Objektno-orijentisano programiranje, ispit
Matematički fakultet, školska godina 2023/2024

Napomena: Na Desktop-u napraviti direktorijum pod imenom `oop_InicijaliAsistenta_Prezime_Ime_Alas` (npr. `oop_VK_Peric_Pera_mi22082`). Pokrenuti *Intellij Idea* i u napravljenom direktorijumu napraviti projekat sa istim nazivom. Zadatke sačuvati redom u okviru paketa `zadatak1`, `zadatak2` i `zadatak3`.
Kod **ne sme** imati sintaksnih grešaka niti izbacivanje `NullPointerException`-a.
U tekstu je dat opis klasa, njihovih atributa i metoda. **Dozvoljeno** je dodati nove atribute, klase, metode, enume, interfejse u slučaju da olakšavaju implementaciju i/ili poboljšavaju kvalitet koda i slično.
Da bi se uspešno položio ispit potrebno je osvojiti **barem 30 poena**.

1. **[20p]** `Zivotinjska farma`.

   Napisati klasu Direktiva i obezbediti sledeće metode:

   - konstruktor klase Direktiva, ako je poznato da se direktiva karakteriše imenom osobe kojoj je namenjena kao jednim podatkom tipa String, i naredba koju ta osoba treba da izvrši, takođe tipa String.
   - konstruktor kopije.
   - odgovarajuće set*() i get*() metode.
   - metod `toString()` koji vraća String-reprezentaciju direktive.

   Napisati baznu klasu Zivotinja koja pamti samo podatak o imenu životinje. Napraviti konstruktor koji prihvata ime životinje, konstruktor kopije, set*() i get*() metode. Predefinisati metod toString() tako da vraća prazan String.

   Napisati klasu Svinja koja nasleđuje klasu Zivotinja. Pored imena, ova klasa ima podatak o paroli tipa String. Napraviti konstruktor koji prihvata sve potrebne podatke, set*() i get*() metode za parolu. Napraviti konstruktor koji prihvata samo ime, a parolu postavlja na "Sve su zivotinje jednake". Predefinisati metod `toString()` tako da vraća tekst u sledećem formatu:

   "Svinja IME_SVINJE kaze: PAROLA"

   Napisati klasu Ovca koja takođe nasleđuje klasu Zivotinja. Ovca u odnosu na životinju ima dodatni niz parola koje može da izgovori. Niz parola je niz stringova koje se prosleđuju prilikom konstrukcije objekta. Napraviti konstruktor koji prihvata ime i niz parola, kao i konstruktor kopije, set*() i get*() metode. Predefinisati metod `toString()` tako da vraća tekst u sledećem formatu:

   "Ovca kaze: PAROLA"

   gde ovca ne ispisuje svoje ime, ali svaki put kad se pozove metod `toString()` ispisuje novu parolu iz sačuvanog niza parola. Parole se uzimaju redom – prvo parola[0], pa parola[1], i tako dalje. Kad se dođe do kraja niza parola, ovca kreće ponovo od prve parole. Obezbediti da se na adekvatan način broji koliko objekata klase Ovca je napravljeno u programu. Napraviti metod `uglas()` koji na standardni izlaz ispisuje trenutnu parolu onoliko puta koliko ukupno ima ovaca (dakle, istu parolu više puta).

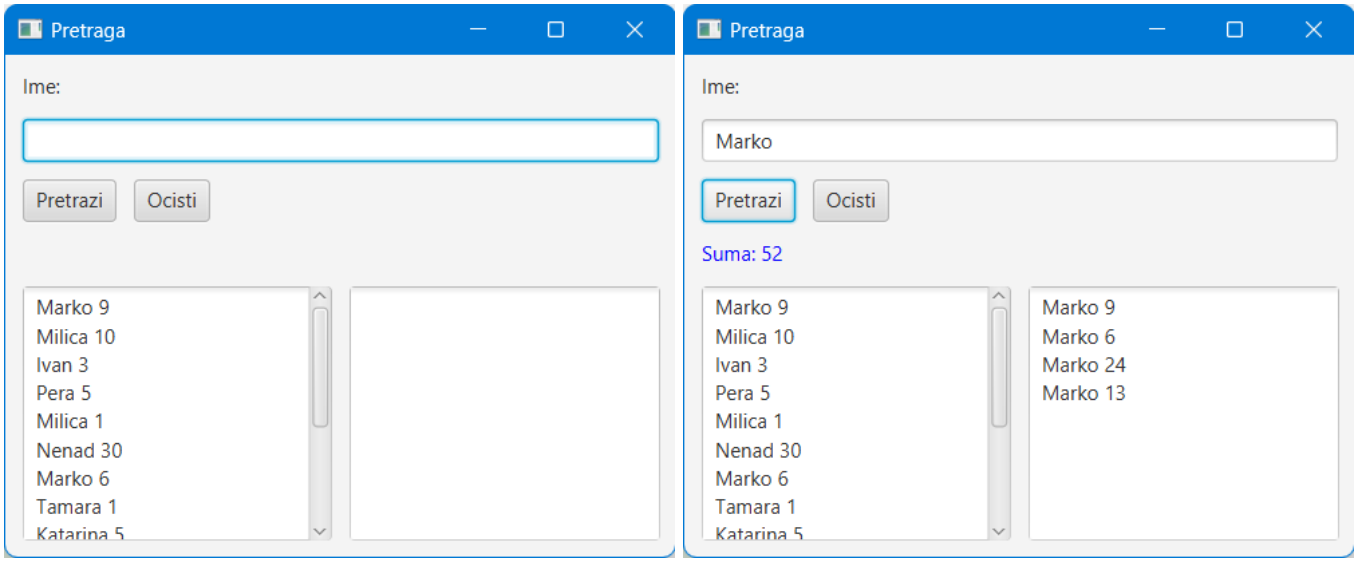
   Napisati metod `naredi(Direktiva d)` koji proverava da li ime ovce odgovara imenu koje je sačuvano u direktivi. Ako ne odgovara, metod ne radi ništa. Ako odgovara, zamenjuje svoj niz parola jednom parolom – parolom koja je definisana u polju komanda klase Direktiva.

   U test klasi *TestZivotinjskaFarma* napraviti i jedan objekat klase Svinja imena "Napoleon", i dve ovce sa imenima "Moli" i "Poli". Nizove parola za ove dve ovce i svinju učitavati sa standardnog ulaza, prvo broj rečenica, pa onda jednu po jednu rečenicu, jedna rečenica je jedan token

   (npr. Cetiri_Noge_Dobre_Dve_Noge_Lose).

   - Ispisati na standardni izlaz podatke o Napoleonu – ono što vraća njegov metod toString()
   - Pozvati metod uglas() i za Moli i za Poli
   - Narediti Poli (metodom naredi(Direktiva d)) da promeni parolu u "CETIRI NOGE DOBRE, DVE BOLJE".
   - Ispisati rezultat metoda toString() za Poli
   - Promeniti parolu Napoleonu na "Sve su zivotinje jednake, ali su neke jednakije od drugih" i ispisati ga ponovo na standardni izlaz.

2. **[20p]** Napisati aplikaciju korišćenjem JavaFX biblioteke koja izgleda kao na slikama. U korenom direktorijumu projekta nalazi se datoteka *Podaci.txt*. Sadržaj reda datoteke je ime osobe (jedna reč) kao i jedan prirodni broj. Klikom na dugme *Pretrazi* pronalaze se sva pojavljivanja unesenog imena, prikazuju se, i ispisuje se zbir brojeva. Klikom na dugme *Ocisti* tekstualni sadržaj se briše kao i tekst iz labele. Dimenzije prozora su 400x300.



3. **[25p]** Implementirati generičku klasu `MatematickiSkup`, koja predstavlja matematički skup tj. strukturu koja nema duplikate. Skup karakteriše ime, elementi koji se čuvaju u nizu i inicijalni kapacitet koji predstavlja početnu dimenziju niza elemenata. Implementirati sledeće metode:

   - `public boolean postoji(T element)` - vraća *true* ukoliko element postoji u skupu, inače *false*.
   - `public void dodaj(T element)` - dodaje element u skup. Ukoliko je broj elemenata jednak dužini niza, duplirati dužinu niza.
   - `public Optional<T> nadjiMaksimum()` - vraća opcionu vrednost od maksimuma ukoliko postoji, inače vraća praznu opcionu vrednost.
   - `public MatematickiSkup<T> unija(MatematickiSkup<T> s)` - vraća matematički skup koji predstavlja uniju tekućeg skupa i skupa s. Naziv rezultujućeg skupa je oblika: *nazivPrvogSkupa* u *nazivDrugogSkupa* (npr. $s1$ u $s2$).
   - Predefinisati metod `toString` da vraća string oblika: *imeSkupa* = { *elementiSkupaRazdvojeniZarezom* }

   Definisati klasu *Osoba* koju karakterišu ime i prezime. Osobe porediti leksikografski na osnovu imena, ukoliko su imena jednaka, porediti prezimena.

   Napisati klasu *Test* u kojoj treba testirati rad sa skupom *Osoba* na sledeći način:

   - Definisati skupove $s1$ i $s2$ i ispisati ih.
   - Odrediti uniju skupova $s1$ i $s2$ i ispisati je.
   - Odrediti maksimum u uniji. Ukoliko maksimum postoji ispisati njegovu vrednost, inače ispisati poruku da maksimum ne postoji.

   Napomena: Prilikom instanciranja generičkog niza koristiti *Object* niz ili niz ograničenja tipa T (ukoliko ograničenje postoji).