Osnove Programiranja I
Dodatne naloge za študijsko leto 2006/07

PRAVILA:

- V študijskem letu 2006/07 bodo vaje dvanajstkrat, izvajale pa se bodo v računalniški učilnici LRI5 oziroma v računalniški učilnici PIC-a (v Sežani). Na vajah se bomo praktično učili programiranja v programskem jeziku Java.

- Udeležba na vajah je obvezna za vse študente, ki so v študijskem letu 2006/07 prvič vpisani v 1. letnik, vaj pa se lahko udeležijo tudi drugi študentje, ki so vpisani v 1. letnik (ponavljalci in prepisani študentje, ki imajo status). Zaradi prostorske stiske vaj ne morejo obiskovati študentje, ki niso vpisani v 1. letnik.

- Na vsakih vajah bodo asistenti postopoma reševali primere nalog, ki bodo usklajeni s snovjo predavanj. Temu bo namenjena vsaj polovica časa, predvidenega za vaje. V preostalem času pa bodo morali študenti sami na računalniku rešiti podobno nalogo (ali nadgraditi podan primer). Pri tem lahko za nasvete prosijo asistenta. Udeležba na posameznih vajah se bo upoštevala samo tistim študentom, ki bodo prišli do rešitve (ali pokazali primerno delavnost).

- Prvič vpisani študentje 1. letnika se morajo udeležiti vsaj 9 od 12 vaj. V tem primeru se lahko udeležijo kateregakoli izpitnega roka, v nasprotnem primeru pa se ne bodo smeli udeležiti zimskih izpitnih rokov (januar/februar 2007) in bodo šli lahko prvič na izpit šele junija 2007.

  Študenti, ki so poslušali predmet Osnove programiranja I v prejšnjih študijskih letih, se lahko prijavijo za katerikoli izpitni rok.

- Študentje, ki že imajo dovolj predzanaja Jave, lahko namesto udeležbe na vajah doma rešujejo dodatne naloge. Namesto 9 obveznih udeležb je na voljo 9 nalog (3 skupine po 3 naloge), ki jih morajo študenti zagovarjati v predpisanih rokih na laboratorijskih vajah. Vsaka pravilno rešena naloga nadomešča udeležbo na enih laboratorijskih vajah, pri čemer naloga iz določene skupine nadomesti udeležbo na vajah v določenem obdobju. Za študijsko leto 2006/07 so predpisane naslednje naloge iz seznama dodatnih nalog:

  1. skupina: naloge **I.2**, **I.3**, **II.2**, rok za zagovor **13.-17. november 2006**, nadomeščajo udeležbo na vajah med 16. oktobrom in 10. novembrom 2006

  2. skupina: naloge **III.5**, **IV.7**, **V.2**, rok za zagovor **18.-22. december 2006**, nadomeščajo udeležbo na vajah med 13. novembrom in 15. decembrom 2006

  3. skupina: naloge **V.3**,**VI.2**, **VI.3**, rok za zagovor **8.-12. januar 2007**, nadomeščajo udeležbo na vajah med 11. decembrom 2006 in 12. januarjem 2007

- V kolikor asistent oceni, da naloge niso zadovoljivo rešene, lahko od študenta zahteva obvezno udeležbo na vajah.

I. SPLOŠNO - ALGORITMI

1. Romb iz zvezdic

Napišite program, ki najprej prebere podatek o številu vrstic, nato pa izpiše lik naslednje oblike (slika prikazuje primer, ko je število vrstic enako 7):

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

2. Smrekice iz zvezdic

Napišite program, ki najprej prebere podatek o številu vrstic *n*, nato pa izpiše *n* trikotnikov enega za drugim, kot je prikazano na sliki (slika prikazuje primer, ko je prebrano število *n* enako 5):

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

3. Kvadrati števil

Napišite program, ki najprej prebere število vrstic *n* (*n* je pozitivno celo število), nato pa s pomočjo gnezdenja zank izpiše števila v kvadratni obliki tako, kot prikazujejo naslednji primeri:

```
a) n=1
1

b) n=5
12345
6   6
7   7
8   8
12345

c) n=8
12345678
9      9
0      0
1      1
2      2
3      3
4      4
12345678

c) n=11
12345678901
2          2
3          3
4          4
5          5
6          6
7          7
8          8
9          9
0          0
12345678901
```

4. Iskanje praštevil

Charles Babbage je pokazal, da s pomočjo formule x^2+x+41 dobimo nenavadno veliko praštevil (npr. za x=0, dobimo praštevilo 41, za x=1 dobimo praštevilo 43, za x=2 dobimo praštevilo 47 itd.). Napišite program, ki za vse x od 0 do 100 izpiše, ali je vrednost x^2+x+41 praštevilo. Prvih 5 vrstic izpisa naj izgleda tako:

```
0       41 je prastevilo
1       43 je prastevilo
2       47 je prastevilo
3       53 je prastevilo
4       61 je prastevilo
```

Napotek: Najprej sprogramirajte metodo `jePrastevilo()`, ki za argument *n* vrne vrednost `true`, če je *n* praštevilo, in `false`, če *n* ni praštevilo. To funkcijo nato kličite v zanki za vse *x* od 0 do 100.

5. Praštevila

Pruski matematik Christian Goldbach (1690 - 1764) je pokazal, da lahko vsako sodo število *x,* ki je večje od 2, zapišemo kot vsoto dveh praštevil (npr. 4 je vsota praštevil 2+2, 6 je 3+3, 8 je 3+5, 10 je 3+7 ali 5+5, 12 je 5+7 itd.). Napišite program, ki za vse sode *x* med 4 do 100 izpiše vse pare praštevil na naslednji način:

```
4 = 2 + 2
6 = 3 + 3
8 = 3 + 5
10 = 3 + 7 = 5 + 5
12 = 5 + 7
14 = 3 + 11 = 7 + 7
16 = 3 + 13 = 5 + 11
…
22 = 3 + 19 = 5 + 17 = 11 + 11
…
```

Napotek: Uporabite metodo `jePrastevilo()` iz prejšnje naloge. To metodo kličite v zanki za vse sode *x* od 4 do 100.

6. Skupni delitelji

Napišite program, ki izračuna in izpiše skupne delitelje dveh naravnih števil. Za primer, delitelji števila 10 so 2 in 5, delitelji števila 15 so 3 in 5, skupni delitelji obeh števil 10 in 15 pa je tako le 5.

7. Amortizacija osnovnih sredstev

Amortizacijo nekega osnovnega sredstva lahko računamo na tri načine:

a) Nabavno vrednost delimo z življenjsko dobo, tj. s številom let, v katerih se osnovno sredstvo amortizira. V tem primeru se vrednost osnovnega sredstva vsako leto zmanjša za prej omenjeni kvocient.
b) Vrednost osnovnega sredstva vsako leto zmanjšamo za enak delež, ki ga izračunamo kot kvocient 2/*n*, kjer je *n* življenjska doba v letih (npr. če se osnovno sredstvo amortizira v 10 letih, je ta delež 2/10). Delež se računa od trenutne vrednosti osnovnega sredstva, zato je zmanjšanje vrednosti vsako leto drugačno.
c) Vrednost osnovnega sredstva vsako leto zmanjšamo za delež, ki ga za vsako leto posebej izračunamo po formuli

$$\frac{n-i+1}{1+2+3+\ldots+n}$$

kjer je *n* življenjska doba osnovnega sredstva, *i* pa zaporedna številka leta (*i*=1,2,3,...,*n*). Delež se računa od nabavne vrednosti osnovnega sredstva.

Napišite program, ki za neko osnovno sredstvo prebere njegovo nabavno vrednost, življenjsko dobo v letih in način računanja amortizacije (A, B ali C), nato pa za vsako leto "življenja" osnovnega sredstva izpiše zaporedno številko leta, zmanjšanje vrednosti v tistem letu in vrednost osnovnega sredstva na koncu leta.

Primer: Za osnovno sredstvo z nabavno vrednostjo 8000.00 in življenjsko dobo 10 let bi z metodo C dobili naslednji izpis:

```
konec leta 1   zmanjšanje vrednosti: 1454.55 trenutna vrednost: 6545.45
konec leta 2   zmanjšanje vrednosti: 1309.09 trenutna vrednost: 5236.36
konec leta 3   zmanjšanje vrednosti: 1163.64 trenutna vrednost: 4072.72
...
konec leta 10  zmanjšanje vrednosti: 145.45  trenutna vrednost: 0
```

Pozor: Na koncu lahko zaradi zaokrožitve in načina računanja amortizacije dobimo trenutno vrednost tudi različno od 0 !

8. Številske vrste

Napišite program za izračun vsote naslednjih številskih vrst:

$$\ln x = \frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \frac{(x-1)^3}{3x^3} + \frac{(x-1)^4}{4x^4} + \ldots \qquad x > \frac{1}{2}$$

$$(1+x)^{-4} = 1 - \frac{1}{1\cdot 2\cdot 3}\cdot(2\cdot 3\cdot 4\cdot x - 3\cdot 4\cdot 5\cdot x^2 + 4\cdot 5\cdot 6\cdot x^3 - 5\cdot 6\cdot 7\cdot x^4 + \ldots) \qquad |x|<1$$

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \frac{x^7}{7!} + \ldots \qquad |x|<\infty$$

$$\sqrt[4]{(1+x)} = 1 - \frac{(-1)}{4}x + \frac{(-1)\cdot 3}{4\cdot 8}x^2 - \frac{(-1)\cdot 3\cdot 7}{4\cdot 8\cdot 12}x^3 + \frac{(-1)\cdot 3\cdot 7\cdot 11}{4\cdot 8\cdot 12\cdot 16}x^4 - \ldots \qquad |x|\le 1$$

$$\arccos x = \frac{\pi}{2} - \left[x + \frac{1}{2\cdot 3}x^3 + \frac{1\cdot 3}{2\cdot 4\cdot 5}x^5 + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7}x^7 + \frac{1\cdot 3\cdot 5\cdot 7}{2\cdot 4\cdot 6\cdot 8\cdot 9}x^9 + \ldots\right] \qquad |x|<1$$

Napotek: Vsoto številske vrste določite tako, da seštejete vrednosti posameznih členov številske vrste za določen x, pri čemer vsak naslednji člen vsote izračunajte s pomočjo predhodnega člena. S prištevanjem členov vsoti zaključite, ko absolutna vrednost člena pade pod 10^{-5}.

II. RAZREDI IN OBJEKTI

1. Razred Točka

Lego neke točke v dvodimenzionalnem prostoru želimo opisati s pomočjo kartezičnih in polarnih koordinat. Sprogramirajte razred `Tocka` z atributi `x` (koordinata x), `y` (koordinata y), `r` (radij v polarnem zapisu) in `kot` (kot v polarnem zapisu) ter metodami:

- `nastaviX(a)`: nastavi koordinato `x` na `a`,
- `nastaviY(b)`: nastavi koordinato `y` na `b`,
- `nastaviPolarne(c,alfa)`: nastavi vrednosti polarnih koordinat (`r` dobi vrednost `c`, `kot` pa `alfa`),
- `pretvoriVPolarne()`: izračuna atributa `r` in `kot` na podlagi atributov `x` in `y`
- `pretvoriVKartezicne()`: izračuna atributa `x` in `y` na podlagi atributov `r` in `kot`
- `premakni(dx,dy)`: koordinati `x` in `y` se povečata za `dx` in `dy`,
- `zavrti(alfa)`: zavrti točko okrog koordinatnega izhodišča (poveča `kot` za vrednost argumenta `alfa`)
- `povecajRadij(m)`: poveča vrednost polarnega atributa `r` za `m`-krat
- `razdalja(t)`: izračuna razdaljo do točke `t`
- `izpisiKartezicne()`: izpiše vrednosti atributov `x` in `y`
- `izpisiPolarne()`: izpiše vrednosti polarnih koordinat `r` in `kot`

Za generiranje točk sprogramirajte tri konstruktorje:

- prvi naj bo brez parametrov in zgenerira točko z vrednostmi vseh štirih atributov 0
- drugi naj ima dva parametra `a` in `b` in naj zgenerira točko s koodinatama `x=a` in `y=b`, vrednosti `r` in `kot` pa se izračunata.
- tretji naj ima dva parametra `c` in `alfa` in naj zgenerira točko s (polarnimi) koodinatami `r=c` in `kot=alfa`, vrednosti `x` in `y` pa se izračunata.

Za testiranje razreda krog napišite razred `TestTocka`. Z uporabo vsakega konstruktorja definirajte eno točko. Izpišite koordinate vseh točk v obeh načinih, nato pa prvo točko premaknite, drugo zavrtite in tretji povečajte radij. Ob vsaki operaciji poskrbite za konsistentnost koordinat v obeh zapisih. Ponovno izpišite vse točke ter na koncu izpišite še razdalje med vsemi tremi točkami.

2. Razred Pravokotnik

Sprogramirajte razred `Pravokotnik` z atributi `x` (vodoravni odmik gornjega levega oglišča pravokotnika od izhodišča), `y` (navpični odmik gornjega levega oglišča pravokotnika od izhodišča), `a` (širina pravokotnika), `b` (višina pravokotnika), `obseg` in `ploscina` ter metodami `nastaviLego` (vrednosti atributov `x` in `y`), `nastaviVelikost` (vrednosti atributov `a` in `b`), `izracunajObseg`, `izracunajPloscino`, `premakni` (trenutna lega se spremeni za `dx` in `dy`, ki sta podana kot parametra), `izpisiLego`, `izpisiObseg` in `izpisiPloscino`. Za generiranje pravokotnikov sprogramirajte dva konstruktorja: prvi naj bo brez parametrov, drugi pa naj ima 4 parametre: `x`, `y`, `a` in `b`. Ob klicu prvega konstruktorja se generira pravokotnik s stranicama `a=b=1` in gornjim levim ogliščem v točki (0,1), vrednosti obsega in ploščine pa ostaneta nedefinirani. Ob klicu drugega konstruktorja naj se pravilno nastavijo vrednosti vseh atributov: `x`, `y`, `a`, `b`, `obseg` in `ploscina`.

Za testiranje razreda krog napišite razred `TestPravokotnik`. Z uporabo prvega konstruktorja definirajte dva pravokotnika, z uporabo drugega pa še dva. Izpišite lege, obsege in ploščine za vse pravokotnike. Nato spremenite lego in velikost pravokotnikov, jih premaknite in ponovno izpišite vse vrednosti.

3. Razred Avto

Sprogramirajte razred `Avto`, ki predstavlja avtomobil z njegovimi tehničnimi značilnostmi in trenutnim stanjem. Poleg atributa `stevilka`, ki omogoča ločevanje posameznih avtomobilov, so tehnične značilnosti opisane z atributi `najvecjaHitrost` (v km/h), `velikostRezervoarja` (v litrih) in `povprečnaPoraba` (v litrih/100 km). Trenutno stanje avtomobila predstavljajo atributi `avtoVzgan` (tipa boolean), `stevecKm` (npr. 234.142 km), `trenutnaHitrost` (npr. 56.12 km/h) in `kolicinaGoriva` (npr. 14.325 litrov).

Razred naj vsebuje naslednje metode:

- `nastaviStevilko(s)`: nastavi atribut `stevilka` na `s` (recimo 1),
- `nastaviNajvecjoHitrost(h)`: nastavi atribut `najvecjaHitrost` na `h` km/h (recimo 180),
- `nastaviVelRezervoar(l)`: nastavi atribut `velikostRezervoarja` na `l` litrov (recimo 48.0),
- `nastaviPovPorabo(l)`: nastavi atribut `povprecnaPoraba` na `l` litrov/100 km (recimo 7.5),
- `napolniGorivo()`: nastavi atribut `kolicinaGoriva` na vrednost `velikostRezervoarja`,
- `vzgiAvto()`: nastavi atribut `avtoVzgan` na true, pri čemer se porabi 0.05 litra goriva (če ni zadosti goriva se avto ne vžge)
- `ugasniAvto()`: nastavi atribut `avtoVzgan` na false, pri čemer se lahko avto ugasne le, ko je njegova hitrost enaka 0.
- `vozi(t)`: vozi `t` minut s trenutno hitrostjo in vrne koliko poti je v tem času prevozil. Metoda povzroči spremembo atributov `stevecKm` in `kolicinaGoriva`. Če je goriva premalo, naj vožnja poteka tako dolgo, dokler ne zmanjka goriva. Avto lahko vozi le v primeru, da je vžgan (atribut `avtoVzgan` nastavljen na `true`) in ima trenutno hitrost večjo kot 0.
- `spremeniHitrost(h)`: spremeni `trenutnaHitrost` na vrednost `h` km/h. Hitrost avtomobila je med `0` in `najvecjaHitrost`.
- `ustavi()`: spremeni `trenutnaHitrost` na vrednost `0` km/h.
- `najvecjiCas()`: metoda vrne podatek, koliko minut in sekund lahko še vozimo glede na trenutno količino goriva. Trenutno stanje avtomobila se ne spremeni.
- `doseg(l)`: `metoda` vrne podatek, koliko kilometrov lahko prevozimo z največjo hitrostjo, če je v rezervoarju še l litrov goriva. Trenutno stanje avtomobila se ne spremeni.
- `casVoznje(d)`: metoda nam pove vrne koliko časa (v minutah in sekundah) potrebujemo, da prevozimo `d` kilometrov, če vozimo z nespremenjeno hitrostjo, v rezervoarju pa imamo trenutno količino goriva. Pri tem je treba upoštevati, da količina goriva lahko tudi ne zadošča; v tem primeru bo treba ustaviti, ugasniti avtomobil, napolniti rezervoar, ga znova prižgati in odpeljati naprej, kar nam vzame dodatnih 15 minut. Trenutno stanje avtomobila naj se sicer ne spremeni.
- `potovalnaHitrost(d,t)`: metoda vrne hitrost, s katero naj avto vozi, da bi `d` km oddaljen kraj dosegel v `t` minutah. Pri tem naj upošteva tudi eventuelna polnjenja rezervoarja, ki vzamejo dodatnih 15 minut. Trenutno stanje avtomobila se ne spremeni.
- `boljsi(a)`: primerja trenutni avtomobil z avtomobilom `a` in izpiše `stevilko` boljšega avtomobila. Boljši je tisti avtomobil, ki v krajšem času prevozi 1000 km, pri čemer upoštevajte tudi čas polnjenja goriva.
- `izpis()`: izpiše vse atribute avtomobila.

Pri vseh metodah predpostavljamo, da avto pri vseh hitrostih porabi enako količino goriva.

Za generiranje avtomobilov sprogramirajte tri konstruktorje:

- prvi naj bo brez parametrov in zgenerira avtomobil, ki ima vse atribute prazne (vrednost 0 ali `false`)
- drugi naj ima le parameter s in naj zgenerira avtomobil s številko s, ostale vrednosti pa naj bodo prazne.

- tretji naj ima parametre s, `h`, `v` in `p`, zgenerira pa naj avtomobil številke s, z največjo hitrostjo `h`, velikostjo rezervoarja `v` in povprečno porabo `p`. Atributi `stevecKm`, `trenutnaHitrost` in `kolicinaGoriva` naj imajo vrednost 0, `avtoVzgan` pa na `false`.

Za testiranje razreda Avto napišite razred `TestAvto`. Z uporabo vsakega konstruktorja definirajte po en avto. S pomočjo metod za nastavljanje atributov dopolnite podatke za prva dva avtomobila, nato pa izpišite vse podatke o vseh avtomobilih. Z vsemi simulirajte polnjenje goriva, vožnjo, spreminjanje hitrosti in vse informativne izpise. Ponovno izpišite vse avtomobile ter jih primerjajte med seboj.

III. TABELE

1. Ocene študentov

Študent mora v 1. letniku opraviti izpite iz 10 predmetov. Njegove ocene hranimo v dveh tabelah: tabela `izpit` hrani ocene izpitov, tabela `vaje` pa ocene vaj. Če študent še ni polagal izpita pri določenem predmetu, je vpisana ocena 0 (pri izpitih in vajah), v nasprotnem primeru pa ocena zadnjega polaganja.

```
private static byte[] izpit = new byte[10];
private static byte[] vaje = new byte[10];
```

- Napišite metodo `public static void napolniTabeliOcen()`, ki z uporabo generatorja naključnih števil napolni obe tabeli z vrednostmi med 0 in 10. Pri polnjenju bodite pozorni na medsebojno skladnost podatkov v obeh tabelah:
  a) če ima študent oceno izpita 0, mora biti tudi ocena vaj 0 (študent še ni polagal izpita);
  b) če ima študent negativno oceno izpita, mora biti ocena vaj 0 (na naši fakulteti velja pravilo, da se ocena vaj vpiše šele takrat, ko študent opravi izpit);
  c) če ima študent pozitivno oceno izpita, mora biti tudi ocena vaj pozitivna.
- Napišite metodo `public static void analizaOcen()`, ki :
  a) izračuna povprečno oceno izpitov in povprečno oceno vaj, pri čemer upošteva samo pozitivne ocene
  b) izračuna indeks predmetov, kjer je bila dosežena najvišja ocena izpita, najvišja ocena vaj in najvišja skupna ocena izpita in vaj
  c) izračuna standardno deviacijo ocen izpitov in vaj, upoštevajoč samo pozitivne ocene. Formula za izračun standardne deviacije je naslednja:

$$\bar{x} = \frac{\sum x_i}{n} \qquad i = 0,1,\cdots,n-1$$

$$\sigma = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}} \qquad i = 0,1,\cdots,n-1$$

  Izračunane podatke vpišite v tabele `povOc`, `maxOc in std` tako, da velja naslednje:

  - `povOc[0]` vsebuje povprečno oceno izpitov;
  - `povOc[1]` vsebuje povprečno oceno vaj;

  - `maxOc[0]` vsebuje indeks predmeta, kjer je študent dosegel najvišjo oceno izpita;
  - `maxOc[1]` vsebuje indeks predmeta, kjer je študent dosegel najvišjo oceno vaj;
  - `maxOc[2]` vsebuje indeks predmeta, kjer je študent dosegel najvišjo skupno oceno izpita in vaj;

  - `std[0]` vsebuje standardno deviacijo za ocene izpitov;
  - `std[1]` vsebuje standardno deviacijo za ocene vaj;

- Napišite metodo `public static void izpisiRezultate()`, ki izpiše vrednosti tabel `povOc`, `maxOc` in `std`.

Vse tabele deklarirajte kot statične spremenljivke na ravni razreda. Delovanje metod preverite z ustreznim zaporedjem klicev metod v metodi `main`.

2. Kozarci marmelade

V neki tovarni polnijo marmelado v 700-gramske kozarce. Delovanje polnilne naprave preverjajo tako, da v vsaki izmeni (tovarna dela v treh izmenah) naključno izberejo 10 kozarcev in jih stehtajo. Podatke,

ki jih dobijo pri tehtanju, zabeležijo v dvodimenzionalno tabelo `vzorec`, ki ima 3 vrstice (vsaka vrstica predstavlja eno izmeno) in 10 stolpcev (vsak stolpec ustreza enemu vzorcu).

```
private static int[][] vzorec = new int[3][10];
```

- Napišite metodo `public static void napolniVzorce()`, ki z uporabo generatorja naključnih števil napolni tabelo `vzorec` z vrednostmi med 670 in 730.
- Napišite metodo `public static void analizaVzorcev()`, ki za vsako izmeno posebej:
  d) izračuna povprečno maso vseh 10 naključno izbranih kozarcev
  e) poišče indeks tistega kozarca, ki najbolj odstopa od predpisane mase 700 gramov.

  Izračunane podatke vpišite v tabeli `pv` in `ik` tako, da velja naslednje:

  - `pv[0]` vsebuje povprečno maso kozarcev, odvzetih v prvi izmeni;
  - `pv[1]` vsebuje povprečno maso kozarcev, odvzetih v drugi izmeni;
  - `pv[2]` vsebuje povprečno maso kozarcev, odvzetih v tretji izmeni.

  - `ik[0]` vsebuje indeks tistega kozarca iz prve izmene, ki najbolj odstopa od predpisane mase 700 g;
  - `ik[1]` vsebuje indeks tistega kozarca iz druge izmene, ki najbolj odstopa od predpisane mase 700 g;
  - `ik[2]` vsebuje indeks tistega kozarca iz tretje izmene, ki najbolj odstopa od predpisane mase 700 g;

- Napišite metodo `public static void izpisiAnalizo()`, ki izpiše vrednosti tabel `pv` in `ik`.

Upoštevajte, da so vse tabele deklarirane na ravni razreda. Delovanje metod preverite z ustreznim zaporedjem klicev metod v metodi `main`.

3. Trgovine

V podjetju, ki ima 7 trgovin, spremljajo prodajo po posameznih trgovinah za obdobje petih let. Podatki o prodaji (tj. vrednost vseh prodanih izdelkov v milijonih SIT) so shranjeni v dvodimenzionalni tabeli `prodaja`, ki ima 7 vrstic (vsaka vrstica ustreza eni trgovini) in 5 stolpcev (vsak stolpec ustreza enemu letu).

Napišite razred Trgovine, v katerem je tabela prodaja deklarirana kot statična spremenljivka razreda:

```
private static int[][] prodaja = new int[7][5];
```

- Razred naj vsebuje metodo `static void napolniProdajo()`, ki z uporabo generatorja naključnih števil napolni tabelo z vrednostmi med 10 in 1000.
- Napišite metodo `static void izpisiVsoProdajo()`, ki za vsako trgovino posebej izpiše: vrednost prodanih izdelkov v vsakem letu, nato pa še vsoto vrednosti prodaje trgovine v vseh petih letih skupaj.
- Napišite metodo `static void letnaPovprecja()`, ki za vsako leto izpiše vsoto in povprečno vrednost prodaje vseh trgovin skupaj.
- Napišite metodo `static int najTrgovinaLeta(int leto)`, ki vrne indeks trgovine, ki je v letu z indeksom `leto` imela največjo vrednost prodaje. Če je takih trgovin več, vrnite indeks prve izmed njih.
- Napišite metodo `static int[] najProdajaLeta()`, ki vrne celoštevilsko tabelo z dvema elementoma: prvi predstavlja indeks trgovine, drugi pa indeks leta, v katerem je bila dosežena največja prodaja nasploh. Rešitev naj uporablja metodo `najTrgovinaLeta()`. Če je rešitev več, vrnite indeks zadnjega leta.

Delovanje vseh metod preverite z ustreznim zaporedjem klicev metod v metodi `main`. Pri zadnjih dveh metodah poskrbite za izpis vrnjenih vrednosti, pri predzadnji pa še za ustrezen vnos podatkov (indeks leta).

4. Trgovsko podjetje

Neko trgovsko podjetje ima 10 trgovin. V tabeli `promet` so shranjeni podatki o vrednosti prodanega blaga za vsak dan preteklega leta. Tabela je tridimenzionalna: prva dimenzija predstavlja 10 trgovin, druga 12 mesecev in tretja 31 dni v vsakem mesecu (če ima mesec manj kot 31 dni, imajo odvečni elementi tabele vrednost 0).

- Napišite metodo `napolniPodatke`, ki z uporabo generatorja naključnih števil napolni tabelo `promet` z vrednostmi med 10 000 in 10 000 000. Bodite pozorni na elemente tabele, ki morajo biti 0.
- Napišite metodo `vsotaPoTrg`, ki za vsako trgovino posebej izračuna, koliko znaša vrednost prodanega blaga v celem letu, tako da sešteje podatke po dnevih in mesecih ter dobljeno vsoto vpiše v tabelo `letPrometTrg`.
- Napišite metodo `izpisiPromet`, ki izpiše vrednosti tabele `letPrometTrg`.

Delovanje metod preverite s pomočjo razreda `TestTrgPodjetje`. Upoštevajte naslednje deklaracije:

```
class TrgPodjetje
{
  public static void napolniPodatke (int[][][] promet);
  public static void vsotaPoTrg (int[][][] promet, int[] letPrometTrg);
  public static void izpisiPromet (int[] letPrometTrg);
}
class TestTrgPodjetje
{
  final int ST_TRG=10;
  private int[][][] promet = new int[ST_TRG][12][31];
  private int[] letPrometTrg = new int[ST_TRG];
}
```

5. Temperature

Napišite razred Temperature, ki omogoča statistično obdelavo podatkov o povprečnih mesečnih temperaturah zadnjih desetih let v osmih slovenskih krajih. Podatke hranimo v tridimenzionalni tabeli, kjer prva dimenzija predstavlja 8 krajev, druga 10 let in tretja 12 mesecev vsakega leta.

- Napišite metodo `static void napolniTemperature(double[][][] t)`, ki z uporabo generatorja naključnih števil napolni tabelo prispevki z ustreznimi decimalnimimi števili. Da bi bile generirane vrednosti kar najbolj realne upoštevajte, da so povprečne temperature:
  - pozimi (december, januar in februar) med -20.0 in +5.0 C;
  - spomladi (marec, april in maj) med 0.0 in 25.0 C in naraščajo – marec ima najmanj, maj največ;
  - poleti (junij, julij in avgust) med 20.0 in 35.0 C;
  - jeseni (september, oktober in november) med 0.0 in 25.0 C in padajo – september ima največ, november najmanj;

- Napišite metodo `static void izpisiKrajevneTemperature(double[][][] t, int k, int l)`, ki za kraj z indeksom `k` izpiše za leto `l` vse mesečne temperature in celoletno povprečje (celoletno povprečje dobimo tako, da seštejemo povprečne temperature po posameznih mesecih in jih delimo z 12).
- Napišite metodo `static void izpisiTemperaturePoLetih(double[][][] t)`, ki za vsako leto izpiše celoletno povprečje za vsak kraj posebej in za vse kraje skupaj (povprečna vrednost vseh temperatur v vseh mesecih in krajih določenega leta).

- Napišite metodo `static void triNajnizjeTemperature(double[][][] t)`, ki izpiše tri kombinacije indeksov kraja, leta in meseca, kjer so bile dosežene tri najnižje mesečne temperature nasploh. V primeru, da je več enakovrednih rezultatov, izpišite vse.

Delovanje vseh metod preverite z ustreznim zaporedjem klicev metod v metodi `main`. Upoštevajte, da tabela temperatur **ni** statična spremenljivka razreda. V metodi `main` poskrbite tudi za ustrezen vnos podatkov (indeks kraja in leta).

6. Znanstvena konferenca

Na znanstveni konferenci bodo avtorji iz 25 držav EU predstavili tri vrste prispevkov (referate, kratke referate in posterje), ki sodijo v pet različnih področij. V tridimenzionalni tabeli `prispevki` so shranjeni podatki o številu prispevkov posamezne vrste po državah in posameznih področjih: prva dimenzija predstavlja 25 držav, druga tri vrste prispevkov in tretja pet znanstvenih področij.

- Napišite metodo `napolniTabelo`(), ki z uporabo generatorja naključnih števil napolni tabelo `prispevki` z vrednostmi med 0 in 20.

```
public static void napolniTabelo(int[][][] prispevki)
```

- Napišite metodo `skupajPrispevkov`, ki izpiše skupno število vseh prispevkov na konferenci

```
public static void SkupajPrispevkov(int[][][] prispevki)
```

- Napišite metodo `prispevkovPoVrsti`, ki izpiše skupno število prispevkov vsake vrste.

```
public static void prispevkovPoVrsti(int[][][] prispevki)
```

- Napišite metodo `podrocjeZNajvecPrispevki`, ki izpiše zaporedno številko (tj. indeks) tistega področja, za katero bo predstavljenih največ prispevkov. Če je takšnih področij več, naj metoda izpiše zaporedne številke vseh področij.

```
public static void podrocjeZNajvecPrispevki(int[][][] prispevki)
```

IV. NIZI

1. Analiza nizov

Napišite razred `AnalizaNizov`, ki se obnaša kot knjižnica metod, s katerimi lahko analiziramo vsebino poljubnega niza. Razred naj vsebuje naslednje metode:

- Metoda `public static int steviloMalihCrk(String niz)` vrne število malih črk, ki jih vsebuje `niz`.
- Metoda `public static int steviloZnakov(String niz, char c)` vrne število pojavitev znaka `c` v nizu `niz`.
- Metoda `public static int steviloBesed(String niz)` vrne število besed, ki jih vsebuje `niz`. Pri tem predpostavite, da se lahko vsaka beseda začne s črko ali številko (in vsebuje poljubno število črk in številk), ločilo med besedami pa so beli presledki. Med bele presledke (angleško *white space*) štejemo znak za presledek (`' '`), znak za tabulator (`'\t'`), znak za novo vrstico (`'\n'`), znak za prehod na začetek vrstice (angl. *carriage return*, `'\r'`) ter znak za pomik v novo vrsto (angl. *line feed*, `'\f'`).
- Metoda `public static String besedaNaMestu(String niz, int m)` vrne besedo, ki se nahaja na `m`-tem mestu v `niz`. Če je `m` večji od števila besed, metoda vrne prazen niz.
- Metoda `public static int steviloEnakihZnakov(String niz)` vrne največje število enakih (katerihkoli) malih črk (angleške abecede) v nizu `niz`.
- Metoda `public static String besedaZNajvecEnakimiZnaki(String niz)` vrne besedo iz niza `niz`, ki vsebuje največ enakih (katerihkoli) malih črk (angleške abecede). Če je takih besed več, naj metoda vrne prvo izmed njih. Pri rešitvi si pomagajte (tudi) z ustreznimi klici drugih metod razreda `AnalizaNizov`.

Delovanje vseh metod preverite z ustreznim zaporedjem klicev metod v metodi `main`. Pri tem poskrbite za ustrezen vnos niza in drugih potrebnih podatkov.

2. Kodiranje nizov

Napišite metodi `Kodiraj` in `Dekodiraj`, ki omogočata kodiranje niza tipa `String` v število tipa `long` in obratno. Pri tem naj velja:

- niz sestavljajo samo črke iz angleške abecede (26 znakov) in številke (10 znakov)
- v nizu ne razlikujemo med velikimi in malimi črkami
- nizi so lahko dolgi največ 10 znakov

Pri kodiranju si pomagamo s kodirno tabelo, kjer vsakemu izmed 36 znakov pripada ena vrednost (recimo a-ju pripada vrednost 1, b-ju vrednost 2, itd.). Vrednost niza izračunamo tako, da vrednost prvega znaka v nizu pomnožimo s 37, temu prištejemo vrednost drugega znaka in vse skupaj pomnožimo s 37, nato dodamo vrednost tretjega znaka in tako naprej do zadnjega znaka.

Primer:

A,a	B,b	C,c	D,d	...	N,n	...	R,r	...	0
1	2	3	4		14		18		27

ABRAN0 = 1 | 2 | 18 | 1 | 14 | 27 = ((((**1***37+**2**)*37+**18**)*37+**1**)*37+**14**)*37+**27** = **74005947**

Pri dekodiranju ravnamo obratno: ostanek pri deljenju s 37 nam pove kodo zadnjega znaka, nato pa kodo postopoma celoštevilsko delimo in določamo ostanke. S postopkom končamo takrat, ko je rezultat deljenja s 37 enak 0.

Primer:

74005947 % 37 = **27**	»**0**«
74005947 / 37 = 2000160, 2000160 % 37 = **14**	»**N**«
2000160 / 37 = 54058, 54058 % 37 = **1**	»**A**«
54058 / 37 = 1461, 1461 % 37 = **18**	»**R**«
1461 / 37 = 39, 39 % 37 = **2**	»**B**«
39 / 37 = 1, 1 % 37 = **1**	»**A**«
1 / 37 = **0**	

3. Ugibanje pregovorov

```
public class Pregovor
{
  public static void main(String[] args) throws Exception
  {
    String iskaniNiz="Rana ura, zlata ura.";
    String prikazaniNiz="**** ***, ***** ***.";
    char crka;      // vtipkana crka
    int poz;        // pozicija vtipkane crke v iskaniNiz
    System.out.println(prikazaniNiz);
    while (prikazaniNiz.indexOf('*') != -1)
    {
      System.out.print("Vtipkaj crko:");
      crka=(char)System.in.read();
      System.in.read(); System.in.read(); // Enter
      poz=iskaniNiz.indexOf(crka);
      if (poz == -1)
        System.out.println("Te crke ni, ugibaj ponovno!");
      else
      {
        do
        {
          prikazaniNiz=prikazaniNiz.substring(0,poz)+crka+
            prikazaniNiz.substring(poz+1,prikazaniNiz.length());
          poz=iskaniNiz.indexOf(crka,poz+1);
        } while (poz != -1);
        System.out.println(prikazaniNiz);
      }
    }
  }
}
```

Podan program za ugibanje pregovora popravite tako, da bo ustrezal naslednjim zahtevam:

- Pregovor, ki ga je treba uganiti, naj bo podan kot argument metode `main()`.
- Spremenljivka `prikazaniNiz` mora biti tipa `StringBuffer`.
- Začetno vrednost spremenljivke `prikazaniNiz` morate določiti tako, da v zanki pregledate vse znake pregovora (tj. iskanega niza) in tiste znake, ki predstavljajo črke, zamenjate z zvezdico, ločila in presledke pa pustite nespremenjene.
- Nove vrednosti `prikazaniNiz` ne smete več tvoriti s konkatenacijo (uporabite ustrezne metode razreda `StringBuffer`).

4. Popačena govorica

V tabeli `nizi`, ki jo morate kreirate sami, naj bo shranjenih 10 nizov tipa `StringBuffer`, ki jih želimo popačiti s pomočjo metode

```
public static void popaciNiz(StringBuffer enNiz)
```

ki niz `enNiz` (`enNiz` je vhodno-izhodni parameter) preoblikuje tako, da v vsaki besedi (besede so med seboj ločene z enim ali več presledki):

a) odvzame prvo črko in jo prestavi na konec besede
b) na koncu tako preoblikovane besede doda še črko `'a'`.

Primer: Niz z besedilom "`Java je objektno usmerjen programski jezik`" se pretvori v "`avaJa eja bjektnooa smerjenua rogramskipa ezikja`"

5. Šifriranje nizov

Napišite metodo `public static void sifrirajNiz(StringBuffer n)`, ki niz `n` (`n` je vhodno-izhodni parameter) zašifrira na naslednji način:

- če je prvi znak niza črka `'o'` ali `'O'`, mora preostanek niza "obrniti" tako, da se bo pravilno bral z desne proti levi
- če je prvi znak niza črka `'n'` ali `'N'`, mora preostale znake v nizu nadomestiti z njihovimi nasledniki (naslednik je znak, katerega numerična koda je za 1 večja)
- če je prvi znak niza črka `'p'` ali `'P'`, mora preostale znake v nizu nadomestiti z njihovimi predhodniki (predhodnik je znak, katerega numerična koda je za 1 manjša)

Poleg tega mora metoda nastaviti prvi znak vsakega zašifriranega niza tako, da bo možno ponovno dešifriranje (tj. pretvorba niza v prvotno obliko).

Primer:

a) niz "`oobrni`" se pretvori v "`oinrbo`"
b) niz "`nnaslednji`" se pretvori v "`pobtmfeokj`"
c) niz "`ppredhodni`" se pretvori v "`noqdcgncmh`"

Napotek: Upoštevajte dejstvo, da je vsak znak interno predstavljen s svojo numerično kodo. Zato lahko s pomočjo konverzije tipa znak `ch` pretvorimo v celo število (npr. `int num=(int) ch;`) in obratno (npr. `char ch=(char) num;`).

Delovanje metode preverite z ustreznimi klici metode v metodi `main`.

6. Kodiranje

Napišite razred `Kodiranje`, ki vsebuje metodi

```
public static StringBuffer kodiraj(String n)
public static String dekodiraj(StringBuffer n)
```

Metoda `kodiraj` naj niz `n` pretvori v niz tipa `StringBuffer` tako, da vsako črko angleške abecede nadomesti z nizom, ki predstavlja zaporedno številko velike črke v angleški abecedi, vse ostale znake (tudi številke) pa naj ohrani nespremenjene. Da bi omogočili enolično dekodiranje črk, je treba 'A' in 'a' nadomestiti z "01" ter 'B' in 'b' z "02", medtem ko ostale črke lahko nadomestimo z eno ali dvema ciframa ('C' in 'c' s "3", 'D' in 'd' s "4", ... ter 'Z' in 'z' s "26").

Metoda `dekodiraj` pa naj tako zakodiran niz pretvori spet v prvotno obliko. V primeru klica metode `dekodiraj` s pokvarjenim zakodiranim nizom, naj ta metoda javi mesto napake v kodiranem nizu (primer pokvarjenega niza je recimo niz "03").

Primer: Metoda `kodiraj` pretvori niz "RACUNALNIK JE ZAKON !" v "18013211401121491 1 105 26011115 14 !", metoda dekodiraj pa ta niz pretvori ponovno v "RACUNALNIK JE ZAKON !".

Delovanje obeh metod preverite tako, da v metodi `main` zaporedoma vnašate (poljubno število) nizov, za katere program najprej izpiše njihovo kodo, nato pa še niz, ki nastane z dekodiranjem kode. Postopek kodiranja naj se zaključi takrat, ko vnesete prazen niz.

7. Stiskanje nizov

Nize, v katerih se neka črka večkrat ponovi, lahko zapišemo krajše na naslednji način:

`AAA`	zapišemo kot	`A3`
`ABBCD`	zapišemo kot	`AB2CD`
`AAAAEEEEEEEEEEU`	zapišemo kot	`A4E11U`

Kot je razvidno iz primera, stisnjena beseda ni nikoli daljša od originala, saj število zaporednih nastopanj vsake črke dodamo samo v primeru, ko se črka dejansko ponavlja.

Napišite program, ki bo omogočal stiskanje in ponovno razširjanje besed. Program naj bo omejen na besede, ki so sestavljene samo iz črk (a-z in A-Z), upoštevajte pa, da je število pojavitev lahko tudi večmestno število.

Program naj vsebuje tako metodo za stiskanje kot metodo za razširjanje besed. V njem najprej preberemo tabelo `t1` z 10 besedami, besede stisnemo in zapišemo v tabelo `t2` ter stisnjene besede zopet razširimo in zapišemo v tabelo `t3`. Program naj na koncu izpiše vse tri tabele.

V. TABELE OBJEKTOV, SORTIRANJE TABEL

1. Planinci

Razred `Planinec` je namenjen hranjenju podatkov o članih planinskega društva. Obsega številko, priimek in ime planinca, leto prvega vpisa in dvodimenzionalno tabelo, v kateri so v časovnem zaporedju shranjeni podatki o njihovih planinskih podvigih v Sloveniji: prva dimenzija predstavlja zaporedna leta od vpisa, druga število osvojenih vrhov, vrednost sama pa višino določenega vrha v metrih. Ker je najstarejši planinec vpisan v društvu 30 let, je 30 tudi velikost prve dimenzije, medtem ko je število osvojenih vrhov v določenem letu omejeno na 20.

```
class Planinec
{
  private int stevilka;
  private String ime,priimek;
  private int letoVpisa;
  private int[][] podvigi = new int[30][20];
}
```

Primer:

```
stevilka: 4
ime: Janez
priimek: Novak
letoVpisa: 2004
podvigi = {{890,1250,2060,2369,1024,2864,1028},{1290,1462}}
```

Sprogramirajte razred `TestPlaninec`, v katerem naj bo deklarirana tabela s podatki za 10 planincev:

```
class TestPlaninec
{
  final int ST_PL=10;
  Planinec[] planinci = new Planinec[ST_PL];
}
```

Razred `TestPlaninec` naj:

- Omogoča vnos oziroma naključno generiranje podatkov v tabeli `planinci`. Številka planinca naj bo kar za ena povečan indeks iz tabele `planinci`, priimek in ime naj se vneseta ročno, medtem ko se ostali podatki generirajo naključno:
  - leto prvega vpisa kot naključna vrednost med 1976 in 2005,
  - število osvojenih vrhov kot naključna vrednost med 0 in 20,
  - višine posameznih vrhov kot naključna vrednost med 800 in 2864.
- Za vsakega planinca urejeno izpiše podatke o vseh njegovih podvigih (po letih), skupaj z izpisom števila osvojenih vrhov po letih in skupaj.

  Primer:

  ```
  4 Janez Novak, leto vpisa: 2004, skupaj vrhov: 9
  2004 - Stevilo vrhov: 7 ; 890 m, 1250 m, 2060 m, 2369 m, 1024 m,
  2864 m,1028 m
  2005 - Stevilo vrhov: 2 ; 1290 m, 1462 m
  ```

- Za vsakega planinca izpiše povprečno višino osvojenih vrhov v vseh letih.
- Za vsako koledarsko leto na intervalu od `zacLeto` do `konLeto` naj izpiše najvišji in najnižji osvojeni vrh v letu (za vse planince) ter povprečno število osvojenih vrhov na planinca na izbranem intervalu.
- Izpiše urejen seznam planincev glede na najvišji osvojeni vrh. Na prvem mestu naj bo planinec, ki je osvojil najvišji vrh, na zadnjem pa tisti, ki je osvojil najnižjega (oziroma nobenega).

2. Dohodnina

Podatki o dohodnini za 10 davčnih zavezancev davčne uprave so shranjeni v tabeli `zavezanci`. Za vsakega zavezanca obsegajo enotno matično številko občana (EMŠO), priimek, ime, višine odmerjenih dohodnin in višine plačanih akontacij za zadnjih 8 let (od leta 1998 do 2005). Pri tem veljajo naslednje deklaracije:

```
class Zavezanec
{
  private int emso;
  private String ime;
  private String priimek;
  private int[] dohodnina = new int[8];
  private int[] akontacija = new int[8];
}

class TestZavezanec
{
  final int ST_ZAV=10;
  Zavezanec[] zavezanci = new Zavezanec[ST_ZAV];
}
```

Razlika med odmerjeno dohodnino in plačano akontacijo v določenem letu predstavlja doplačilo, ki ga mora zavezanec doplačati davčni upravi v tem letu. Če je ta razlika negativna, govorimo o vračilu, saj davčna uprava tak znesek vrne zavezancu. Napišite program, ki bo:

- omogočal vnos vseh podatkov v tabelo `zavezanci`. Del podatkov (EMŠO, priimek in ime) vnesite ročno, medtem ko elemete tabel tvorite z generatorjem naključnih števil (območje med 500 000 in 10 000 000);
- izpisal urejen seznam zavezancev glede na skupno vsoto doplačil (oziroma vračil) dohodnine. Na prvem mestu naj bo zavezanec, ki je (v vseh letih skupaj) največ doplačal, na zadnjem pa tisti, ki je najmanj doplačal oziroma je največ denarja dobil nazaj.
- izpisal urejen seznam letnic (od 1998 do 2005) glede na skupen znesek doplačil (oziroma vračil) vseh zavezancev v določenem letu. Na prvem mestu naj bo leto, v katerem je bil skupni znesek doplačil največji (oziroma najmanjši znesek vračil), na zadnjem pa tisto, ko je bil največji znesek vračil (oziroma najmanjši znesek doplačil).

3. Telefonski impulzi

Razred `Uporabnik` je namenjen hranjenju podatkov o telefonskem priključku vsakega uporabnika v podjetju. Obsega priimek in ime uporabnika in številko oddelka, kjer je zaposlen, poleg tega pa je v komponenti `stImpulzov` shranjeno število porabljenih impulzov v posameznih mesecih preteklega leta.

```
class Uporabnik
{
  private String ime,priimek;
  private int stOddelka;
  private int[] stImpulzov = new int[12];
}
```

Sprogramirajte razred `TestUporabnik`, v katerem naj bo deklarirana tabela s podatki za 10 uporabnikov:

```
class TestUporabnik
{
  final int STUPOR=10;
  Uporabnik[] uporabniki = new Uporabnik[STUPOR];
}
```

Razred `TestUporabnik` naj:

- omogoča vnos vseh podatkov v tabelo `uporabniki`. Tabela mora biti urejena po komponenti `stOddleka` (oddelki so oštevilčeni z zaporednimi številkami od 1 dalje), tako da se podatki o uporabnikih iz istega oddelka nahajajo skupaj (sortiranje med vnosom podatkov). Del podatkov (priimek, ime, številka oddelka) vnesite ročno, medtem ko lahko elemente iz tabele `stImpulzov` tvorite z generatorjem naključnih števil (vrednosti med 0 in 1000).
- za vsak oddelek posebej izračuna in izpiše povprečno število porabljenih impulzov v preteklem letu. Nalogo morate rešiti z enim samim prehodom skozi tabelo, upoštevajoč, da je tabela urejena po številkah oddelkov.

- izpiše urejen seznam uporabnikov glede na porabljeno število impulzov. Na prvem mestu naj bo uporabnik, ki je v celem letu porabil največ impulzov, na zadnjem pa tisti, ki je jih je porabil najmanj.

VI. ABSTRAKTNI RAZREDI

1. Geometrijska telesa

Podan je abstraktni razred `GeometrijskaTelesa`, iz katerega lahko izpeljemo podrazrede za različna geometrijska telesa (npr. kvadre, krogle, valje ipd.). Za vsako telo želimo izračunati volumen in površino, zato sta v razredu `GeometrijskaTelesa` deklarirani abstraktni metodi `volumen` in `povrsina`.

```
public abstract class GeometrijskaTelesa
{
  public abstract double volumen();
  public abstract double povrsina();
}
```

a) Sprogramirajte podrazreda `Kvadri` in `Valji`, v katerih naj bodo za vsako vrsto teles:
- deklarirani atributi, ki jih potrebujemo za izračun volumna in površine;
- sprogramiran konstruktor z ustreznimi argumenti;
- redefinirani metodi `volumen` in `povrsina`;
- redefinirana metoda `toString()` za izpis vseh podatkov o telesu.

b) Sprogramirajte podrazred `Kocke`, ki je podrazred razreda `Kvadri`. Ta naj vsebuje ustrezen konstruktor ter po potrebi ustrezno redefinirane metode.

c) Sprogramirajte razred `TabelaGeometrijskihTeles`, v katerem naj bo deklarirana tabela `t` z različnimi geometrijskimi telesi. Razred naj:

- napolni tabelo `t` s podatki o različnih telesih;
- poišče telo z največjim volumnom in izpiše vse njegove atribute, njegov volumen in površino.

Za izhodišče uporabite naslednje deklaracije:

```
public class TabelaGeometrijskihTeles
{

  public static void main(String[] args)
  {
    GeometrijskaTelesa[] t=new GeometrijskaTelesa[10];

  }

}
```

2. Atleti

Podan je abstraktni razred `Atlet`, ki predstavlja osnovo za izdelavo programov, s pomočjo katerih lahko obdelujemo rezultate nekega atletskega tekmovanja.

```
public abstract class Atlet
{
  private int startnaStev;    // startna številka
  private String priimek;     // priimek tekmovalca
  private String ime;         // ime tekmovalca
  private String drzava;      // ime države, iz katere prihaja

  public Atlet(int st, String p, String i, String d)
  {
    startnaStev=st;
    priimek=p;
    ime=i;
    drzava=d;
  }

  public abstract void vnesiRezultat();
```

```
}
```

Postopek za vnos rezultata, ki ga je dosegel nek tekmovalec, je odvisen od discipline, v kateri je nastopil. Tako je pri metalcih (krogle, diska, kladiva, kopja) potrebno vnesti ime discipline (npr. "Met kopja") in dolžine posameznih metov (vsak metalec ima na razpolago 6 metov). Pri tekačih pa je potrebno vnesti razdaljo v metrih (npr. 100, 200, 400, 800, 1500, 5000 ali 10000) in čas (lahko kot štiri ločene celoštevilske vrednosti za ure, minute, sekunde in stotinke), ki ga je dosegel.
Sprogramirajte:

a) Podrazred `Metalec`, ki bo vseboval:
   - deklaracije dodatnih atributov, ki so specifični za metalce,
   - konstruktor, ki bo generiral nov objekt tipa `Metalec` z vsemi podatki razen dolžin posameznih metov (le-te se vpišejo kasneje z metodo `vnesiRezultat`),
   - redefinicijo metode za vnos rezultatov,
   - redefinicijo metode `toString()`, ki vrne niz z vsemi podatki o metalcu.

b) Podrazred `Tekac`, ki bo vseboval:
   - deklaracije dodatnih atributov, ki so specifični za tekače,
   - konstruktor, ki bo generiral nov objekt tipa `Tekac` z vsemi podatki razen časa, ki ga je dosegel (le-ta se vpiše kasneje z metodo `vnesiRezultat`),
   - redefinicijo metode za vnos rezultatov,
   - redefinicijo metode `toString()`, ki vrne niz z vsemi podatki o tekaču.

V posebnem razredu `TestAtletov` vnesite podatke za več metalcev in tekačev (recimo 10) in jih shranite v tabelo objektov tipa `Atlet`. Nato z metodo `toString` izpišite vse atlete tako, da se poleg štartne številke, imena, priimka in države pri metalcih izpiše še ime discipline in vsi meti, pri tekačih pa razdalja in doseženi čas.

3. Funkcije

Abstraktni razred `Funkcija` predstavlja osnovo za različne razrede, s katerimi realiziramo različne matematične funkcije ene spremenljivke *f(*x):

```
public abstract class Funkcija
{

  public abstract double vrednost(double x);

  public double enaNicla(double x1, double x2)
  {
    // to metodo morate sprogramirati
  }

}
```

Metoda `vrednost` izračuna **vrednost funkcije** v točki `x` – *f(x)*, metoda `enaNicla` pa **eno ničlo** (vrednost `x`, kjer je vrednost `f(x)=0`), ki se nahaja med točkama `x1` in `x2`.

Dopolnite zgornjo deklaracijo razreda `Funkcija` z metodo `enaNicla`, ki poišče (po metodi bisekcije) eno izmed ničel, ki se nahajajo med točkama `x1` in `x2` na naslednji način:

a. Parametra `x1` in `x2` morata biti izbrana tako, da je v eni izmed teh dveh točk funkcijska vrednost pozitivna, v drugi pa negativna. V kolikor ni tako, ugotovite napako.
b. Interval `(x1,x2)` razpolovimo tako, da izračunamo `x=(x1+x2)/2`.
c. Izračunano srednjo vrednost uporabimo namesto enega od prejšnjih približkov:
   - Če imata vrednosti polinoma v točkah `x` in `x1` enak predznak, nadaljujemo z iskanjem ničle na intervalu `(x,x2)`, torej `x1=x`;
   - V nasprotnem primeru nadaljujemo z iskanjem ničle na intervalu `(x1,x)`, torej `x2=x`.
d. Koraka (ii) in (iii) ponavljamo, dokler ne dosežemo predpisane natančnosti (dokler velikost intervala ne pade pod 10^{-4}).

Napišite razred `Polinom`, ki je podrazred razreda `Funkcija` in služi za predstavitev polinomov. Vsak objekt tega razreda naj bo definiran s tabelo realnih števil `a`, ki predstavljajo koeficiente polinoma:

$$p(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_{n-1} x + a_n$$

V razredu `Polinom` sprogramirajte ustrezen konstruktor in redefinirajte metodo `vrednost`, tako da vrne vrednost polinoma v točki *x*. Za izračun vrednosti polinoma uporabite Hornerjevo shemo, po kateri lahko polinom zapišemo na naslednji način:

$$p(x) = (\ldots((a_0 x + a_1)x + a_2)x + \ldots + a_{n-1})x + a_n$$

Napišite razred `MExpK`, ki je tudi podrazred razreda `Funkcija` in predstavlja funkcijo me^x-*k*. Vsak objekt tega razreda naj bo definiran z realnima številoma `m` in `k`. V razredu `MExpK` sprogramirajte ustrezen konstruktor in redefinirajte metodo `vrednost`. Vrednost funkcije izračunate z vrsto.

$$me^x = m * \left(1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots\right) - k \quad | x | < \infty, m * k \geq 0$$

Napišite testni razred `TestFun`, v katerem v metodi `main` s pomočjo generatorja naključnih števil tvorite 10 funkcij (nekaj polinomov in nekaj eksponentnih funkcij). Za vsako od njih izpišite njihove ničle na ustreznih intervalih.