Dodatne naloge – Dedovanje

Splošna navodila

Pri nalogah 1 in 2 lahko svoje rešitve preverite z množico vhodnih in pripadajočih izhodnih datotek, pri nalogah 3 in 4 pa z množico testnih razredov in pripadajočih izhodnih datotek.

1 Poštne pošiljke

Naloga

Na poštnem uradu razpošiljajo tri vrste pošiljk: navadna pisma, priporočena pisma in telegrame. Za vsako pošiljko sta podana naslovnik in vsebina. Za navadna pisma je poleg tega podana še razdalja do naslovnika, za priporočena pisma pa (poleg naslovnika, vsebine in razdalje) še pošiljatelj. Naslovniki, pošiljatelji in vsebine pošiljk so podani kot nizi, sestavljeni iz črk angleške abecede, števk in podčrtajev.

Cena oddaje navadnega pisma je odvisna zgolj od razdalje: za razdaljo od 0 do vključno $(R-1)$ dolžinskih enot znaša cena Z denarnih enot, za razdaljo od R do vključno $(2R-1)$ znaša $(Z+D)$ enot, za razdaljo od $2R$ do vključno $(3R-1)$ je cena enaka $(Z+2D)$ itd. Cena oddaje priporočenega pisma se izračuna tako, da se cena oddaje navadnega pisma pomnoži s faktorjem P. Cena telegrama je enaka cT, kjer je c število črk v vsebini telegrama (števke in podčrtaji ne štejejo).

Napišite program, ki prebere zaporedje poštnih pošiljk in ukaz (celo število med 1 in vključno 3), nato pa na podlagi ukaza proizvede ustrezni izpis:

- Če je ukaz enak 1, naj se izpišejo podatki o vseh pošiljkah. Za vsako pošiljko mora izpis vsebovati njeno vrsto, naslovnika, vsebino, morebitne dodatne podatke (odvisno od vrste pošiljke) in ceno.
- Če je ukaz enak 2, naj se izpišejo podatki o najdražji pošiljki (o prvi od njih, če jih je več).
- Če je ukaz enak 3, naj se za vsako priporočeno pismo izpišejo podatki o njegovi povratnici. Povratnica priporočenega pisma je priporočeno pismo, pri katerem je razdalja enaka kot pri izhodiščnem pismu, naslovnik in pošiljatelj sta med seboj zamenjana, vsebina pa je niz `povratnica`.

Vrstni red izpisa pošiljk na izhodu mora biti enak vrstnemu redu pošiljk na vhodu.

Vhod

V prvi vrstici vhoda je v tem vrstnem redu nanizanih pet celih števil z intervala $[1, 1000]$: Z, R, D, P in T. V drugi vrstici je podano celo število $n \in [1, 100]$. Sledi n vrstic s podatki o pošiljkah. V vsaki vrstici je najprej podana vrsta pošiljke (`navadnoPismo`, `priporocenoPismo` oziroma `telegram`), nato pa sledita naslovnik in vsebina. Pri navadnem pismu sledi še razdalja, pri priporočenem pa razdalja in pošiljatelj. V zadnji vrstici vhoda je zapisano število $U \in \{1, 2, 3\}$, ki predstavlja ukaz.

Vsi nizi vsebujejo od 1 do 100 znakov. Vse razdalje so cela števila z intervala $[0, 1000]$. Podatki znotraj iste vrstice so med seboj ločeni s presledkom.

Izhod

Na izhodu izpišite podatke, ki jih zahteva ukaz. Zgledujte se po sledečih primerih. Zadnji podatek v vsaki vrstici je cena pošiljke.

Testni primer 1

Vhod:

```
30 50 10 3 9
7
telegram Ana_Antolinc vsebina_telegrama_1
priporocenoPismo Branko_Bizjak vsebina_pp_1 150 Cvetka_Cevc
navadnoPismo Danica_Dobravc vsebina_np_1 75
telegram Eva_Erker zelo_dolga_VSEBINA_s_5t3v1lk4m1
navadnoPismo Franci_Furman vsebina_np_2 49
navadnoPismo Gorazd_Gaber vsebina_np_2 50
priporocenoPismo Hinko_Hojc vsebina_pp_2 50 Iva_Intihar
1
```

Izhod:

```
T | Ana_Antolinc | vsebina_telegrama_1 | 144
PP | Branko_Bizjak | vsebina_pp_1 | 150 | Cvetka_Cevc | 180
NP | Danica_Dobravc | vsebina_np_1 | 75 | 40
T | Eva_Erker | zelo_dolga_VSEBINA_s_5t3v1lk4m1 | 198
NP | Franci_Furman | vsebina_np_2 | 49 | 30
NP | Gorazd_Gaber | vsebina_np_2 | 50 | 40
PP | Hinko_Hojc | vsebina_pp_2 | 50 | Iva_Intihar | 120
```

Testni primer 2

Vhod:

```
30 50 10 3 9
7
telegram Ana_Antolinc vsebina_telegrama_1
priporocenoPismo Branko_Bizjak vsebina_pp_1 150 Cvetka_Cevc
navadnoPismo Danica_Dobravc vsebina_np_1 75
telegram Eva_Erker zelo_dolga_VSEBINA_s_5t3v1lk4m1
navadnoPismo Franci_Furman vsebina_np_2 49
navadnoPismo Gorazd_Gaber vsebina_np_2 50
priporocenoPismo Hinko_Hojc vsebina_pp_2 50 Iva_Intihar
2
```

Izhod:

```
T | Eva_Erker | zelo_dolga_VSEBINA_s_5t3v1lk4m1 | 198
```

Testni primer 3

Vhod:

```
30 50 10 3 9
7
telegram Ana_Antolinc vsebina_telegrama_1
priporocenoPismo Branko_Bizjak vsebina_pp_1 150 Cvetka_Cevc
navadnoPismo Danica_Dobravc vsebina_np_1 75
telegram Eva_Erker zelo_dolga_VSEBINA_s_5t3v1lk4m1
navadnoPismo Franci_Furman vsebina_np_2 49
navadnoPismo Gorazd_Gaber vsebina_np_2 50
priporocenoPismo Hinko_Hojc vsebina_pp_2 50 Iva_Intihar
3
```

Izhod:

```
PP | Cvetka_Cevc | povratnica | 150 | Branko_Bizjak | 180
PP | Iva_Intihar | povratnica | 50 | Hinko_Hojc | 120
```

Napotki

Nize beremo s klicem `sc.next()`, pri čemer je `sc` objekt razreda `Scanner`. Če je `s` niz (objekt tipa `String`), potem njegovo dolžino (število znakov) pridobimo s klicem `s.length()`, znak z indeksom `i` pa s klicem `s.charAt(i)`. Na vprašanje, ali je podani znak `c` (spremenljivka tipa `char`) črka, odgovorimo s klicem `Character.isLetter(c)`.

2 Geometrijska telesa

Naloga

Na vhodu so zapisani osnovni podatki o različnih geometrijskih telesih: kvadrih, kockah in kroglah. Kvader je določen s tremi stranicami, kocka z eno samo, krogla pa s polmerom. Napišite program, ki prebere podatke o telesih ter izpiše njihove prostornine in površine, pri kvadrih in kockah pa tudi mreže. Prostornine in površine naj bodo zaokrožene na najbližje celo število.

Pri risanju mreže naj se zgornja in spodnja stranica kvadra oz. kocke prikažeta z znaki `o`, sprednja in zadnja z znaki `*`, leva in desna pa z znaki `+`. Znaki v isti vrstici naj bodo med seboj ločeni s presledkom. Na primer, mreža kvadra s stranicami $a = 2$, $b = 3$ in $c = 4$ naj se nariše tako:

```
        * * *
        * * *
        * * *
        * * *
+ + + + o o o + + + +
+ + + + o o o + + + +
        * * *
        * * *
        * * *
```

```
        * * *
        o o o
        o o o
```

oziroma (s prikazanimi presledki)

```
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
+␣+␣+␣+␣o␣o␣o␣+␣+␣+␣+
+␣+␣+␣+␣o␣o␣o␣+␣+␣+␣+
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣o␣o␣o
␣␣␣␣␣␣␣␣o␣o␣o
```

V izpisu morajo biti telesa urejena po padajočih prostorninah. Urejanje mora biti stabilno: telesa z enakimi prostorninami morajo biti na izhodu izpisana v istem medsebojnem vrstnem redu, kot so podana na vhodu.

Vhod

V prvi vrstici vhoda je podano celo število $n \in [1, 100]$, nato pa sledi n vrstic s podatki o posameznih telesih. Na začetku vsake vrstice je zapisano število 1 (to število predstavlja kvader), 2 (kocka) oziroma 3 (krogla), nato pa sledijo dolžine stranic (pri kvadru), dolžina stranice (pri kocki) oziroma polmer (pri krogli). Vsi navedeni podatki so cela števila z intervala $[1, 100]$, med seboj pa so ločeni s presledkom.

Izhod

Pri izpisu na izhod se zgledujte po primeru v nadaljevanju. Ne izpisujte odvečnih presledkov ali praznih vrstic!

Testni primer 1

Vhod:

```
6
2 4
1 4 2 8
3 4
2 3
3 2
1 8 4 2
```

Izhod:

```
krogla
r = 4
```

```
V = 268
P = 201
======
kocka
a = 4
V = 64
P = 96
        * * * *
        * * * *
        * * * *
        * * * *
+ + + + o o o o + + + +
+ + + + o o o o + + + +
+ + + + o o o o + + + +
+ + + + o o o o + + + +
        * * * *
        * * * *
        * * * *
        * * * *
        o o o o
        o o o o
        o o o o
        o o o o
======
kvader
a = 4
b = 2
c = 8
V = 64
P = 112
                * *
                * *
                * *
                * *
                * *
                * *
                * *
                * *
+ + + + + + + + o o + + + + + + + + +
+ + + + + + + + o o + + + + + + + + +
+ + + + + + + + o o + + + + + + + + +
+ + + + + + + + o o + + + + + + + + +
                * *
                * *
                * *
                * *
                * *
                * *
                * *
                * *
                o o
```

```
                o o
                o o
                o o
======
kvader
a = 8
b = 4
c = 2
V = 64
P = 112
    * * * *
    * * * *
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
    * * * *
    * * * *
    o o o o
    o o o o
    o o o o
    o o o o
    o o o o
    o o o o
    o o o o
    o o o o
======
krogla
r = 2
V = 34
P = 50
======
kocka
a = 3
V = 27
P = 54
      * * *
      * * *
      * * *
+ + + o o o + + +
+ + + o o o + + +
+ + + o o o + + +
      * * *
      * * *
      * * *
      o o o
      o o o
```

```
     o o o
======
```

3 Seznam (⋆)

Naloga

Seznam je podatkovna struktura, ki predstavlja zaporedje elementov. Obstajata dve vrsti seznamov:

Prazen seznam: To je seznam, ki ne vsebuje nobenih elementov.

Neprazen seznam: To je seznam, ki je sestavljen iz *glave* in *repa*. Glava je prvi element seznama (v tej nalogi bo to vedno neko celo število), rep pa je seznam (prazen ali neprazen). Seznam z elementi 3, 5 in 7 je sestavljen iz glave 3 in repa, ki je seznam, sestavljen iz glave 5 in repa, ki je seznam, sestavljen iz glave 7 in praznega seznama. Če prazen seznam zapišemo kot `[]`, neprazen pa kot `[`*glava* `|` *rep*`]`, bi lahko seznam z elementi 3, 5 in 7 zapisali kot `[3 | [5 | [7 | []]]]`.

To definicijo lahko prevedemo v hierarhijo razredov, ki jo sestavljajo abstrakten razred `Seznam` ter njegova podrazreda `Prazen` in `Neprazen`. Realizirajte vse tri razrede. Razred `Prazen` mora vsebovati konstruktor

```
public Prazen()
```

ki izdela objekt, ki predstavlja prazen seznam. Razred `Neprazen` pa mora vsebovati konstruktor

```
public Neprazen(int glava, Seznam rep)
```

ki izdela objekt, ki predstavlja seznam s podano glavo in repom. Na primer, objekt, ki predstavlja seznam z elementi 3, 5 in 7, bi s pomočjo navedenih konstruktorjev ustvarili takole:

```
Seznam s = new Neprazen(3, new Neprazen(5, new Neprazen(7, new Prazen())));
```

V nadaljevanju bomo podali množico javno dostopnih metod, ki jih mora ponujati razred `Seznam`. Premislite, katere od teh metod naj bodo v razredu `Seznam` abstraktne (in definirane v obeh podrazredih), katere pa je možno definirati že v razredu `Seznam`. Nobena metoda ne spremeni seznama `this` in nobena ni daljša od nekaj vrstic.

Razred `Seznam` mora ponujati sledeče metode:

- `public int glava()`:

  Če je seznam `this` neprazen, vrne njegovo glavo, sicer pa sproži izjemo tipa `NoSuchElementException`:

  ```
  throw new java.util.NoSuchElementException();
  ```

- `public Seznam rep()`:

  Če je seznam `this` neprazen, vrne njegov rep, sicer pa sproži izjemo tipa `NoSuchElementException`:

- `public boolean jePrazen()`:

  Vrne `true` natanko v primeru, če je seznam prazen.

- `public Seznam dodajZ(int element)`:

  Vrne nov seznam, ki ima v glavi podani element, njegov rep pa je seznam `this`.

- `public Seznam dodajK(int element)`:

  Vrne nov seznam, ki je enak seznamu `this`, le da ima na koncu dodan še podani element.

  Pri praznem seznamu ta metoda deluje enako kot metoda `dodajZ`, pri nepraznem pa metoda `dodajK` izdela in vrne nov neprazen seznam, čigar glava je enaka glavi seznama `this`, rep pa je enak seznamu, ki ga dobimo, če na konec repa seznama `this` dodamo podani element.

- `public Seznam dodajU(int element)`:

  Ob predpostavki, da je seznam `this` naraščajoče urejen, ta metoda vrne nov seznam, ki je enak seznamu `this`, le da ima podani element vstavljen na mestu, kamor spada po velikosti. Na primer, če seznam `s` po vrsti vsebuje elemente 3, 5 in 7, potem klic `s.dodajU(6)` vrne seznam [3, 5, 6, 7], klic `s.dodajU(2)` pa seznam [2, 3, 5, 7].

  Pri nepraznem seznamu upoštevajte dve možnosti: (1) podani element je manjši ali enak glavi seznama `this`; (2) podani element je večji od glave.

- `public boolean vsebuje(int element)`:

  Vrne `true` natanko v primeru, če seznam vsebuje podani element. Prazen seznam elementa gotovo ne vsebuje, pri nepraznem seznamu pa moramo najprej preveriti glavo, če ta ni enaka iskanemu elementu, pa preiščemo še rep.

- `public Seznam odstrani(int element)`:

  Vrne nov seznam, ki je enak seznamu `this`, le da je v njem odstranjen prvi element, ki je enak podanemu elementu. Na primer, če je seznam `s` enak [7, 3, 6, 3, 2, 3], potem klic `s.odstrani(3)` vrne seznam [7, 6, 3, 2, 3]. Če seznam `this` podanega elementa ne vsebuje, naj bo rezultat metode enak seznamu `this`.

- `public Seznam uredi()`:

  Vrne naraščajoče urejeno kopijo seznama `this`. Na primer, klic metode nad seznamom [7, 6, 3, 2, 3] bi vrnil seznam [2, 3, 3, 6, 7].

  Namig: pri nepraznem seznamu najprej uredimo rep.

- `public Seznam odstraniDuplikate()`:

  Vrne nov seznam, ki je enak seznamu `this`, le da v njem vsak element nastopa samo po enkrat. Ohraniti se mora le *zadnja* pojavitev elementa. Na primer, klic metode nad seznamom [7, 3, 6, 3, 2, 3, 7, 2, 9] bi vrnil seznam [6, 3, 7, 2, 9].

- `public String toString()`:

  Vrne niz oblike

  `[`a_1`,␣`a_2`,␣...,␣`a_n`]`

  kjer so a_1, a_2, ..., a_n elementi seznama `this`. Na primer, klic metode nad seznamom [7, 6, 3, 2, 3] bi vrnil sledeči niz:

  ```
  [7, 6, 3, 2, 3]
  ```

Pri realizaciji metode `toString` vam bo morda koristila pomožna metoda, ki izdela niz brez oklepajev.

Testni primer 10

Testni razred (in pripadajoči izhod):

```
public class Test10 {

    public static void main(String[] args) {
        Seznam seznam = new Prazen();
        System.out.println(seznam.jePrazen());  // true
        System.out.println(seznam.toString());  // []

        seznam = seznam.dodajU(30);
        seznam = seznam.dodajU(10);
        seznam = seznam.dodajU(40);
        seznam = seznam.dodajU(20);
        seznam = seznam.dodajU(60);
        seznam = seznam.dodajU(50);
        seznam = seznam.dodajU(10);
        seznam = seznam.dodajU(20);
        System.out.println(seznam.jePrazen());  // false
        System.out.println(seznam.toString());
            // [10, 10, 20, 20, 30, 40, 50, 60]

        seznam = seznam.dodajZ(50);
        seznam = seznam.dodajZ(60);
        seznam = seznam.dodajK(20);
        seznam = seznam.dodajK(50);
        System.out.println(seznam.toString());
            // [60, 50, 10, 10, 20, 20, 30, 40, 50, 60, 20, 50]

        seznam = seznam.odstrani(30);
        seznam = seznam.odstrani(50);
        seznam = seznam.odstrani(30);
        System.out.println(seznam.toString());
            // [60, 10, 10, 20, 20, 40, 50, 60, 20, 50]

        Seznam urejen = seznam.uredi();
        System.out.println(urejen.toString());
            // [10, 10, 20, 20, 20, 40, 50, 50, 60, 60]

        seznam = seznam.odstraniDuplikate();
        System.out.println(seznam.toString());  // [10, 40, 60, 20, 50]
    }
}
```

4 Naravno število (⋆)

Naloga

Cilj te naloge je predstaviti naravna števila (1, 2, 3, . . .) in realizirati nekatere operacije nad njimi s pomočjo dedovanja in rekurzije. Ker bi lahko nalogo brez posebnih težav razširili na celotno množico celih števil, se izkaže, da celoštevilskih podatkovnih tipov pravzaprav sploh ne potrebujemo. Ta drzna trditev pa velja le na konceptualni ravni, saj časovna in prostorska učinkovitost operacij, ki jih boste realizirali v tej nalogi, ne bo ravno najboljša. Vendar pa naj tokrat estetske vrednote prevladajo nad materialnimi!

Množico naravnih števil lahko definiramo na sledeči način (Peanovi aksiomi):

- Število 1 je naravno število.
- Naslednik naravnega števila je tudi naravno število.

To definicijo lahko prevedemo v hierarhijo razredov, ki jo sestavljajo abstrakten razred `NaravnoStevilo` ter njegova podrazreda `Ena` in `Naslednik`. Realizirajte vse tri razrede. Razred `Ena` mora vsebovati konstruktor

```
public Ena()
```

ki izdela objekt, ki predstavlja naravno število 1. Razred `Naslednik` pa naj vsebuje konstruktor

```
public Naslednik(NaravnoStevilo stevilo)
```

ki izdela objekt, ki predstavlja naslednika podanega naravnega števila. Na primer, objekt, ki predstavlja naravno število 3, bi s pomočjo navedenih konstruktorjev ustvarili takole:

```
NaravnoStevilo tri = new Naslednik(new Naslednik(new Ena()));
```

V nadaljevanju bomo podali množico javno dostopnih metod, ki jih mora ponujati razred `NaravnoStevilo`. Premislite, katere od teh metod naj bodo v razredu `NaravnoStevilo` abstraktne (in definirane v obeh podrazredih), katere pa je možno definirati že v razredu `NaravnoStevilo`. Nobena metoda ne spremeni naravnega števila `this` in nobena ni daljša od nekaj vrstic.

Razred `NaravnoStevilo` naj ponuja sledeče metode:

- `public boolean jeEna()`:

  Vrne `true` natanko v primeru, če objekt `this` predstavlja število 1.

- `public NaravnoStevilo naslednik()`:

  Vrne naravno število, ki je naslednik naravnega števila `this`.

- `public NaravnoStevilo predhodnik()`:

  Vrne naravno število, ki je predhodnik naravnega števila `this`. Upoštevajte, da je predhodnik števila `new Naslednik(`n`)` kar število n. Ker število 1 nima predhodnika, naj metoda zanj sproži izjemo tipa `NoSuchElementException`:

  ```
  throw new java.util.NoSuchElementException();
  ```

- `public NaravnoStevilo vsota(NaravnoStevilo stevilo)`:

  Vrne naravno število, ki je vsota naravnega števila `this` in števila `stevilo`. Namig: primer $1 + n$ sodi v razred `Ena`, primer $m + n$ za $m > 1$ pa v razred `Naslednik`.

- `public NaravnoStevilo razlika(NaravnoStevilo stevilo)`:

  Vrne naravno število, ki je razlika naravnega števila `this` in števila `stevilo`. Če rezultat razlike ni naravno število, naj metoda sproži izjemo tipa `NoSuchElementException`.

- `public NaravnoStevilo zmnozek(NaravnoStevilo stevilo)`:

  Vrne naravno število, ki je zmnožek naravnega števila `this` in števila `stevilo`.

- `public String toString()`:

  Če objekt `this` predstavlja število 1, vrne niz `1`, sicer pa vrne niz `s(`t`)`, kjer je t niz, ki ga metoda `toString` vrne za predhodnika števila `this`. Na primer, za število 3 naj metoda vrne niz `s(s(1))`.

- `public int toInt()`:

  Vrne celo število, ki ga predstavlja objekt `this`.

- `public static NaravnoStevilo ustvariIzInt(int n)`:

  Ustvari objekt, ki predstavlja podano pozitivno celo število `n`. To metodo morate seveda definirati v razredu `NaravnoStevilo`.

Testni primer 10

Testni razred (in pripadajoči izhod):

```
public class Test10 {

    public static void main(String[] args) {
        NaravnoStevilo ena = new Ena();
        NaravnoStevilo dve = ena.naslednik().naslednik().predhodnik();
        NaravnoStevilo pet = new Naslednik(new Naslednik(new Naslednik(dve)));
        NaravnoStevilo osem = NaravnoStevilo.ustvariIzInt(10).
                                predhodnik().predhodnik();

        System.out.println(ena.jeEna());  // true
        System.out.println(dve.jeEna());  // false
        System.out.println(dve.predhodnik().jeEna());  // true

        NaravnoStevilo a = dve.vsota(pet);
        NaravnoStevilo b = osem.razlika(dve);
        NaravnoStevilo c = pet.zmnozek(dve);
        NaravnoStevilo d = pet.razlika(dve).razlika(dve);
        NaravnoStevilo e = ena.vsota(pet).zmnozek(osem.razlika(ena));

        System.out.println(a.toInt());      // 7
        System.out.println(b.toString());   // s(s(s(s(s(1)))))
        System.out.println(c.toInt());      // 10
        System.out.println(d.toString());   // 1
        System.out.println(e.toInt());      // 42
    }
}
```