Harjutustund 3

Tsüklidirektiivid. Klass Math. Staatilised meetodid. Massiivid.

Tsüklidirektiivid

Vaadake teooriat eelmisest praktikumist.

Ülesanne 1

Koostada programm, mis leiab esimeste 20 arvude summat (1 kuni 20).

Ülesanne 2

Koostada programm, mis leiab positiivsete paarisarvude summat vahemikus 1 kuni 20.

Ülesanne 3

Koostada programm, mis leiab esimeste positiivsete paaritute arvude summat vahemikus 1 kuni 20.

Ülesanne 4

Koostada programm, mis leiab naturaalarvu numbrite summat (kui arv on 327, siis tulemus on 3+2+7=12).

Klass Math

Kui on vaja ümardada, siis saame kasutada meetodit `Math.round`. Näiteks `Math.round(0.7)` või `Math.round(m1)`, kui `m1` on enne väärtustatud. Klass `Math` on juba valmis tehtud ja vajadusel saame selle klassi meetodeid kasutada. Näiteks juhusliku reaalarvu leidmiseks võib kasutada meetodit `Math.random`, mis tagastab (pseudo)juhusliku `double`-tüüpi arvu poollõigust [0,0; 1,0) (st 0 on kaasa arvatud, 1 aga mitte). Omistamisdirektiivi

```
double randomNumber = Math.random();
```

täitmisel omandab muutuja `randomNumber` konkreetse väärtuse, mida me ette ei tea. Kui on vaja juhuslikku arvu mingis teises poollõigus, siis saab selle moodustada korrutamise ja liitmise abil. Näiteks avaldise `Math.random()*10+20` väärtus ei ole väiksem kui 20.0 ja on väiksem kui 30.0.

Kui on tarvis juhuslikku täisarvu, siis oleks sobivaks teeks näiteks reaalarvulise juhusliku arvu ümardamine. Tähele tuleb panna, et `Math.random` tagastab `double`-tüüpi arvu, kuid `Math.round` tagastab `double`-tüüpi argumendi puhul `long`-tüüpi täisarvu. Niisiis toimivad kenasti järgmised omistamised:

```
double realRandomNumber = Math.random()*5+15;
long longRandomNumber = Math.round(Math.random()*5+15);
```

Kui aga tahaksime `int`-tüüpi juhuslikku arvu, siis on vajalik tüübiteisendus:

```
int intRandomNumber = (int)Math.round(Math.random()*5+15);
```

Näide:

```
int k;
int count = 0;
do {
   k=(int) ( Math.random()*10 );
   System.out.println(" k = "   +  k);
   count++;
}while(k  !=  5);
System.out.println("count  = "+count);
```

Meetodid

Kuigi olemasolevates sadades klassides on tuhandeid meetodeid, mida saame kasutada, peame oskama ka ise neid juurde teha. Selles praktikumis piirdume *staatiliste meetoditega* (nimetatakse ka klassimeetoditeks). Eristatavad on need seeläbi, et piiritlejaks on võtmesõna `static`. Kui vaadata klassi `Math` meetodeid, siis need on kõik staatilised, mida kasutades tuleb meetodi ette lisada klassinimi. Staatiline on ka peameetod `main`.

Signatuur on meetodi (või konstruktori) iseloomustus, mis koosneb nimest ning formaalsete parameetrite tüüpide loetelust. Näiteks kui meetodi päis on

```
static double methodName (int a, double b, char c),
```

siis signatuur on `methodName`*(int, double, char)*.

Meetodi igal väljakutsel väärtustatakse vastavate argumentide väärtusega muutujad (formaalsed parameetrid), mis on meetodi formaalsete parameetrite loetelus kirjeldatud. Need muutujad on kasutatavad ainult selle meetodi kehas. Rõhutame, et kui meetod ei tagasta väärtust, siis on tagastustüübiks void (tühitüüp). Kõigil teistel juhtudel aga peab tagastustüüp olema kooskõlas

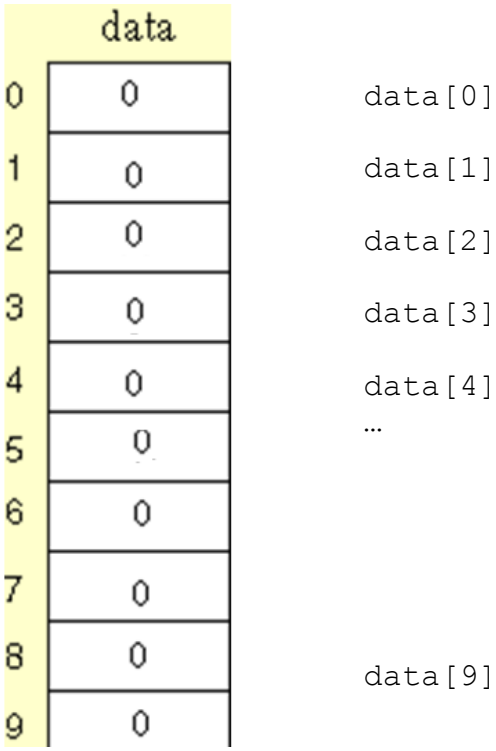
sellega, mis tüüpi väärtuse meetod tagastab. Tagastatakse aga selle avaldise väärtus, mis asub naasmisdirektiivis võtmesõna `return` järel.

Meetodeid võib ühes klassis olla mitmeid, näiteks järgmises klassis on kolm meetodit, kõik staatilised.

Massiivid

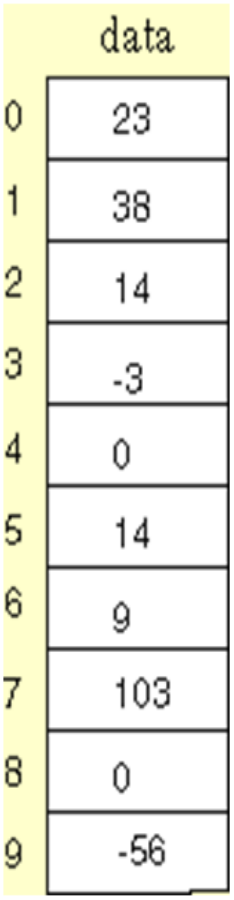
Lineaarselt järjestatud hulka nimetatakse *jadaks*, igal jada elemendil on tema järjenumbrit näitav indeks. *Järjendiks* nimetatakse lõplikku jada. Programmeerimises on järjendeid sageli vaja ning nende kujutamiseks sobib ühemõõtmeline *massiiv*. *Javas* saab järjend (massiiv) koosneda vaid üht tüüpi elementidest. Võime rääkida näiteks täisarvude massiivist. Kuna järjend ja massiiv on tihedalt seotud, siis järgnevas tekstis ongi neid käsitletud praktiliselt sünonüümidena.

```
int[] data = new
int[10];
int length =
data.length;
```



Väärtuste omistamine:

```
data [0] =
23;
data [1] =
38;
data [2] =
14;
data [3] = -
3;
data [5] =
14;
data [6] = 9;
data [7] =
103;
data [9] = -
56;
```




```
System.out.println(data[2]);//14
int  element = data[0];
System.out.println(element);//23
```

Kõik elemendid on ühte ja sama tüüpi, arvu nurksulgudes (järjekorranumbrit) nimetatakse elemendi *indeksiks*. Iga element võib sõltumata teistest elementidest sisaldada korraga ühte *väärtust* – täisarvujärjendi puhul täisarvu, reaalarvujärjendi puhul reaalarvu jne, ning igale elemendile võib väärtuse omistada teistest elementidest sõltumatult. *Javas* kehtib reegel, et järjendi elementide numeratsioon algab alati nullist. Kui järjendis on kokku n elementi, siis viimase elemendi indeks on n-1. Niisugust ühekohalist „nihet“ võrreldes harjunud loendamisviisiga tuleb *Java*-programmide koostamisel silmas pidada. (Järjendi elemendid võivad olla ka ise järjendid (kõik siis muidugi üht tüüpi järjendid, küll aga võib nende pikkus olla erinev). See lause on lugemiskontrolliks - palun kirjutage paberile ennustus, kas pärast oktoobrikuiseid valimisi jääb Tartu linnapea koht reformierakonnale ja Tallinna linnapea koht keskerakonnale. Sellisel juhul kujutatakse seda *Javas* kahemõõtmelise massiivina ja konkreetsele elemendile viidatakse kahe indeksiga, nt. b[2][3].)

Järjendi kirjeldamiseks (programmis kasutuselevõtmiseks) on mitu võimalust, kõige lihtsam on seda teha *massiivialgati* abil. Näiteks kui oleme viiel päeval lugenud kraadiklaasilt temperatuuri väärtused

```
10 9 12 11 8
```

siis võime nendest moodustada järjendi

```
int[] a = {10, 9, 12, 11, 8};
```

Niisuguse kirjelduse toimel võtab arvuti kasutusele täisarvujärjendi `a` ja omistab selle elementidele järjestikku loogelistes sulgudes antud väärtused (seega näiteks `a[0]` väärtuseks saab `10` ja `a[1]` väärtuseks `9`). Ühtlasi määratakse kindlaks järjendi elementide arv ehk massiivi *pikkus*, mis saab konstandi a.length väärtuseks (praegusel juhul on selleks `5`). (Tegelikult on lubatud ka nurksulgude teine paigutus: `int a[] = ...`)

Teine võimalus järjendi kasutuselevõtuks on *massiiviloome*. Eelneva näitega samaväärse tulemuse saame ka nii, et moodustame kõigepealt käsuga

```
int[] a = new int[5];
```

tühja viieelemendilise järjendi. Selle kõigi elementide algväärtuseks määratakse automaatselt `0`. Seejärel omistame elementidele nullide asemel vajalikud väärtused:

```
a[0] = 10;
a[1] = 9;
a[2] = 12;
a[3] = 11;
a[4] = 8;
```

Järjendi ühe elemendi väärtust võib *väljastada* standardsel viisil nagu iga teise muutuja väärtust.

```
System.out.println("Jarjendi esimene element on " + a[0]);
```

Näide:

```
int[] data = {23, 38, 14, -3, 0, 14, 9, 103, 0, -56 };

for (int indx= 0; indx < data.length ; indx++){

    System.out.print( data[ indx ] + "   " );

}

System.out.println();

for (int elem : data){  // read only
    System.out.println(elem);
}
```