5. laboratorijska vježba

Važna napomena: u svim zadacima potrebno je napisati Javadoc komentare za svaki razred, sučelje i enumeraciju te generirati dokumentaciju. Svi nazivi razreda, metoda i varijabli i slično moraju biti na engleskom. Sav napisani programski kod mora biti napisan u skladu s konvencijama imenovanja varijabli, metoda i razreda (varijable i metode: malo početno slovo, camel-case; razredi i sučelja: veliko početno slovo, camel-case; konstante: uobičajeno sve veliko i razdvajanje podvlakom) te ostalim pozitivnim praksama (uključivo i korektno uvlačenje redaka; smisleno razdvajanje više različitih semantički grupiranih redaka praznim recima, pravilnim razmještajem otvorene i zatvorene vitičaste zagrade i slično). Za više informacija pogledajte http://www.oracle.com/technetwork/java/codeconventions-150003.pdf.

U okviru ove laboratorijske vježbe rješavaju se dva zadatka. U prvom zadatku naglasak je na uporabi Javinog okvira kolekcija a u drugom na radu s datotečnim sustavom i tokovima okteta.

Zadatak 1.

U paket `hr.fer.oop.lab5.shell` smjestite razred `MyShell`. Taj razred sadrži metodu `main` i predstavlja ljusku (slično kao Command prompt na Windowsima odnosno Bash na Linuxu) koja s korisnikom komunicira preko tipkovnice i ekrana. Jednom pokrenuta, ljuska s tipkovnice čita redak po redak i svaki redak tumači kao jednu naredbu (redak je sve što se unese do pritiska tipke ENTER). Naredba se ne može protezati na više od jednog retka.

Inicijalni radni direktorij ljuske je onaj u koji se razriješi uporabom `Paths.get(".")`. Prompt ljuske ispisuje znak dolar ($), naziv trenutnog direktorija ljuske (ali samo posljednju komponentu; ako je trenutni direktorij `C:\Windows\System32`, pisat će `System32`), znak veće (>) i jedan razmak. Potom korisnik unosi naredbu i pritiskom na ENTER ta se naredba izvršava.

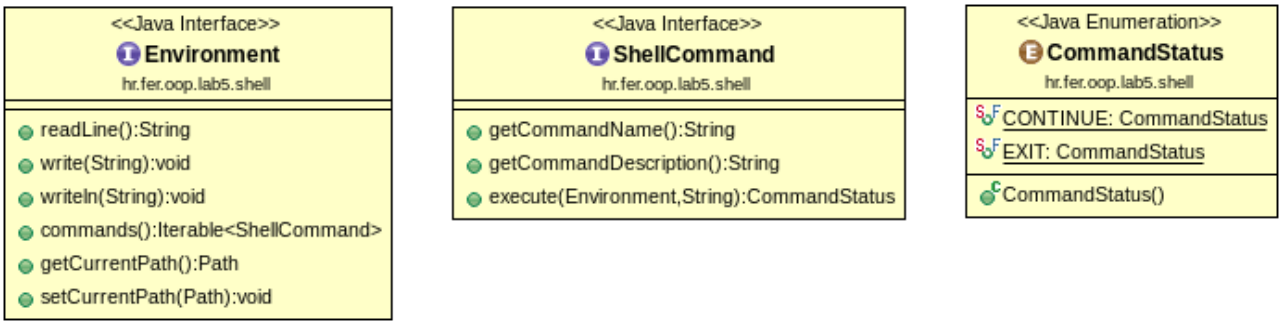
Naredba **`date`** na zaslon ispisuje trenutni datum i vrijeme (primjerice: 2015-12-25 13:24:17). Naredba **`pwd`** na zaslon ispisuje trenutni direktorij (u apsolutnom formatu tj. počevši od vršnog direktorija; na Windowsima bi to značilo stazu koja počinje oznakom diska, primjerice `C:\java\lab5`). Naredba **`cd staza`** kao trenutni direktorij postavlja direktorij zadan kao argument *staza*. To može biti i relativna staza koja se onda razrješava s obzirom na trenutnu stazu ljuske; koristite za to postojeću funkcionalnost razreda `Path`, nemojte sami pisati svoju implementaciju razrješavanja. Naredba **`help`** ispisuje sve naredbe i njihov opis dok naredba **`quit`** prekida rad ljuske. Primjer rada s ljuskom prikazan je u nastavku.

```
Welcome to MyShell! You may enter commands.
$lab5> pwd
D:\eclipse_workspaces\oop\lab5
$lab5> cd ..
Current directory is now set to D:\eclipse_workspaces\oop.
$oop> cd D:\Filmovi\StarWarsTheForceAwakens
Current directory is now set to D:\filmovi\StarWarsTheForceAwakens.
$StarWarsTheForceAwakens> date
2015-12-25 13:24:17
$StarWarsTheForceAwakens> quit
Thank you for using this shell. Goodbye!
```

Po pokretanju, ljuska ispisuje pozdravnu poruku, prikazuje prompt i čeka. Korisnik zadaje naredbu "pwd" (prikazano crveno) koja ispisuje u novom retku apsolutnu stazu trenutnog direktorija.

Korisnik potom zadaje naredbu "cd .."; uočite, staza u primjeru je relativna. Izvođenjem naredbe mijenja se trenutna staza ljuske. Korisnik potom zadaje naredbu "cd D:\Filmovi\StarWarsTheForceAwakens" kojom traži promjenu trenutne staze u zadani direktorij. Korisnik zatim zadaje naredbu "date" koja ispisuje trenutni datum i vrijeme. Konačno, korisnik zadaje naredbu "quit" kojom prekida rad s ljuskom. Iskoristite ovaj primjer kako biste istestirali rad ljuske.

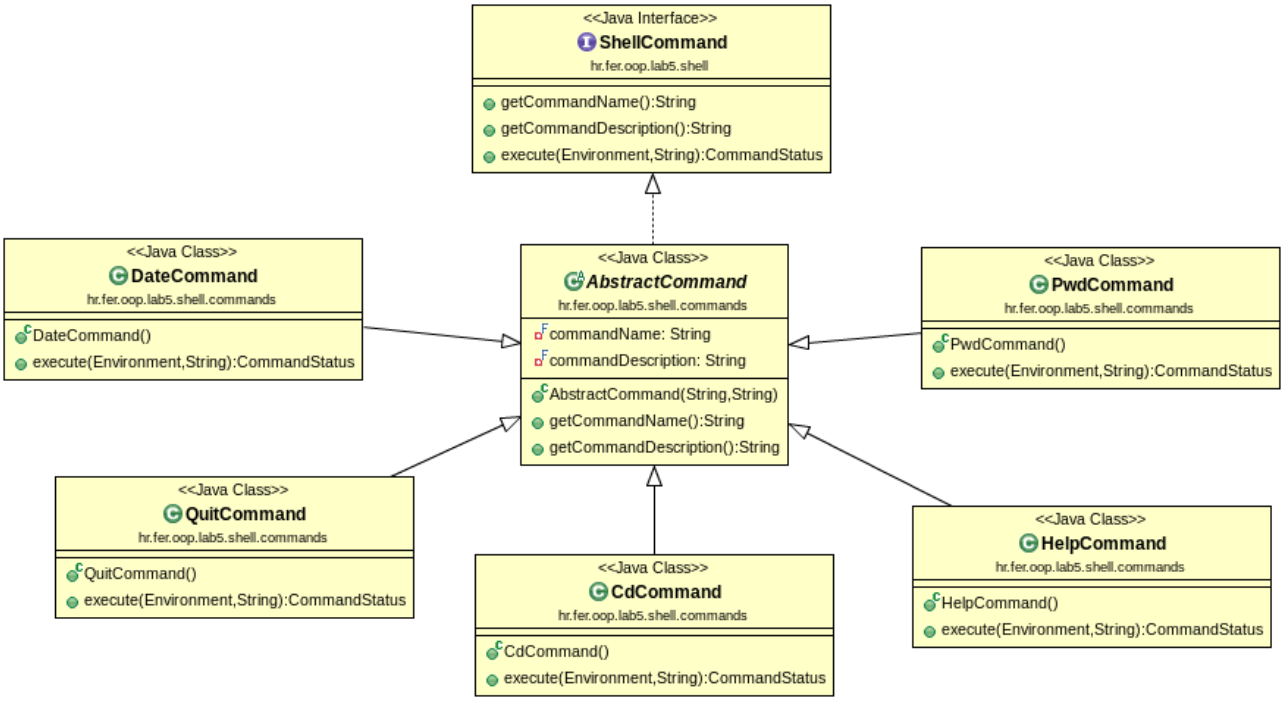
Kako su organizirani razredi/sučelja? Slika u nastavku prikazuje jedan dio razreda i sučelja.



Sučelje `Environment` modelira okruženje ljuske: nudi funkcionalnost čitanja redaka s tipkovnice, zapisivanja teksta na ekran (bez automatskog dodavanja ENTERa na kraju ili s dodavanjem) te metodu tvornicu za `Iterable<ShellCommand>` objekt preko kojeg se može dobiti iterator koji obilazi po svim implementiranim naredbama. Također nudi metode za dohvat i postavljanje trenutnog direktorija.

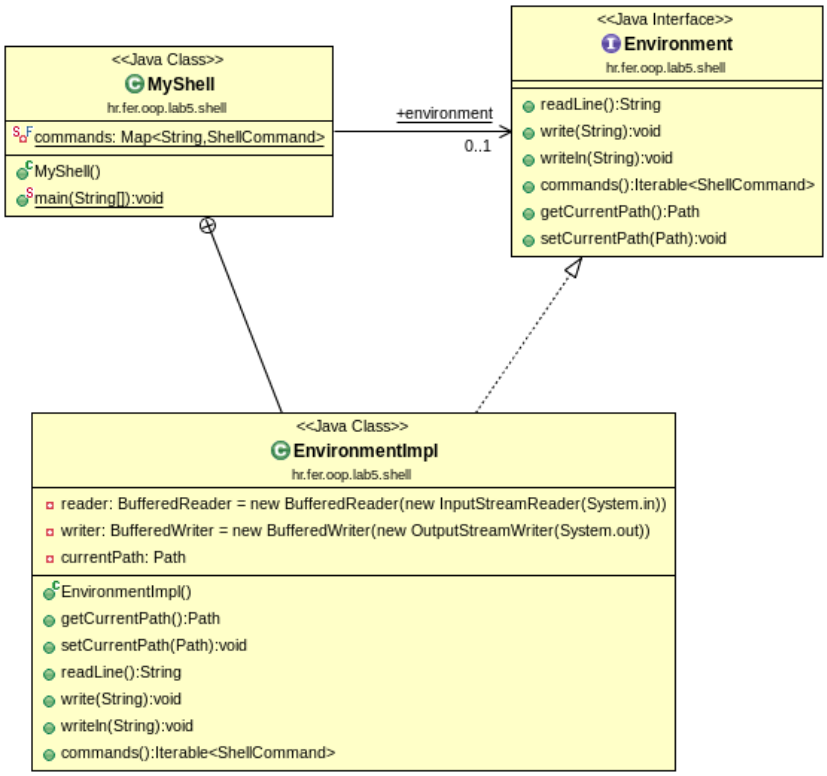
Sučelje `ShellCommand` opisuje jednu naredbu. Naredba ima ime (`cd`, `quit`, `date`, …), opis te metodu `execute` čijim se pozivom naredba izvodi. Što treba napraviti po završetku izvođenja, naredba ljusci govori putem povratne vrijednosti koja je tipa `CommandStatus` – enumeracija pri čemu `CONTINUE` znači da ljuska treba nastaviti s izvođenjem i korisnika pitati za novu naredbu, a `EXIT` znači da ljuska treba završiti s radom. Metoda `execute` kao argumente mora dobiti referencu na okruženje u kojem se naredba izvodi (taj objekt stvara i održava sama ljuska) te string koji sadrži sve što je korisnik unio kao naredbeni redak nakon uklanjanja same naredbe (primjerice, ako je korisnik zadao "`copy a.txt b.txt`", drugi argument će biti "`a.txt b.txt`"). Važno: razred koji implementira pojedinu naredbu definira pravila po kojima se ostatak retka rastavlja na argumente; primjerice, naredba `cd` bi trebala podržati zadavanje argumenta pod navodnicima ako isti sadrži prazninu kako bi ispravno bio protumačen: npr. `cd "C:\Program Files\Java"`.

Hijerarhija naredbi prikazana je na slici u nastavku.



Razred `AbstractCommand` definira mehanizam čuvanja imena i opisa tako da to ne mora svaka naredba kopirati.

Konačno, pogledajmo i samu ljusku (slika u nastavku).



Ljuska je predstavljena razredom `MyShell`, koji ima statički ugniježđeni razred `EnvironmentImpl` koji implementira sučelje `Environment`. Ljuska ima statičku tablicu raspršenog adresiranja `commands` u koju je dodan po jedan primjerak svake naredbe (ključ je naziv naredbe, vrijednost je

referenca na primjerak same naredbe). Da bi se osiguralo da korisnik može naredbe pisati i velikim i malim slovima (ili kombinacijom), neka ključevi budu imena naredbi pisana velikim slovima. Ljuska također ima statičku referencu na jedan primjerak okruženja (statička članska varijabla `environment`) koja se inicijalizira na primjerak razreda `EnvironmentImpl`. Iterator u tom razredu ima pristup statičkoj tablici koja čuva naredbe pa može izvesti traženi iterator.

Za inicijalizaciju naredbi ljuske iskoristite statički inicijalizacijski blok. Kostur ljuske prikazan je u nastavku.

```
package hr.fer.oop.lab5.shell;

...

public class MyShell {

	private static Map<String, ShellCommand> commands;

	static {
		commands = new HashMap<>();
		ShellCommand[] cc = {
			new HelpCommand(),
			new QuitCommand(),
			new CdCommand(),
			new PwdCommand(),
			new DateCommand()
		};
		for(ShellCommand c : cc) {
			commands.put(c.getCommandName(), c);
		}
	}

	public static class EnvironmentImpl implements Environment { ... }

	public static Environment environment = new EnvironmentImpl();

	public static void main(String[] args) throws IOException {
		environment.writeln("Welcome to MyShell! You may enter commands.");

		while(true) {
			environment.write( ... prompt ...);
			String line = environment.readLine();
			String cmd = naredba...;
			String arg = predani argumenti...;
			ShellCommand shellCommand = dohvati iz tablice naredbu...
			if(shellCommand==null) {
				environment.writeln("Unknown command!");
				continue;
			}
			izvrsi naredbu; ako vrati EXIT => break;
		}

		environment.writeln("Thank you for using this shell. Goodbye!");
	}
}
```

Konačno, dodajte u ljusku sljedeće naredbe.

type filename
- na zaslon ispisuje sadržaj datoteke

filter img*.png
- pretražuje trenutni direktorij i sve njegove poddirektorije za datotekama čije ime odgovara zadanom uzorku; za svaku pronađenu datoteku na zaslon ispisuje punu stazu do te datoteke. Od zamjenskih znakova može se pojaviti znak * i to samo jednom (ali na bilo kojem mjestu: na početku, u sredini ili na kraju); taj znak je zamjena za 0 ili više proizvoljnih znakova. Pri podudaranju imena velika i mala slova se **ne razlikuju** (tj. uzorak "img*.png" je primjenjiv na "img-split.png" kao i na "ImG-Svemir.Png"). Za obilazak koristite `FileVisitor`.

copy staza1 staza2
- kopira datoteku (i samo datoteku, ako se zada direktorij, prijaviti pogrešku) zadanu prvim argumentom u direktorij staza2 (ako je to direktorij, ime datoteke se preuzima iz staze1) ili pod imenom koje definira staza2 (ako je roditelj od staza2 postojeći direktorij, pretpostavlja se da je zadnja komponenta novo ime datoteke). Ako ne postoji staza2 niti roditelj, onda prijaviti pogrešku. Kopiranje izvesti direktno koristeći binarne tokove (**ne koristiti** pomoćne metode razreda Files).

xcopy staza1 staza2
- staza1 mora biti postojeći direktorij; staza2 ili njezin roditelj mora biti postojeći direktorij. Naredba rekurzivno kopira strukturu direktorija počev od staza1 u staza2 (ako je to postojeći direktorij) odnosno u roditelja ali pod novim imenom ako je roditelj direktorij.

Naredba xcopy zahtijeva još malo pojašnjenja. Pretpostavimo da postoji:

```
D:\
  dir1
    B.txt
    dir2
      A.txt

D:\
  dirA
```

Naredba `xcopy D:\dir1 D:\dirA` mora rezultirati ovime:

```
D:\
  dirA
    dir1
      B.txt
      dir2
        A.txt
```

dok naredba `xcopy D:\dir1 D:\dirA\dirB` mora rezultirati ovime (dir1 preimenuje u dirB pri kopiranju):

```
D:\
  dirA
    dirB
      B.txt
      dir2
        A.txt
```

Pri implementaciji ove naredbe sami ste zaduženi za fizičko kopiranje datoteka (uporabom binarnih tokova); nije dopušteno napraviti kopiranje uporabom gotovih funkcija.

Zadatak 2.

Rješenje ovog zadatka smjestite u paket `hr.fer.oop.lab5.exams`. U okviru ovog zadatka potrebno je ostvariti podršku za ocjenjivanje i analizu ispitnih obrazaca koji se koriste za provođenje ispita koji studentima nude skup pitanja i ponuđenih odgovora a studenti na pripremljeni obrazac zacrnjivanjem unose svoje odgovore.

Pretpostavite da je proces digitalizacije i očitavanja odgovora sa skeniranih slika već riješen te asistent raspolaže s dvije datoteke. U jednoj datoteci nalaze se rezultati očitavanja. U toj datoteci u svakom retku nalazi se zapis za jednog studenta. Elementi u retku su razdvojeni znakom TAB, i redom su JMBAG studenta, očitana grupa te očitani odgovori na svako od pitanja. Ako student nije odgovorio na neko pitanje, kao odgovor će pisati `BLANK` (doslovno). Evo primjera za ispit koji ima 6 zadataka.

```
0036491799  D     C     D     C     B     C     E
7705816719  A     BLANK A     BLANK C     A     BLANK
...
```

U ovom primjeru prvi student je označio da je grupa D, te je na pitanja redom odgovorio C, D, C, B, C, E.

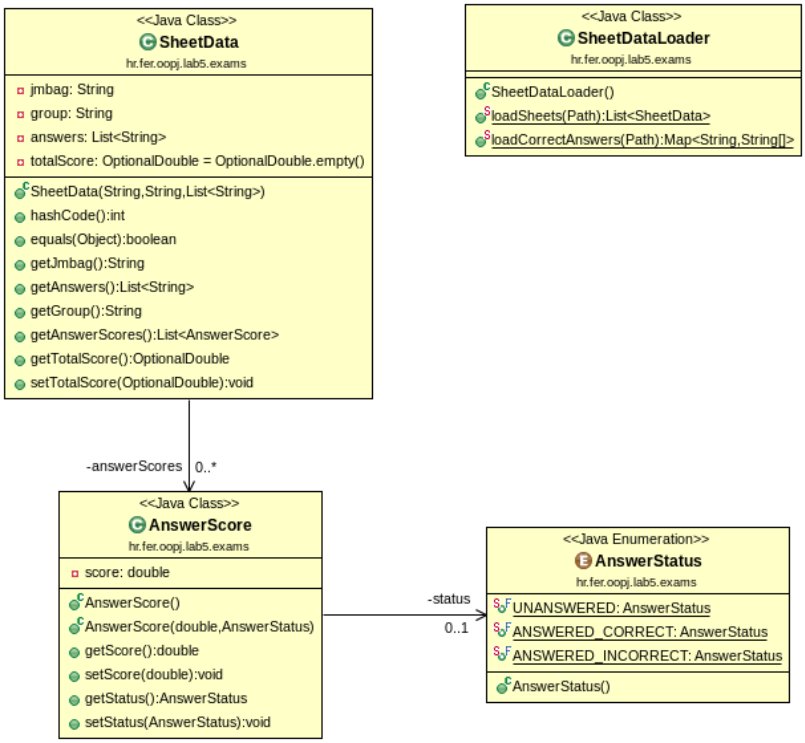
Druga datoteka sadrži točne odgovore. U svakom retku nalaze se točni odgovori za jednu grupu. Prvi element je oznaka grupe a slijede točni odgovori. Primjer je dan u nastavku.

```
A     C     B     C     C     A     D
B     C     E     D     A     B     C
C     C     D     E     B     E     A
D     C     A     B     D     C     E
```

Potrebno je napisati program koji nakon pokretanja od korisnika prima naredbe preko standardnog ulaza.

Osnovna naredba je naredba `load` koja prima pet argumenata: stazu do datoteke s odgovorima studenata, stazu do datoteke s točnim odgovorima, broj bodova za svaki točan odgovor, broj bodova za svaki netočan odgovor te broj bodova za svaki neodgovoreni zadatak. Naredba u memoriju učitava obje datoteke, stvara potrebne strukture podataka i obavlja postupak bodovanja svih ispita.

Koristite sljedeći model podataka.



Enumeracija `AnswerStatus` definira status jednog zadatka jednog studenta; može biti neodgovoren, točno odgovoren ili netočno odgovoren.

Razred `AnswerScore` predstavlja status ocjenjivanja te dodijeljene bodove za jedan zadatak jednog studenta.

Razred `SheetData` predstavlja ukupne podatke o ispitu jednog studenta. Sadrži studentov JMBAG, označenu grupu (može biti i prazna ako student nije označio grupu), listu studentovih odgovora po zadatcima, listu statusa ocjenjivanja i bodovanja svakog zadatka te ukupni broj bodova. Kako ispit nije moguće bodovati ako student nije unio grupu, bodovanje se pamti kao opcionalni decimalni broj (`OptionalDouble` – pogledajte dokumentaciju).

Razred `SheetDataLoader` sadrži dvije metode: jednu za učitavanje podataka iz datoteke s odgovorima studenata te drugu koja učitava točne odgovore.

Po izvođenju naredbe `load` program treba zapamtiti učitane podatke i na zaslon ispisati eventualne poruke (poput: "student X nije unio grupu", "student X je unio grupu za koju nisu dani točni odgovori" i slično; ocjenjivanje takvih obrazaca rezultira time da ukupni bodovi nisu postavljeni) te prosječan broj bodova ostvaren na ispitu za sve provjere koje su bodovane.

Sve učitane i ocijenjene obrasce potrebno je dodati u novu kolekciju (zvat ćemo je *aktivan skup obrazaca*) nad kojim rade naredbe koje su opisane u nastavku.

Naredba `filter` radi filtriranje aktivnog skupa obrazaca temeljem određenih kriterija čime potencijalno smanjuje taj skup. Potrebno je podržati sljedeće pozive ove naredbe:

`filter graded`	U aktivnom skupu ostavlja samo obrasce koji su bodovani.
`filter !graded`	U aktivnom skupu ostavlja samo obrasce koji nisu bodovani.
`filter group=X`	U aktivnom skupu ostavlja samo obrasce čija je grupa X (pri čemu je X

	grupa koju se zadaje).
`filter score>=X` `filter score<=X`	U aktivnom skupu ostavlja samo bodovane obrasce kojima je broj bodova veći ili jednak (odnosno manji ili jednak) od X.
`filter status X Y`	U aktivnom skupu ostavlja samo bodovane obrasce kojima je status zadatka X jednak Y; X je 1 do broj zadataka a Y može biti CORRECT, INCORRECT ili UNANSWERED.
`filter answer X Y`	U aktivnom skupu ostavlja samo obrasce kod kojih je u zadatku X student odgovorio Y (primjerice, u 3. zadatku je odgovorio B).

Nakon zadavanja naredbe `filter` naredba dodatno na zaslon mora ispisati koliko se obrazaca nalazi u aktivnom skupu. Za implementiranje ove naredbe koristite tokovni API i ponuđene mogućnosti filtriranja.

Naredba `reset` nema argumenata i aktivni skup obrazaca postavlja ponovno na cjelokupni učitani skup obrazaca.

Naredba `statistics` za sve obrasce koji su u aktivnom skupu obrazaca ispisuje prosječan broj bodova (ili informaciju da se ne može izračunati) te potom za svaki od zadataka ispisuje sljedeće podatke: broj studenata koji su na njega točno odgovorili, broj studenata koji su na njega netočno odgovorili te broj studenata koji na njega nisu odgovorili. Dodatno, naredba ispisuje i prosječni broj bodova za sve obrasce u aktivnom skupu ali po grupama (primjerice, ako se u aktivnom skupu nalaze obrasci s grupama A i B, ispisat će se i prosjek obrazaca koji su u grupi A te prosjek obrazaca koji su u grupi B). Za izračun prosjeka te za razdiobu aktivnog skupa po grupama koristite tokovni API (filtriranje, pretvorbu u tok decimalnih brojeva, izračun prosjeka; za izračun razdiobe proučite i upoznajte se s kolektorom `Collectors.groupingBy(Function<...> classifier)`).

Naredba `problem-statistics` na temelju svih obrazaca u aktivnom skupu računa statistiku za svaki od zadataka te za svaki od zadataka ispisuje broj studenata koji su odgovorili svako od slova (npr. u prvom zadatku 30 studenata je odgovorilo A, 25 B, 116 C, itd).

Naredba `student-result` prima jedan argument: JMBAG studenta. Naredba dohvaća ispit studenta (neovisno je li on u aktivnom skupu ili nije) te ispisuje podatke o tom ispitu (JMBAG, ostvareni broj bodova te potom za svaki zadatak uneseno slovo odgovora, status (T za točno, N za netočno, - za neodgovoreno) te broj bodova ostvaren na tom zadatku. Nakon učitavanja podataka iz datoteke pripremite potrebne podatkovne strukture uz koje ćete dohvat studentovog ispita moći ostvariti u složenosti O(1).

Naredba `print` na zaslon ispisuje broj obrazaca koji su u aktivnom skupu te zatim podatke o svim obrascima koji su u aktivnom skupu. Za svaki obrazac generira se jedan redak u kojem su elementi odvojeni *razmakom* (ne TABom); redak sadrži JMBAG studenta, grupu, uneseno slovo za svaki od odgovora ("BLANK" u ispisu zamijenite znakom "-") te na kraju retka broj bodova koje je student ostvario na ispitu. Ovaj ispis mora biti sortiran prema ostvarenom broju bodova (prvo se ispisuju ispiti s najvećim brojem bodova a posljednje ispiti koji nisu ocijenjeni).

Naredbu `load` korisnik može pozivati više puta (nema potrebe da gasi program te ga ponovno pokreće kako bi mogao učitati podatke novog ispita). Učitavanjem se aktivni skup automatski postavlja na skup svih obrazaca koji su stvoreni pri posljednjem učitavanju.

Dvije datoteke koje sadrže ogledne podatke s kojima možete isprobati Vaše rješenje također su dostupne na FERWeb-stranici kolegija uz ovu uputu.