# 4. laboratorijska vježba, 1. dio

## *Zadatak 1.*

Ovaj zadatak nastavlja se na zadatak u kojem ste trebali implementirati ljusku. Prilikom rješavanja tog zadatka preuzeli ste razred `SimpleHashtable` i doradili ga. Sada na taj razred (i `TableEntry`) primijenite tehnologiju Java Generics: parametrizirajte oba razreda. Potom prođite kroz kod ljuske i naredbi gdje koristite bilo što od ovog razreda i uklonite eksplicitna ukalupljivanja.

Dodajte u razred `SimpleHashtable` još dvije metode tvornice:

```
public Iterable<...> keys() { … }
public Iterable<...> values() { … }
```

Naravno, popravite prototip kako je potrebno da bi bio ispravno parametriziran. Prva treba vraćati parametrizirani objekt koji će vraćati iteratore po pohranjenim ključevima a druga treba vraćati parametrizirani objekt koji će vraćati iteratore po pohranjenim vrijednostima. Nakon toga, ovaj kod bi morao raditi:

```
SimpleHashtable<String,Integer> exams = new SimpleHashtable<>();
exams.put("Ivana", Integer.valueOf(5));
exams.put("Janko", Integer.valueOf(4));

for(String name : exams.keys()) {
  System.out.println("Ime = " + name);
}

for(Integer grade : exams.values()) {
  System.out.println("Ocjena = " + grade);
}

for(SimpleHashtable.TableEntry<String,Integer> pair : exams) {
  System.out.println("(Ime, Ocjena) = (" + pair.getKey()+", "+pair.getValue()+")");
}
```

## *Zadatak 2.*

Ovaj zadatak nastavlja se na zadatak u kojem ste trebali implementirati ljusku (odnosno na popravljenu verziju iz zadatka 1). Potrebno je ljusci dodati naredbu `dir`. Primjeri pozivanja su sljedeći.

dir D:\windows /sort=eSdn /filter=a* /type=f

Naredba može imati do četiri argumenta (i bilo koji može biti izostavljen). Primjerice, ako nije dana staza, lista se trenutni direktorij. Ako nije dan /sort=, ispis se radi redosljedom kojim su datoteke pronađene na datotečnom sustavu (kako ih vraća API); ako nije dan filter, ispisuju se sve datoteke i direktorij.

**Oznake kod sortiranja:**
E ekstenzija, leksikografski obrnuto (od Z prema A)
e ekstenzija, leksikografski (od A prema Z)
S veličina (od većih prema manjima)
s veličina (od manjih prema većina)

D datum i vrijeme (posljednje stvoreni se ispisuju prvi; najstariji zadnji)
d datum i vrijeme (najstarije datoteke idu prve)
N ime, leksikografski obrnuto (od Z prema A)
n ime, leksikografiski (od A prema Z)
T vrsta (prvo direktoriji, potom datoteke)
t vrsta (prvo datoteke, potom direktoriji)

Ako je navedeno više oznaka (kao u primjeru), sortiranje se radi prema prvom kriteriju, pa unutar toga prema drugom (gdje prvi sort nema odluke), pa unutar toga prema trećem, itd. Program treba biti napravljen tako da se bez većih izmjena u kodu mogu dodavati i novi načini sortiranja (primjerice, prema duljini imena, broju znamenki u imenu, itd.)

Filter, ako je prisutan, ograničava ispis samo na datoteke čije je ime u skladu sa zadanom maskom (maska može sadržavati samo jednu zvjezdicu).

Konačno, tip može biti nezadan (pa se rezultati ne filtriraju po tipu), može biti "f" pa će se ispisati samo datoteke (*file*), ili može biti "d" pa će se ispisati samo direktoriji (*directory*).

Da biste riješili ovaj zadatak, trebat ćete definirati/koristiti barem jedno sučelje (po potrebi i više) i više njegovih implementacija. Razmislite dobro kako ćete oblikovati rješenje. Idealan kôd bi bio onaj koji ne ovisi o implementacijskim detaljima načina sortiranja i filtriranja, odnosno kod kojeg se novi načini mogu dodavati bez da se mijenja dio koda koji radi konkretnu obradu (glavni dio algoritma). Nacrtajte na papiru sučelja i razrede koji će Vam trebati (tj. dijagram razreda).

## *Zadatak 3.*

Radite komandno-linijski program koji analizira sadržaj direktorija (i svih njegovih poddirektorija). Direktorij se predaje kao jedini argument pri pokretanju programa. Program nazovite `Analyzer` i smjestite ga u paket `hr.fer.oop.week9.stat`. Program treba proći kroz cjelokupno stablo te temeljem pronađenih datoteka izvući sljedeće statistike.

- Za svaku različitu pronađenu ekstenziju (velika i mala slova se ne razlikuju) izbrojati
  - koliko takvih datoteka ima
  - kolika je ukupna veličina svih takvih datoteka
- Za svako početno slovo imena (velika i mala slova se ne razlikuju) izbrojati:
  - koliko takvih datoteka ima
  - kolika je ukupna veličina svih takvih datoteka
- Koliko je ukupno datoteka pronađeno
- Koliko su ukupno velike sve datoteke

Program na zaslon treba ispisati sve prikupljene statistike. Pri tome se statistike po ekstenzijama ispisuju najprije sortirano po ukupnoj veličini tih datoteka (prvo najveće) a unutar toga po ekstenzijama leksikografski. Statistike po početnim slovima ispisuju se leksikografskim poretkom.

## *Zadatak 4.*

Potrebno je izraditi jednostavan emulator baze podataka. Smjestite implementaciju u paket `hr.fer.oop.week9.db`. U repozitoriju Ferka pronaći ćete datoteku `database.txt`. Radi se o tekspovnoj datoteci koja u svakom retku sadrži zapis za jednog studenta. Atributi su: *jmbag*, *lastName*, *firstName*, *finalGrade*. Nazovite program `StudentDB`. Kada se program pokrene, čita i obrađuje u trenutnom direktoriju sadržaj datoteke `database.txt`. Napišite razred `StudentRecord`; primjerci ovog razreda predstavljaju zapise

za pojedine studente. Pretpostavite da za jednog studenta ne može postojati više zapisa. Implementirajte metode `hashCode` i `equals` tako da su dva zapisa jednaka ako su jmbagovi jednaki.

Napišite razred `StudentDatabase`: njegov konstruktor mora dobiti listu objekata tipa `String` (sadržaj iz `database.txt`). Potom mora stvoriti internu reprezentaciju liste zapisa. Također, mora stvoriti indeks koji će omogućiti sa složenošću O(1) brzi dohvat zapisa studenta ako je poznat njegov jmbag (koristite prikladnu implementaciju mape). U razred dodajte i sljedeće dvije javne metode:

```
public StudentRecord forJMBAG(String jmbag);
public List<StudentRecord> filter(IFilter filter);
```

Prva metoda koristi indeks kako bi dohvatila traženi zapis u O(1); ako zapis ne postoji, metoda vraća `null`. Druga metoda prihvaća referencu na objekt koji je primjerak razreda koji implementira sučelje `IFilter`:

```
public interface IFilter {
public boolean accepts(StudentRecord record);
}
```

Metoda `filter` u razredu `StudentDatabase` prolazi kroz sve zapise interne liste i stvara novu listu u koju dodaje samo one zapise za koje metoda `accepts` predanog objekta vrati **`true`**. Novostvorena lista vraća se kao povratna vrijednost metode.
Program korisnički ulaz dobiva preko tipkovnice. Morate podržati samo jednu naredbu: **query**, koja dolazi u dva oblika. Prvi oblik omogućava specificiranje JMBAG-a i na ekran ispisuje zapis s tim JMBAG-om. Srugi oblik omogućava da korisnik zada prezime (ili njegov dio koristeći zamjenski znak * koji se može pojaviti samo jednom ali na bilo kojem mjestu) i ispisuje sve zapise koji zadovoljavaju to prezime.

Napišite razred `LastNameFilter` koji implementira sučelje `IFilter`. Ima samo jedan javni konstruktor koji prima jedan argument: masku prezimena. Implementirajte drugi oblik upita koristeći ovaj razred. Evo primjera koji prikazuje interakciju s korisnikom i očekivano formatiranje ispisa.

Simbol za prompt koji program ispisuje je "`> `".

```
> query jmbag="0000000003"
+============+========+========+===+
| 0000000003 | Bosnić | Andrea | 4 |
+============+========+========+===+
Records selected: 1
> query lastName="B*"
+============+===========+===========+===+
| 0000000002 | Bakamović | Petra     | 3 |
| 0000000003 | Bosnić    | Andrea    | 4 |
| 0000000004 | Božić     | Marin     | 5 |
| 0000000005 | Brezović  | Jusufadis | 2 |
+============+===========+===========+===+
Records selected: 4
> query lastName="Be*"
Records selected: 0
```

Uočite da se stupci tablice trebaju automatski proširivati tako da se sav sadržaj može ispisati. Redosljed ispisa odgovara redosljedu kojim su podatci pohranjeni u bazi podataka.

Ako korisnik zada pogrešan unos, ispišite prikladnu poruku pogreške. Dopustite korisniku da unosi proizvoljan broj praznina. Primjerice, sve sljedeće bi trebalo prihvatiti kao ispravan unos:

```
query lastName="Be*"
query lastName ="Be*"
query lastName= "Be*"
query lastName = "Be*"
```

Sve osim toga je pogrešan unos. **Ne morate** raditi podršku za kompleksnije unose poput:

```
query lastName="Be*" or (lastName != "*e" and firstName = "Ivan")
```

ali možete razmisliti o tome. Kako biste pristupili izradi podrške za takav način filtriranja?