te implicitna grupa 0; stoga iza imena svake datoteke imamo za grupe 0, 1 i 2 prikazano na što su se mapirale.

Ako je podnaredba show, tada naredba prima još jedan argument: IZRAZ koji definira kako se generira novo ime. Naredba ispisuje selektirana imena i nova imena. Primjer je dan u nastavku.

massrename DIR1 DIR2 show slika(\d+)-([^.]+).jpg gradovi-${2}-${1,03}.jpg

slika1-zagreb.jpg => gradovi-zagreb-001.jpg

slika2-zagreb.jpg => gradovi-zagreb-002.jpg

slika3-zagreb.jpg => gradovi-zagreb-003.jpg

slika4-zagreb.jpg => gradovi-zagreb-004.jpg

slika1-zadar.jpg => gradovi-zadar-001.jpg

slika2-zadar.jpg => gradovi-zadar-002.jpg

slika3-zadar.jpg => gradovi-zadar-003.jpg

slika4-zadar.jpg => gradovi-zadar-004.jpg

IZRAZ može biti ili nešto pod navodnicima (pa se mogu pojavljivati praznine) ili kompaktan niz znakova (do prvog razmaka/taba/…). Izraz može sadržavati supstitucijske naredbe koje su oblika ${brojGrupe} ili ${brojGrupe,dodatnoPojašnjenje}. Ako je supstitucijska naredba oblika ${brojGrupe}, ona "sebe" zamijenjuje nizom koji je mapiran na zadanu grupu. Prilikom parsiranja izraza, obratite pažnju da ovo mora biti cijeli nenegativan broj (više od toga u trenutku parsiranja nećemo znati) pa ako nešto ne štima, javite pogrešku. Ako je supstitucijska naredba oblika ${brojGrupe,dodatnoPojašnjenje}, tada dodatno pojašnjenje mora biti broj ili nula broj (pri čemu broj može biti višeznamenkasti). Sam broj određuje koliko će minimalno znakova biti "emitirano" prilikom zapisivanja tražene grupe; npr. "${1,3}" bi značilo da se zapiše grupa 1, minimalno na tri znaka širine; ako je grupa 1 dulja od toga, zapisuje se čitava; ako je kraća, naprije se ispisuje potreban broj praznina (SPACE) a potom grupa, tako da je ukupan broj znakova tada jednak 3. "${1,03}" definira da se umjesto praznina nadopune rade znakom 0.

Konačno, podnaredba execute će napraviti zadano preimenovanje/premještanje. Koristite Files#move za provedbu.

massrename DIR1 DIR2 execute slika(\d+)-([^.]+).jpg gradovi-${2}-${1,03}.jpg

DIR1/slika1-zagreb.jpg => DIR2/gradovi-zagreb-001.jpg

DIR1/slika2-zagreb.jpg => DIR2/gradovi-zagreb-002.jpg

DIR1/slika3-zagreb.jpg => DIR2/gradovi-zagreb-003.jpg

DIR1/slika4-zagreb.jpg => DIR2/gradovi-zagreb-004.jpg

DIR1/slika1-zadar.jpg => DIR2/gradovi-zadar-001.jpg

DIR1/slika2-zadar.jpg => DIR2/gradovi-zadar-002.jpg

DIR1/slika3-zadar.jpg => DIR2/gradovi-zadar-003.jpg

DIR1/slika4-zadar.jpg => DIR2/gradovi-zadar-004.jpg

Naredba massrename svakim pokretanjem obavlja sve relevantne korake ispočetka te nigdje ništa ne pamti. Tako podnaredba groups najprije obavlja filtriranje, a potom za sve selektirane datoteke ispisuje mapirane grupe.

Implementacijski naputak. Za izvedbu generiranja imena definirajte sučelja:

NameBuilderInfo

StringBuilder getStringBuilder()

String getGroup(int index)

NameBuilder

    void execute(NameBuilderInfo info)

Objekti tipa NameBuilder generiraju dijelove imena zapisivanjem u StringBuilder koji dobiju preko argumenta info u metodi execute. Napravite razred NameBuilderParser koji kroz konstruktor dobiva IZRAZ, parsira ga i vraća jedan NameBuilder objekt:

NameBuilderParser:

    public NameBuilderParser(String izraz);

    public NameBuilder getNameBuilder();

    private … vaše ostale potrebne metode …

Pogledajmo primjer:

NameBuilderParser parser = new NameBuilderParser(“gradovi-${2}-${1,03}.jpg”);

NameBuilder builder = parser.getNameBuilder();

Parser će na temelju predanog izraza napraviti:

- jedan objekt tipa NameBuilder koji će u metodi execute u stringbuilder zapisati “gradovi-”

- jedan objekt tipa NameBuilder koji će u metodi execute u stringbuilder zapisati

na što god je postavljena grupa 2

- jedan objekt tipa NameBuilder koji će u metodi execute u stringbuilder zapisati “-”

- jedan objekt tipa NameBuilder koji će u metodi execute u stringbuilder zapisati na što god je postavljena grupa 1, na minimalno tri znaka širine uz dopunu nulama

- jedan objekt tipa NameBuilder koji će u metodi execute u stringbuilder zapisati “.jpg”

- jedan objekt tipa NameBuilder koji će imati reference na ove prethodno stvorene NameBuildere i koji će u metodi execute redom nad svima njima pozvati execute

Poziv parser.getNameBuilder() će vratiti upravo referencu na ovaj posljednji objekt. Primijetite da sve u svemu imate tri različite vrste NameBuilder objekata (čitaj: tri konkretna razreda): jedan koji uvijek upisuje konstantan string koji mora primiti kroz konstruktor, jedan koji uvijek zapisuje zadanu grupu uz eventualno zadanu minimalnu širinu (podatke prima kroz konstruktor) te jedan koji kroz konstruktor prima reference na niz drugih i u svojoj execute poziva njihove execute.

Vezano za zadavanje supstitucijskih naredbi u IZRAZU (npr. ${1,03}), pravila su sljedeća. Naredba započinje s ${ (nema razmaka između; $ { ne započinje supstitucijsku naredbu). Jednom kad je naredba započela, pripada joj sve do }. Unutra mogu biti proizvoljni razmaci (ali ne između znamenaka broja); npr. ${1,03}; ${ 1 , 03 }; ${1 , 03}; ${ 1 }; ako unutar izraza nešto ne štima, parser treba baciti iznimku a naredba treba korisniku ispisati prikladnu poruku. Ne postoje nikakvi escapeovi. Primjerice, ${$1} mora pri parsiranju baciti iznimku.

Jednom kad ste ovo složili na opisani način, pseudokod postupka preimenovanja prikazan je u nastavku:

```
NameBuilderParser parser = new NameBuilderParser(IZRAZ);
NameBuilder builder = parser.getNameBuilder();
Pattern pattern = napravi za MASKA
for(Path file : DIR1) {
    Matcher matcher = pattern.matcher(file.ime);
    ako matcher ne pali, idi na sljedeću datoteku
    NameBuilderInfo info = napraviInfo(matcher);
    builder.execute(info);
    String novoIme = builder.getStringBuilder().toString();
    preimenuj/premjesti file u DIR2/novoIme
                                                    }
```

Primijetite: parser parsira izraz samo jednom i stvara “program” za izgradnju imena. Pattern se također stvara samo jednom. Za svako ime na temelju patterna stvori se i pokrene matcher te ako isti pali za ime, stvara se novi objekt NameBuilderInfo te se s njime pokreće program za izgradnju imena.

Ako se prilikom izgradnje imena dogodi pogreška, ili prilikom preimenovanja, naredba se prekida i u ljusci se ispisuje prikladna poruka pogreške.

Prilikom rješavanja zadaće vrijede sve uobičajene ograde.

Nudim termine konzultacija u srijedu, četvrtak i petak u 12h.

Kako su u tijeku međuispiti, u okviru ove zadaće ne morate napisati niti jedan junit test (iako ih toplo preporučam, posebice kod parsera za IZRAZ).

Rok za predaju domaće zadaće: subota, 5.5.2018. 08:00:00 ujutro.

U imenu ZIP arhive stavite hw08, ali svi paketi ostaju isti kao i kod prošle zadaće.