Izpitni rok pri predmetu Programiranje 1
18. avgust 2022

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`

(1) Napišite program, ki prebere števili $h \in [2, 100]$ in $d \in [1, 100]$ ter po zgledu sledečih primerov nariše enakokraki trapez z višino h in dolžino zgornje (krajše) stranice d.

Vhod:

```
6 3
```

Izhod:

```
     ***
    *   *
   *     *
  *       *
 *         *
*************
```

Vhod:

```
4 7
```

Izhod:

```
   *******
  *       *
 *         *
*************
```

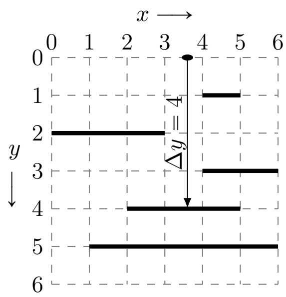
Pozor: vsaka vrstica se konča z zvezdico (ne s presledkom!).

(2) Na steni velikosti $m \times m$ ($m \in [1, 100]$) je montiranih $n \in [0, 100]$ polic. Koordinate levih robov in dolžine polic so zapisane v tabeli velikosti $n \times 3$ (prvi stolpec podaja koordinate x, drugi koordinate y, tretji pa dolžine posameznih polic). Koordinate x naraščajo v desno, koordinate y pa navzdol.

Napišite metodo

```
public static int najGlobina(int m, int[][] p),
```

ki vrne največjo možno dolžino poti, ki jo prepotuje kamen, če ga navpično spustimo z neke točke na vrhu stene. Kamen seveda pade samo do najvišje police na svoji poti (oziroma do tal, če ima prosto pot).



V primeru na sliki imamo $m = 6$ in tabelo `{{0, 2, 3}, {4, 1, 1}, {2, 4, 3}, {1, 5, 5}, {4, 3, 2}}`, rezultat pa je enak 4.

V 50% testnih primerov velja, da vsaka navpična premica seka natanko eno polico.

(3) Na neki fakulteti (brez skrbi, ne na FRI) so odnosi med pedagogi strogo hierarhični. Celo pri uporabi skupnega tiskalnika imajo profesorji prednost pred docenti, ti pa pred asistenti. Ko želi asistent natisniti nek dokument, ga tiskalnik postavi povsem na konec čakalne vrste. Docenti so na boljšem, saj njihovi dokumenti prehitijo vse asistentske, a nobenega drugega. Dokumenti profesorjev prehitijo vse docentske in asistentske dokumente.

Kaj pa tiskalnik? Ko se mu »zljubi«, vzame prvi dokument iz vrste in ga natisne, sicer pa lagodno čaka na zahteve in jih razvršča po opisanih pravilih.

V razredu `Tretja.Tiskalnik` dopolnite sledeče elemente:

- `public Tiskalnik()`

  Glede konstruktorja (in atributov) ste svobodni.

- `public void prejmi(Pedagog pedagog, String dokument)`

  Ta metoda se pokliče, ko podani pedagog pošlje tiskalniku zahtevo po tiskanju podanega dokumenta. Objekt `pedagog` je tipa `Asistent`, `Docent` ali `Profesor` (to so podrazredi abstraktnega razreda `Pedagog`).

- [32%] `public int dolzinaVrste()`

  Vrne trenutno število vseh dokumentov v vrsti.

- [34%] `public int steviloDokumentov(Pedagog pedagog)`

  Vrne trenutno število dokumentov v vrsti, ki pripadajo pedagogom, ki so istega tipa kot podani pedagog. (Na primer, če je objekt `pedagog` tipa `Docent`, nas zanima število vseh docentskih dokumentov v vrsti.)

- [34%] `public String natisni()`

  Ta metoda se pokliče, ko tiskalnik vzame prvi dokument iz vrste in ga natisne. Metoda naj vrne natisnjeni dokument. (Če je vrsta ob klicu metode prazna, naj metoda ne stori ničesar in zgolj vrne `null`.)

Noben testni primer ne vsebuje več kot 1000 klicev metod razreda `Tretja`.

Oglejmo si primer (`Test01.java`):

```
Tiskalnik t = new Tiskalnik();          // vrsta: []
System.out.println(t.natisni());        // null; vrsta: []
t.prejmi(new Docent(), "d1");           // vrsta: [d1]
t.prejmi(new Profesor(), "p1");         // vrsta: [p1, d1]
t.prejmi(new Asistent(), "a1");         // vrsta: [p1, d1, a1]
t.prejmi(new Docent(), "d2");           // vrsta: [p1, d1, d2, a1]
t.prejmi(new Profesor(), "p2");         // vrsta: [p1, p2, d1, d2, a1]
System.out.println(t.natisni());        // p1; vrsta: [p2, d1, d2, a1]
t.prejmi(new Profesor(), "p3");         // vrsta: [p2, p3, d1, d2, a1]
System.out.println(t.dolzinaVrste());   // 5
System.out.println(t.steviloDokumentov(new Profesor()));  // 2
```

(4) V razredu `Cetrta` dopolnite sledeče metode:

- [32%] `public static Set<Integer> vecji(Map<Integer, Integer> slovar)`

  Vrne množico vseh ključev v podanem slovarju, ki so večji od pripadajočih vrednosti.

- [34%] `public static <T> Set<T> vecjiPrim(Map<T, T> slovar,`
  `Comparator<T> primerjalnik)`

  Vrne množico vseh ključev v podanem slovarju, ki so (sodeč po podanem primerjalniku) večji od pripadajočih vrednosti.

- [34%] `public static <T> Comparator<T> primerjalnik(Map<T, T> slovar,`
  `Set<T> mnozica)`

  Vrne primerjalnik, ki objekt A proglasi za večjega od objekta B, če velja vse sledeče: (1) objekt A nastopa kot ključ v podanem slovarju; (2) temu ključu (objektu A) v slovarju pripada vrednost B; (3) objekt A je element podane množice.

  V vseh ostalih primerih naj primerjalnik proglasi objekt A za manjšega od objekta B.