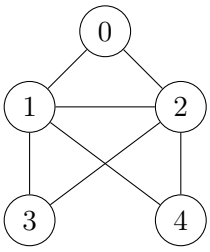
Primer izpita pri predmetu Programiranje I

Tik pred začetkom ...

- Na voljo imate 90 minut časa.
- Ni nujno, da se vam bo prva naloga zdela najlažja, tretja pa najtežja.
- Pazite, da se ne boste predolgo ukvarjali s težjimi primeri oz. podnalogami pri nalogi X, pri tem pa povsem pozabili na nalogo Y.
- Navodila za testiranje in oddajo najdete v datotekah `README.txt` v mapah, ki pripadajo posameznim nalogam.
- Na učilnico oddajte točno tisto, kar piše v navodilih. Oddajajte sproti!
- Sproti testirajte!
- Pazite na zahrbtne napake, kot so npr. zamenjava indeksov `i` in `j`, podpičje na koncu glave zanke, ...
- Najboljši zatiralec hroščev in izjem je `System.out.println`/`printf`.
- Dopolnite že pripravljene razrede. Seveda lahko vanje po potrebi dodajate tudi pomožne metode.
- Goljufe čaka Dantejev pekel.

Držimo pesti!

(1) Enostaven neusmerjen graf z vozlišči 0, 1, 2, ..., $n-1$ ($n \geqslant 2$) lahko predstavimo z *matriko sosednosti* (tabela tipa `boolean[][]` velikosti $n \times n$, v kateri element `[i][j]` pove, ali sta vozlišči i in j povezani) ali pa s *seznamom sosednosti* (nepravokotna tabela tipa `int[][]` z n vrsticami, v kateri vrstica `[i]` vsebuje indekse vozlišč, povezanih z vozliščem i). Na primer:




```
// Matrika sosednosti:
boolean[][] m = {
    {F, T, T, F, F},
    {T, F, T, T, T},
    {T, T, F, T, T},
    {F, T, T, F, F},
    {F, T, T, F, F}
}; // (T = true, F = false)
```

```
// Seznam sosednosti:
int[][] s = {
    {1, 2},
    {0, 2, 3, 4},
    {0, 1, 3, 4},
    {1, 2},
    {1, 2}
};
```

V razredu `Graf` dopolnite sledeče metode:

- `public static boolean imaEulerjevObhod(boolean[][] m)`: [1–16]

  Vrne `true` natanko v primeru, če ima graf, podan z matriko sosednosti `m`, Eulerjev obhod. To velja natanko tedaj, ko ima vsako vozlišče sodo število sosedov. Na primer, graf na gornji sliki ima Eulerjev obhod.

- `public static boolean[][] vMatriko(int[][] s)`: [17–33]

  Vrne matriko sosednosti, ki opisuje isti graf kot podani seznam sosednosti `s`.

- `public static boolean istiGraf(boolean[][] m, int[][] s)`: [34–50]

  Vrne `true` natanko v primeru, če matrika sosednosti `m` in seznam sosednosti `s` opisujeta isti graf.

(2) Zaporedje k-kotnih števil je definirano takole:

$$a_1 = 1 \xrightarrow{+k-1} a_2 \xrightarrow{+2k-3} a_3 \xrightarrow{+3k-5} a_4 \xrightarrow{+4k-7} a_5 \xrightarrow{+5k-9} a_6 \ldots$$

Na primer, zaporedje 5-kotnih števil je 1, 5, 12, 22, 35, 51, ...:

$$1 \xrightarrow{+4} 5 \xrightarrow{+7} 12 \xrightarrow{+10} 22 \xrightarrow{+13} 35 \xrightarrow{+16} 51 \ldots$$

Razred `Zaporedje` dopolnite s sledečimi konstruktorji in metodami (ter po potrebi z atributi):

- `public Zaporedje(int k)`: [1–50]

  Ustvari objekt razreda `Zaporedje`, s pomočjo katerega bo mogoče (z zaporednimi klici metode `naslednje`) tvoriti zaporedje `k`-kotnih števil.

- `public int naslednje()`: [1–50]

  Vrne naslednje število v zaporedju. Ko se metoda nad nekim objektom tipa `Zaporedje` pokliče prvič, naj vrne prvo število (to je vedno 1), ko se pokliče drugič, naj vrne drugo število v zaporedju itd.

- `public int katero()`: [17–50]

  Vrne zaporedno številko števila, ki ga bo vrnil naslednji klic metode `naslednje`.

- `public void ponastavi(int n)`: [34–50]

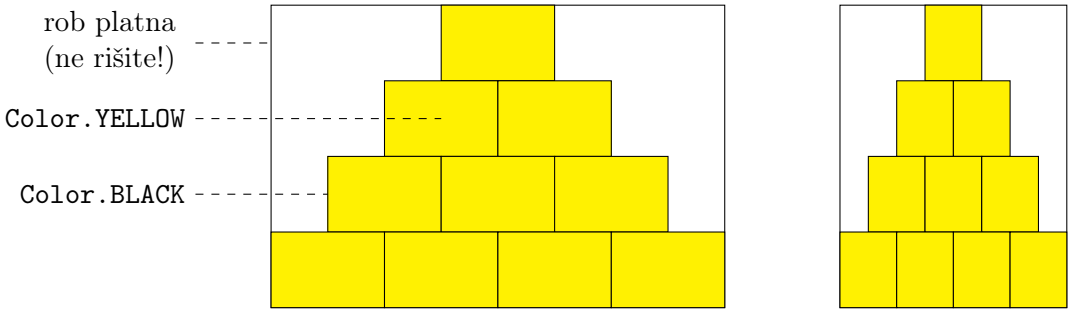
  Naredi vse, kar je treba, da bo naslednji klic metode `naslednje` vrnil `n`-to število v zaporedju. Velja $\texttt{n} \geqslant 1$.

Za lažje razumevanje si oglejmo primer uporabe razreda `Zaporedje`:

```
Zaporedje z = new Zaporedje(5);         // tvorili bomo 5-kotna števila
System.out.println( z.naslednje() );    // 1
System.out.println( z.naslednje() );    // 5
System.out.println( z.naslednje() );    // 12
System.out.println( z.katero() );       // 4 (metoda naslednje bo naslednjič
                                        //    vrnila četrto število v zaporedju)
System.out.println( z.naslednje() );    // 22
System.out.println( z.katero() );       // 5

z.ponastavi(3);
System.out.println( z.katero() );       // 3
System.out.println( z.naslednje() );    // 12
System.out.println( z.naslednje() );    // 22
```

(3) Dopolnite metodo `narisi` v razredu `Piramida` tako, da bo narisala »piramido« višine `visina`. Število `visina` (podano je kot atribut) je v vseh primerih enako najmanj 2. Sledeča slika prikazuje piramido višine 4 pri dveh različnih velikostih platna:



Piramida naj bo sestavljena iz enako velikih blokov (obrobljenih pravokotnikov) in naj po obeh dimenzijah zavzema celotno površino platna.

Dopolnite tudi sledeči metodi:

- `public double sirinaBloka(double wp, double hp)` [1–5]
- `public double visinaBloka(double wp, double hp)` [6–10]

Metodi naj vrneta širino oziroma višino posameznega bloka piramide v odvisnosti od širine (`wp`) in višine (`hp`) platna.