

# Prvi izpitni rok pri predmetu Programiranje 1

## 18. januar 2023

Oddajte datoteke `Prva.java`, `Druga.java`, `Tretja.java` in `Cetrta.java`. Testirate jih lahko takole:

(1) `tj.exe Prva.java . .` (2) `tj.exe` (3) `tj.exe` (4) `tj.exe Cetrta.java . .`

*Število* je celo število, *beseda* pa je neprazno zaporedje črk angleške abecede.

- ① Na vhodu je podano zaporedje najmanj dveh in največ  $10^5$  števil z intervala  $[-10^5, 10^5]$ . Vsako število je zapisano v svoji vrstici. Napišite program `Prva.java`, ki izpiše maksimum vsote dveh zaporednih členov.

V 30% testnih primerov velja, da je na vhodu natanko 42 števil.

**Primer 1 (vhod/izhod):**

7	14
6	
8	
5	
9	
3	

Zaporedna člena 6 in 8 imata največjo vsoto, tj. 14. Isto vsoto imata tudi člena 5 in 9, a to ni pomembno, saj nas zanima le največja vsota.

- ② V razredu `Druga` napišite metodo

```
public static boolean križankaOK(char[][] besede, char[][] polozaji),
```

ki vrne `true` natanko v primeru, če tabeli `besede` in `polozaji` določata veljavno križanko. Križanka je veljavna, če za vsako polje velja, da imajo vse besede, ki zasedajo to polje, na pripadajočem mestu isto črko.

Tabeli `besede` in `polozaji` imata enako število ( $\leq 1000$ ) vrstic. Vrstice tabele `besede` podajajo posamezne besede, vrstice tabele `polozaji` pa njihove položaje v križanki ( $i$ -ta vrstica te tabele se nanaša na  $i$ -to besedo). Položaj besede je določen s tremi znaki: znaka med '0' in '9' podajata koordinati (indeks vrstice in stolpca) polja v križanki, kjer se pripadajoča beseda prične, znak 'v' oz. 'n' pa pove, ali je beseda postavljena vodoravno ali navpično.

Vsaka beseda je sestavljena iz samih malih črk. Dolžina vsake besede je med 1 in 9, v 30% testnih primerov pa je dolžina vsake besede enaka 1.

Sledeči tabeli določata veljavno križanko (npr. beseda `pes` se nahaja na položaju (0, 0, vodoravno), beseda `en` pa na položaju (0, 1, navpično)), če pa bi, denimo, podčrtano črko `e` spremenili v `a`, bi dobili neveljavno križanko (konflikt `pas/en`). Kot vidimo, se lahko besede tudi prekrivajo, ne samo križajo (npr. `sol/so`). Če ni konflikta, ni s tem nič narobe.

```
besede: {                polozaji: {
    {'p', 'e', 's'},      {'0', '0', 'v'},
    {'e', 'n'},           {'0', '1', 'n'},
    {'o', 'n', 'o'},      {'1', '0', 'v'},
    {'s', 'o', 'l'},      {'0', '2', 'n'},
    {'p', 'o', 't'},      {'0', '0', 'n'},
    {'s', 'o'}            {'0', '2', 'n'}
}
```

	0	1	2
0	p	e	s
1	o	n	o
2	t		l

- ③ Objekti razreda **Tablica** v razredu **Tretja** predstavljajo veljavne registrske tablice v neki državi. Vsaka tablica se prične z dvočrkovno oznako kraja (atribut **kraj** — tabela dveh znakov), nato pa sledi štirimestno število, ki ne vsebuje nobene ničle (atribut **stevilo**). Tablica je lahko začasna (atribut **zacasna** ima vrednost **true**) ali trajna.

V razredu **Tretja** redefinirajte metode **toString**, **equals** in **hashCode**:

- [32%] Metoda **toString** naj vrne niz oblike *KK NN-NN*, kjer *KK* predstavlja oznako kraja, *NNNN* pa štirimestno število. (Kraj je od števila ločen s presledkom, znak minus pa ločuje prvi dve številki števila od drugih dveh števk.)
- [34%] Metoda **equals** naj vrne **true** natanko v primeru, če podana objekta predstavljata tablici, ki se ujemata v oznaki kraja, štirimestnem številu in začasnosti (če je ena tablica začasna, druga pa trajna, mora metoda vrniti **false**).
- [34%] Metoda **hashCode** lahko vrne karkoli, morate pa zagotoviti, da bo za vsak veljaven par tablic *A* in *B* vrednost *A.equals(B)* enaka **true** *natanko tedaj, ko* bo vrednost *A.hashCode()* enaka vrednosti *B.hashCode()*.

- ④ V prvi vrstici vhoda je podano število  $n \in [1, 1000]$  in število  $u \in \{1, 2\}$  (ukaz), v primeru  $u = 2$  pa sledi še beseda *a* (artikel, ki nas zanima). Nato sledi *n* vrstic, od katerih je vsaka sestavljena iz dveh besed (ponudnik in artikel) in števila z intervala  $[1, 10^6]$  (cena podanega artikla pri podanem ponudniku). Napišite program (**Cetrta.java**), ki v primeru  $u = 1$  izpiše vse ponudnike, urejene po abecedi, v primeru  $u = 2$  pa vse ponudnike, ki prodajajo artikel *a*; ti ponudniki naj bodo naraščajoče urejeni glede na njihove cene artikla *a*. V nobenem testnem primeru se ne zgodi, da bi dva ponudnika isti artikel ponujala po isti ceni.

**Primer 1 (vhod/izhod):**

```
7 1
Golob zelje 30
Pirc korenje 60
Cvetko krompir 50
Golob krompir 30
Debeljak korenje 40
Cvetko zelje 60
Debeljak krompir 40
```

```
Cvetko
Debeljak
Golob
Pirc
```

**Primer 2 (vhod/izhod):**

```
7 2 krompir
Golob zelje 30
Pirc korenje 60
Cvetko krompir 50
Golob krompir 30
Debeljak korenje 40
Cvetko zelje 60
Debeljak krompir 40
```

```
Golob
Debeljak
Cvetko
```

Vsak ponudnik naj bo izpisan v svoji vrstici.

Namig: besede (vsaka je dolga največ 20) berite s klicem **sc.next()**, kjer je **sc** objekt tipa **Scanner**.

V 50% testnih primerov velja  $u = 1$ , v preostalih 50% pa  $u = 2$ .