TARTU ÜLIKOOL
Arvutiteaduse instituut

PROGRAMMEERIMISE PRAKTIKUMID

Algklassid

TARTU 2003

TARTU ÜLIKOOL

ARVUTITEADUSE INSTITUUT

Programmeerimise praktikumid

Algklassid

Koostajad:

Ireen Meho, Heli Uibo, Jaanus Jaeger,
Jüri Kiho, Reimo Palm, Ahti Peder,
Uuno Puus, Priit Salumaa ja Eno Tõnisson

Tartu 2003

Selle õppevahendi koostamist toetas Eesti Infotehnoloogia Sihtasutus Tiigriülikooli projekti raames.

Sisukord

ISBN 9985-56-801-x

Tartu Ülikooli Kirjastus
Tiigi 78, Tartu 50410
Tellimus nr. 667

Saateks

Tartu Ülikooli arvutiteaduse instituudi kursus *MTAT.03.100. Programmeerimine* on kõigile matemaatika-informaatikateaduskonna ning paljudele füüsika-keemiateaduskonna tudengitele kohustuslik. Kursusel osaleb ka mitmete teiste erialade üliõpilasi. Käesolevad materjalid (nn. algklassid) moodustavad umbes poole kursuse praktikumimaterjalidest. Kursusel osalejatele on materjalid elektrooniliselt kättesaadavad WebCT keskkonnas.

Palume anda ebakorrektsustest ja soovitustest teada aadressil eno@ut.ee. Käsitlegemgi seda trükki kui esimest ja mõnes mõttes proovitrükki.

Programmeerimist ei ole kerge õppida, väga tähtis osa on praktilistel harjutustel. Loodame, et materjalidest leiavad tudengid abi õppimisel, samuti ka tahvlipraktikumi kontrolltööl ja eksamil. Kursusel kasutatakse programmeerimiskeelt Java, millega tutvumist nüüd alustagemgi:

```
for ( ; ; ) {
   õppida( );
   System.out.println("Jälle natuke rohkem selge!");
}
```

I klass

Sissejuhatus

Teemad. *Programmi sisestamine, redigeerimine ja laadimine veebist. Kompileerimine. Käivitamine.*

Pärast selle klassi läbimist üliõpilane

- oskab luua tekstiredaktori abil näite põhjal lihtsat programmiteksti ja salvestada seda ettenähtud nimega ettenähtud kohta;
- oskab kopeerida veebilehelt programmitekstilõiku oma programmiteksti;
- oskab salvestada veebist saadud programme (nt aabitsaprogramme) oma arvutisse;
- oskab teha muudatusi programmitekstis;
- oskab programmi kompileerida;
- oskab programmi käivitada;
- tunneb põhilisi Java-programmidega seostuvaid failitüüpe (laiendiga java või class);
- oskab adekvaatselt reageerida tavalisematele veateadetele (kompilaator või interpretaator pole kättesaadav, `NoClassDefFoundError` jms).

Praktikumijuhend

Esimese arvutipraktikumi põhieesmärgiks on omandada elementaarsed tehnilised oskused programmeerimiskeelega Java tegelemiseks. Alustuseks tuleb sisse tippida üks lihtne Java-programm, salvestada see ning seejärel programm kompileerida ja käivitada. Ärge kartke küsimustega praktikumijuhendaja poole pöörduda!

Ülesanne 1. Esimene Java-programm

1. Avage tekstiredaktor (Windowsi puhul näiteks Notepad, Wordpad vms, Solarise puhul Text Editor menüüst Application).

2. Sisestage järgmine programm:

   ```
   // Programm, mis kirjutab ekraanile tervituse.
   class Tere {
       public static void main(String[] args) {
           System.out.println("Tere, tudeng!");
       }
   }
   ```

   Mida need käsud täpsemalt tähendavad, sellega tutvume edaspidi.

3. Salvestage fail nime all `Tere.java` võrgukettale (mitmes Windowsi-klassis on selleks H-ketas; soovitatav on luua omaette alamkataloog nt nimega `prog`).

   Selleks, et Notepad või Wordpad ei paneks nime lõppu segavat laiendit txt, peate salvestamisel lisama failinime ümber jutumärgid: `"Tere.java"`.

4. Avage käsurida.

   Windowsi puhul avaneb käsurida siis, kui valite Start-menüüst programmide hulgast Command Prompt (ka MS-DOS Prompt) või annate Start-menüüst käsu Run... ja trükite avanevale reale `command` (Win 95, 98) või `cmd` (Win NT, 2000, XP).

   Solarise puhul valige menüüst Application programm Terminal.

Võimalik on tegutseda ka TÜ serveris, kasutades mõnda terminaliprogrammi (nagu näiteks telnet, SSH Client, TeraTerm, PuTTY). Sisselogimisel sisestage oma kasutajanimi ja parool, mis Teil on TÜ serveris (st need, millega logisite sellesse arvutisse, kus töötate).

5. Tõenäoliselt on Teil nüüd ees käsuviip, näiteks selline:

```
H:\>
```

   või

```
nimi@arvutinimi:~>
```

   Muutke käsu cd abil aktiivseks see kataloog, kuhu salvestasite oma programmi.

   Windowsis:

```
H:\>cd prog
```

   Solarises ja Unixis:

```
nimi@arvutinimi:~>cd prog
```

   Tulemus peaks olema selline:

```
H:\>cd prog
```

   või

```
nimi@arvutinimi:~/prog>
```

6. Kontrollige, kas java-fail, mille just salvestasite, on ikka olemas. Selleks andke Windowsis käsk dir ning Solarises ja Unixis käsk ls -l. Nüüd peaks Teil ekraanil olema järgmine tekst.

   Windowsis:

```
H:\prog>dir
 Directory of H:\prog
27.02.2003  08:37    <DIR>          .
27.02.2003  08:35    <DIR>          ..
27.02.2003  08:37               152 Tere.java
```

   Solarises ja Unixis:

```
nimi@arvutinimi:~/prog>ls -l
-rwxr--r--   1 nimi      users        152 27. veebr 08:37 Tere.java*
```

7. Kompileerige oma programm, sisestades käsurealt

```
javac Tere.java
```

   Kui kompileerimine õnnestub, siis kompilaator mingeid teateid ei anna ja pärast töö lõppu ilmub uuesti ilmub käsuviip `prog>`. Kataloogi tekib fail `Tere.class`. Kui kompilaator leiab programmi tekstis vigu, siis kuvab ta ekraanile vastavad teated.

   Kui Windowsis ilmub veateade `The name specified is not recognized as an internal or external command ...` või `Bad command or file name`, siis ei õnnestunud arvutil kompilaatorit javac käivitada. Sel juhul tuleks käsurealt paika panna otsimistee (andes käsu `path P:\j2sdk1.4.1\bin`) ja proovida uuesti.

8. Vaadake kataloogi `prog` sisu käsuga dir (Solarises ja Unixis ls -l). Seal peaks olema nüüd kaks faili, `Tere.java` ja `Tere.class`:

```
H:\prog>dir
 Directory of H:\prog
27.02.2003  08:49    <DIR>          .
27.02.2003  08:49    <DIR>          ..
27.02.2003  08:37               152 Tere.java
27.02.2003  08:49               415 Tere.class
```

9. Käivitage programm Java-interpretaatori abil, sisestades käsurealt

```
java Tere
```

   Kui tekkis veateade `Exception in thread "main" java.lang.NoClassDefFoundError: Tere` (mis näitab, et interpretaator ei leidnud vajalikku class-faili), siis sisestage käsurealt `set classpath=` ja proovige uuesti.

10. Kui tulemusena väljastati ekraanile lause `Tere, tudeng!`, siis võite ennast õnnitleda: olete esimese Java-programmi tööle saanud. Kutsuge ka praktikumi juhendaja vaatama ja rõõmustama.

Ülesanne 2. Programmi muutmine

Nüüd proovige programmi muuta. Uurige, mis juhtub, kui muuta

- jutumärkides olevat teksti `Tere, tudeng!`
- klassi nime `Tere`
- meetodi nime `main`

Iga muudatuse järel salvestage, kompileerige ja käivitage programm uuesti. Praktikumi juhendajale pole vaja selle ülesande lahendust näidata, küll aga kirjutage üles, millised veateated tekivad ja kuidas neist jagu saada.

Ülesanne 3. Programmilõik veebist

1. Avage tekstiredaktoris uus dokument. Kopeerige sinna järgnev lõik.

```
// Programm, mis kirjutab Gaudeamuse kaks esimest rida.
class Gaudeamus {
    public static void main(String[] args) {
        System.out.println("Gaudeamus igitur,");
        System.out.println("juvenes dum sumus!");
    }
}
```

2. Salvestage, kompileerige ja käivitage. Kui kõik õnnestub, siis näidake tulemust ka praktikumi juhendajale.

Ülesanne 4. Programm veebist

Veebist võib alla laadida ka terveid programme. Harjutuseks otsige veebist üles J. Kiho Java programmeerimise aabitsa programmid (http://www.ut.ee/~kiho/progr/Aabits/Programmid). Salvestage kataloogist `vihik1` oma arvutisse programm `Kaugus.java` ning kompileerige ja käivitage see. Kui käivitamine õnnestub, siis näidake seda ka praktikumi juhendajale.

Iseseisev töö

- Soovitatav on iseseisvalt kõiki vaadeldud operatsioone korrata, isegi kui praktikum arvestatud sai.
- Kui tahate Javat oma koduarvutisse installeerida, siis võite leida vajalikud failid näiteks aadressilt http://www.cs.ut.ee/~jaanus/java/download/. Pärast installeerimist kontrollige, kas arvutiklassis töötanud programmid ka kodus tööle hakkavad.

Viited samateemalistele materjalidele

- Kompileerimise ja käivitamise õpetus on olemas paljudes Java-programmeerimist puudutavates materjalides, mis on saadaval veebis.
- Tekstiredaktorite ja failikäsitluse kohta leiab juhendeid mitmesugustest arvuti kasutamist käsitlevatest materjalidest. Samuti vaadeldakse neid teemasid TÜ ainetes *Arvutikäsitusõpetus* ja *Arvutiõpetus*.

Märkus

On olemas mitmeid integreeritud programmeerimiskeskkondi, mis on mõeldud spetsiaalselt Java-programmide koostamiseks (nt JBuilder, JCreator, Sun ONE Studio, nimekirja leiab aadressilt http://java.about.com/cs/ides/). Need keskkonnad sisaldavad mitmeid mugavaid redigeerimisvõimalusi, aga jaosvara- või demoversioonide funktsionaalsus võib olla mõneti piiratud. Praktikumiülesannete lahendamisel pole keelatud kasutada mõnda niisugust keskkonda, kuid peab arvestama, et eksamil ei pruugi see kättesaadav olla.

II klass

Teksti väljastamine. Muutuja. Avaldis

Teemad. *Üheklassilise programmi struktuur. Teksti väljastamine. Täisarvutüüpi muutuja. Omistamine. Avaldis. Avaldise väärtuse väljastamine koos selgitava tekstiga.*

Pärast selle klassi läbimist üliõpilane

- tunneb üheklassilise ja vaid peameetodit sisaldava programmi struktuuri;
- oskab väljastada teksti käskude `System.out.println` ja `System.out.print` abil;
- tunneb täisarvutüüpi `int`;
- oskab kirjeldada täisarvulist muutujat ja omistada sellele väärtust;
- oskab sooritada täisarvutüüpi muutujatega lihtsamaid aritmeetilisi tehteid (`+`, `-`, `*`,`/`,`%`);
- tunneb põgusalt sõnemuutuja kasutamist;
- oskab ekraanile väljastada muutujate väärtusi koos selgitava tekstiga;
- oskab adekvaatselt reageerida veateadetele, mis tekivad nt puuduva looksulu, ümarsulu, semikooloni jms korral.

Praktikumijuhend

Teema 1. Programmi struktuur

Juba eespool on olnud juttu, et programmeerimine keeles Java ei tähenda midagi muud kui klasside koostamist. Kursuse alguses piirdume selliste programmidega, mis sisaldavad ainult ühte klassi, *peaklassi*, mis sisaldab *peameetodit* nimega `main`. Peameetodist algab alati programmi täitmine.

Vaatleme uuesti eelmises praktikumis kirjutatud tervitamisprogrammi, esitades selle ridahaaval koos põhjalikemate märkustega.

Programmi algusesse lisame kommentaari, mis selgitab programmi sisu:

```
// Programm, mis kirjutab ekraanile tervituse.
```

Nüüd alustame peaklassi kirjeldust. Kõigepealt antakse klassile nimi, soovitatav (mõnel juhul ka kohustuslik) on salvestada programmiteksti sisaldav fail sama nimega, mis on peaklassil:

```
class Tere {
```

Peaklassi sisus kirjeldame peameetodi `main`. Võtmesõnad `public static`, mida nimetatakse *piiritlejateks*, ning tagastustüübi määraja `void` on peameetodi puhul kohustuslikud. Piiritleja `public` tagab, et peameetod on Java käituskeskkonnale kättesaadav. Piiritleja `static` näitab, et tegemist on klassimeetodiga (hiljem tulevad vaatluse alla veel ka isendimeetodid). Tagastustüüpidest teeme juttu järgmises praktikumis. Muutujat `args`, mille tüüp on `String[]`, saab kasutada programmile käsurealt argumentide etteandmiseks.

```
    public static void main(String[] args) {
```

Meetodi kehas võib olla mitu rida, antud näites on neid küll üksainus:

```
        System.out.println("Tere, tudeng!");
```

Sellega lõpevad peameetodi ja klassi sisu, mida näitavad sulgevad sulud

```
        } // lõpeb peameetod main
    } // lõpeb klass Tere
```

Looksulud esinevad alati paarikaupa ning nende vahele jääv programmi osa moodustab ühe terviku.

Ülesanne 1. Vead

Tehke tervitamisprogrammis ühekaupa järgmised muudatused ning püüdke iga muudatuse järel programm salvestada, kompileerida ja käivitada. Tõrke puhul tutvuge veateatega ning selgitage, mida tuleks teha, kui programmis ilmneb selline viga. Võite teha märkmeid. Seejärel muutke programm endiseks ja proovige järgmist muudatust.

1. Kirjutage sõna `main` asemele `Main`.
2. Muutke klassi nime, nt `Tere` asemel `tere`.
3. Eemaldage programmi viimane lõpetav looksulg `}`.
4. Eemaldage `Tere` järelt alustav looksulg `{`.
5. Eemaldage väljatrükikäsu lõpust semikoolon.
6. Eemaldage teksti väljastamise reast lõppev ümarsulg `)`.
7. Asendage sõnas `println` täht `i` tähega `a`.
8. Eemaldage sõnas `println` kaks viimast tähte.

Proovige, küsige.

Teema 2. Teksti väljastamine

Eelmises praktikumis tutvusime elementaarsete tehniliste oskustega, mida läheb vaja programmide koostamisel. Kõik vaadeldud programmid väljastasid ekraanile ka natuke teksti. Nüüd uurime teksti väljastamise võimalusi lähemalt.

Käsk

```
System.out.println("Tere, tudeng!");
```

kirjutab ekraanile jutumärkide vahel asuva teksti ja ühtlasi viib väljastamisjärje uude ritta. Käsk

```
System.out.print("Tere, tudeng!");
```

väljastab küll teksti, aga rida ei vaheta. Käsus võib tekst sulgude sees ka hoopis puududa, näiteks

```
System.out.println();
```

annab ainult reavahetuse.

Teema 3. Täisarvutüüpi muutuja

Muutuja on teatavat tüüpi väärtuste hoidmiseks ettenähtud mäluväli. Muutujale võib omistada teatava väärtuse ning kasutada seda avaldise väärtuse arvutamisel. Muutujale viidatakse tema nime järgi. On olemas mitu väärtuste tüüpi, kuid esialgu piirdume täisarvuliste muutujatega. Ka täisarvulisi muutujaid on üldse nelja erinevat tüüpi, käesolevas praktikumis vaatleme tüüpi `int`. Kui tegelikult on täisarvude hulk lõpmatu, siis `int`-tüüpi muutuja võib omandada ainult lõpliku hulga erinevaid väärtusi: vähim `int`-tüüpi täisarv on -2147483648 ja suurim 2147483647.

Enne, kui muutujat kasutama saab hakata, tuleb see kirjeldada. Muutujakirjelduse toimel reserveeritakse antud tüüpi ja antud nimega muutuja jaoks mälus koht. Korraga võib kirjeldada kas ühe või mitu sama tüüpi muutujat:

```
int a;
int c, b, autodeArv;
```

Muutuja nimi peaks väljendama muutuja sisu ja olema lihtsasti arusaadav, ta võib sisaldada tähti (ka täpitähti), numbreid ning märke _ ja $. Muutuja nimi ei tohi alata numbriga. Kuigi nime pikkusel on olemas piir, on see siiski nii suur, et reaalset kasutust ei häiri. Suured ja väikesed tähed (nt `A` ja `a`) loetakse erinevaks (nt `jaAk` ja `jAak` on erinevad). Muutuja nimes ei tohi esineda tühikuid – kui nimi koosneb mitmest sõnast, siis võib kasutada allkriipsu: `autode_arv`, või kirjutada kõik sõnad alates teisest suure algustähega: `autodeArv`.

Muutujale omistatakse väärtus tehte = abil:

```
a = 12;
b = a + 10;     // b väärtuseks on nüüd 22
c = a + b;      // c väärtus on 34
a = a + 2;      // a väärtus on 14
```

Lubatud on ühendada muutuja kirjeldamine ja muutujale väärtuse omistamine:

```
int koht = 8;
```

Teema 4. Muutujate väärtuste väljastamine

Kuidas näha muutuja väärtusi ekraanil?

Muutuja `a` väärtuse saab väljastada näiteks käsuga

```
System.out.println(a);
```

Väärtusele võib lisada ka selgitava teksti, näiteks

```
System.out.println("Arvude " + a + " ja " + b + " summa on " + c);
```

Teema 5. Tehted. Avaldised

Täisarvuliste muutujatega saab teha mitmesuguseid tehteid, millest siinkohal tulevad vaatluse alla

- liitmine (`+`)
- lahutamine (`-`)
- korrutamine (`*`)
- jagamine (`/`)
- jäägi leidmine (`%`)

Enamasti toimivad need nii nagu matemaatikas kombeks, kuid on ka erinevusi.

Ülesanne 2. Aritmeetilised tehted

Koostage programm nende tehete omadustega tutvumiseks. Selleks kirjeldage täisarvulised muutujad `a`, `b`,`c`,`d`,`e` ning omistage muutujale a väärtus 2147483647, ülejäänute väärtused valige ise.

1. Liitke, lahutage ja korrutage muutujate väärtusi.
2. Mis saab siis, kui muutuja `a` väärtusele liita 1? Kuidas seda tulemust seletada?
3. Leidke jagatised 9/3, 7/4, -3/2, 3/(-2), -7/(-3).
4. Proovige leida jääki nii positiivsetest kui ka negatiivsetest arvudest.

Tulemused väljastage näiteks nii:

```
System.out.println("Arvude " + b + " ja " + c + " summa on " + (b + c));
```

Mis muutub, kui avaldises `(b + c)` sulud ära jätta?

Teema 6. Sõnemuutuja

Muutujatüüpe on Javas mitmeid, täisarvutüüpi muutujaid juba kasutasime eelnevas. Vaatleme veel põgusalt sõnemuutujat (tüüp `String`). Tegelikult on ka sõnet juba eelnevates programmides kasutatud. Näiteks käsus

```
System.out.println("Tere, tudeng!");
```

esineb sõne `Tere, tudeng!`. Sõnesid saab üksteisega liita (sidurdada), näiteks sõnedest `jalg` ja `pall` saame sõne `jalgpall`. Liitmistehet väljendab sõnede nii nagu arvude puhulgi märk `+`. Need märgi `+` erinevad tähendused koos automaatselt toimuva arvude sõnedeks teisendamisega võivad mõnikord väljastamiskäskudes esmapilgul ootamatuid tulemusi anda.

Analoogiliselt täisarvudega saab kirjeldada ka sõnetüüpi muutujaid ning anda neile väärtusi:

```
String sõne1 = "See on selline proovisõne";
```

Loomulikult saab sõnemuutujate väärtusi ka ekraanile väljastada.

Proovige ise.

Ülesanne 3. Busside täitmine

Inimeste transportimiseks ühest kohast teise soovitakse tellida teatav arv ühesuguseid busse. Antud on inimeste arv ja kohtade arv ühes bussis. Kõik bussid peavad saama täiesti täis. Ülejäänud inimesed jäävad maha. Koostage programm selle ülesande lahendamiseks.

Programmi algusesse lisage kommentaar, kus on kirjas ülesande püstitus ja tulemus.

Sisu alguses võtke kasutusele muutujad inimeste arvu jaoks ja ühe bussi kohtade arvu jaoks ning andke nendele muutujatele algväärtused.

Programm peab väljastama

- inimeste arvu;
- kohtade arvu ühes bussis;
- busside arvu;
- bussidesse mahtunud inimeste arvu;
- mahajäänud inimeste arvu.

Ülesande lahendamiseks kirjeldage vajalikud muutujad ning pange kirja avaldised, mille järgi tehakse arvutusi (võib kasutada jäägi leidmise operatsiooni `%`).

Väljatrükis kasutage erinevaid variante (`System.out.println`,`System.out.print`). Väljatrükk peab koosnema täislausetest, näiteks:

```
---
Ülesanne:
Bussidesse tuli paigutada 142 inimest. Ühte bussi mahtus 30 inimest.
Lahend:
Kokku oli vaja 4 bussi, neisse mahtus 120 inimest.
Maha jäi 22 inimest.
---
```

Iseseisev töö

Ülesanne 4. Küpsisetort

Küpsisetordi tegemisel laotakse terved ruudukujulised küpsised ristkülikukujulisele kandikule. Kandikule mahub ühes kihis `kandikuPikkus*kandikuLaius` küpsist. Laotakse `tordiPaksus` kihti. Ühes küpsisepakis on `küpsistPakis` küpsist. Püstitage ülesanne küpsisetordi tegemise kohta ja lahendage see.

Viited samateemalistele materjalidele

- J. Kiho. Java programmeerimise aabits. Tartu, 2002, vihik 1.

III klass

Meetod. Klass

Teemad. *Reaalarvutüüpi muutuja. Klassimeetod. Klass kui klassimeetodite komplekt. Lokaalmuutuja. Tagastustüüp. Naasmisdirektiiv. Meetodi rakendamine.*

Pärast selle klassi läbimist üliõpilane

- tunneb erinevaid muutujatüüpe (täisarv, reaalarv, sõne);
- oskab jagada programmeerimisülesannet alamülesanneteks;
- oskab alamülesande lahendusalgoritmi Java-keeles meetodina kirja panna;
- teab, et meetodi sisus kirjeldatud muutujad on kättesaadavad vaid selles meetodis;
- teab, mis on tagastustüüp ja oskab meetodist väärtusi tagastada;
- teab, mis on meetodi tüüp ja oskab seda seostada tagastustüübiga;
- oskab meetodit rakendada (välja kutsuda);
- oskab kasutada süsteemseid klasse ja meetodeid (`Math.sqrt`,`Math.random`,`Math.round`).

Praktikumijuhend

Teema 1. Reaalarvutüüpi muutuja

Eelmises praktikumis saime teada, kuidas kirjeldada muutujaid ja omistada neile väärtusi aritmeetiliste avaldiste abil. Siiski käsitleti seal ainult täisarvulisi muutujaid ja avaldisi, millest ilmselt ei piisa kõigi ülesannete lahendamiseks. Programmeerimiskeeles Java saab kasutada ka murdarve, seejuures tuleb nende kirjapanekul arvestada, et erinevalt tavalisest (Eestis kasutatavast) kirjaviisist on täisosa ja murdosa eraldajaks punkt. Seega on arv 2,5 Java-keeles `2.5`. Väärtust, mis on kirja pandud punktiga kujul, nimetatakse Javas *reaalarvuliseks väärtuseks* (ka ujukoma- või ujupunktarvuks).

Ülesanne 1. Erinevad tüübid

Vaatleme omistamiskäsku

```
int a = 10/4;
```

Millise väärtuse saab siin muutuja `a`? Selle muutuja väärtuseks on 2, sest tegemist on kahe täisarvulise suuruse jagamisega. Seetõttu peab ka jagamistehte tulemus olema täisarv. Jagamistehte tulemus saadakse nii, et tegeliku jagatise murdosa lõigatakse lihtsalt ära.

Selleks, et saada õiget jagatist 2,5, tuleb omistamiskäsk kirja panna järgmiselt:

```
double b = 10.0/4.0;
```

Siin on tegemist juba kahe reaalarvulise suuruse jagamisega, tulemuseks on reaalarvuline väärtus, mis omistatakse reaalarvulist tüüpi muutujale `b`. Võtmesõna `double` märgibki, et muutuja `b` väärtuseks on reaalarv.

Proovige, milline tulemus tekib juhtudel, kui reaalarvulisele muutujale omistada väärtus `10.0/4` ja `10/4.0`.

Mis aga juhtub, kui kirjutada järgmine omistamisdirektiiv:

```
double c = 10/4;
```

Kirjutage programm (nagu eelmises praktikumis), mis koosneb kolmest eeltoodud omistamisdirektiivist ja vastavate muutujate väärtuste väljastamiskäsust

```
System.out.println("a = " + a + " b = " + b + " c = " + c );
```

Käivitage see programm. Milline on tulemus? Miks nii?

Asi on selles, et muutuja `c` väärtustamisel arvutatakse kahe täisarvu `10` ja `4` jagatis, mis on täisarv. See väärtus saab ka muutuja `c` väärtuseks, vaatamata sellele, et `c` ise on reaalarvulist tüüpi. Seega näeme, et täisarvulist väärtust võib omistada reaalarvutüüpi muutujale.

Kuigi üldiselt peavad muutuja ja sellele omistatava väärtuse tüüp kokku langema, lubab Java reaalarvulisele muutujale omistada täisarvulisi väärtusi. See aga on pigem erand kui reegel. Näiteks täiendage oma programmi järgmise reaga:

```
int d = "123";
```

Mis on tulemuseks?

Programmi muutes ning katsetades täitke nüüd alljärgneva tabeli tühjad lahtrid („jah“ tähendab, et vastava väärtuse omistamine vastavat tüüpi muutujale on programmeerimiskeeles Java võimalik, „ei“ aga tähendab, et niisugune omistamine pole võimalik).

	Täisarvulist tüüpi (`int`) väärtus	Reaalarvulist tüüpi (`double`) väärtus	Sõnetüüpi (`String`) väärtus
Täisarvulist tüüpi (`int`) muutuja	`int a = 123` jah		`int d = "123"` ei
Reaalarvulist tüüpi (`double`) muutuja	`double f = 123` jah	`double g = 123.0` jah	
Sõnetüüpi (`String`) muutuja			`String h = "123"` jah

Reaalarvutüüpi muutuja väärtusevahemik on täisarvutüüpi muutuja omast oluliselt laiem. Kõige väiksem võimalik positiivne väärtus tüübi `double` puhul on 2^{-1074} ja kõige suurem $(2-2^{-52})\times 2^{1023}$ ehk ligikaudu $1{,}7\times 10^{308}$. Muidugi on võimalik kasutada ka negatiivseid arve.

Teema 2. Meetodid programmeerimiskeeles Java

Eelmistes praktikumides koostasime programme mitmesuguste suuruste arvutamiseks ja väljastamiseks. Programm on ülesande lahendamiseks vajalike järjestikuste käskude ehk direktiivide jada, näiteks võib ta koosneda teatavast hulgast omistamis- ja väljastuskäskudest. Mahuka programmi korral võib see jada olla küllaltki pikk. Teksti lühendamiseks (ja ka paljudel muudel põhjustel, millega tutvume edaspidi) on programmeerimiskeeltes olemas võimalus osa programmist kirjeldada korduvalt kasutatavana. Keeles Java kirjeldatakse korduvalt kasutatav programmilõik meetodina. Seda, mis on meetodid ja kuidas neid kirja panna, vaatlemegi alljärgnevalt.

Ülesanne 2. $

Olgu ülesandeks väljastada ekraanile järgmine kujund:

```
  $
 $$$
$$$$$
 $$$
  $
```

Seda saab hõlpsasti teha viie trükkimiskäsuga

```
System.out.println("  $  ");
System.out.println(" $$$ ");
System.out.println("$$$$$");
System.out.println(" $$$ ");
System.out.println("  $  ");
```

Nii oleme ühe kujundi joonistamisega edukalt hakkama saanud. Kui aga soovime joonistada üksteise järele viis sellist kujundit, siis on analoogilist programmi ridahaaval väga tüütu kirja panna. Siin tulebki meile appi võimalus muuta üks programmilõik korduvalt kasutatavaks. Kirjeldame meetodi nimega `kujund` ühe kujundi joonistamiseks ja seejärel rakendame seda meetodit viis korda. Niisugune meetod pannakse keeles Java kirja järgmiselt:

```
static void kujund() {
    System.out.println("  $  ");
    System.out.println(" $$$ ");
    System.out.println("$$$$$");
    System.out.println(" $$$ ");
    System.out.println("  $  ");
}
```

Meetodikirjeldus algab võtmesõnadega `static` ja `void`, mille täpset tähendust selgitame hiljem. Sõna `kujund` on meetodi nimi. Meetodi nime puhul tuleb arvestada samu reegleid kui muutuja nime korral (vt eelmist praktikumi). On võimalik, et meetodi nimi ühtib mõne muutuja nimega: see pole iseenesest keelatud, sest muutuja ja meetod on Java mõistes erinevad asjad. Siiski võiks esialgu programmi selguse mõttes hoiduda sellistest kokkulangemistest. Meetodi nimele järgnevate ümarsulgude vahel edastatakse meetodile andmeid (sellest täpsemalt järgnevas). Meetodi sisu moodustavad käsud kirjutatakse ümarsulgudele järgnevate loogeliste sulgude vahele.

Viie kujundi joonistamise programm, mis kasutab meetodit `kujund`, on siis selline:

```
class Kujundid {
    static void kujund() {
        System.out.println("  $  ");
        System.out.println(" $$$ ");
        System.out.println("$$$$$");
        System.out.println(" $$$ ");
        System.out.println("  $  ");
    }
    public static void main (String[] args) {
        kujund();
        kujund();
        kujund();
        kujund();
        kujund();
    }
}
```

Sisestage see programm (kopeerige tekst) ja pange tööle.

Käesolevas ülesandes väljastati kujund, mis koosnes dollarimärkidest `$`. Võib aga leiduda inimesi, kellele sellised märgid kujundis ei meeldi. Meie meetod `kujund` oleks märksa rohkem „korduvkasutatav“, kui oleks kuidagi võimalik ka kujundit moodustavat märki muuta nii, et meetodit ennast ei peaks ümber kirjutama. Selleks on keeles Java võimalik iga meetodiga seostada formaalseid parameetreid ja nende abil vajalikke väärtusi meetodile ette anda.

Ülesanne 3. XY-kujund

Koostage programm eelmise ülesande kujundite väljastamiseks selliselt, et need kujundid koosneksid dollarite asemel vaheldumisi tähtedest `X` ja `Y`.

Ülesande lahendamiseks muudame meetodit `kujund` selliselt, et lisame meetodi nime järele sulgudesse (selleks need sulud seal mõeldud ongi) sõnetüüpi parameetri `m`, mille väärtuseks on kujundit moodustav sümbol. Meie meetod näeb nüüd välja järgmiselt:

```
    static void kujund(String m) {
        System.out.println("  "+ m +"  ");
        System.out.println(" " + m + m + m + " ");
        System.out.println(m + m + m + m + m);
        System.out.println(" " + m + m + m + " ");
        System.out.println("  "+ m +"  ");
    }
```

Muutuja `m` on selles meetodis *formaalne parameeter*. Formaalne selles mõttes, et `m` saab väärtuse vaid siis, kui antud meetod välja kutsutakse. Meetodi kirjutamise ajal me ei tea (ega peagi teadma), millistest märkidest väljastatav kujund koosneb. Küll aga teab seda see, kes meie meetodit kasutab.

Rida

```
    kujund("X");
```

väljastab tähtedest `X` koosneva kujundi

```
  X
 XXX
XXXXX
 XXX
  X
```

rida

```
    kujund("Y");
```

aga samasuguse, kuid tähtedest `Y` koosneva kujundi. Sulgudes esinevaid sõnesid `"X"` ja `"Y"` nimetatakse *argumentideks*, nad määravad kujundi väljastamiseks kasutatava sümboli. Formaalne parameeter ja argument peavad olema sama tüüpi.

Koostage ülesande 2 eeskujul programm, mis väljastab vaheldumisi tähtedest `X` ja `Y` koostatud kujundid (kasutage meetodit `kujund` sobiva parameetriga):

```
  X
 XXX
XXXXX
 XXX
  X
  Y
 YYY
YYYYY
 YYY
  Y
  X
 XXX
XXXXX
 XXX
  X
  Y
 YYY
YYYYY
 YYY
  Y
```

Ülesanne 4. Rida

Koostage meetodid `rida1`, `rida3` ja `rida5` vastavalt kujundis esineva ühe-, kolme- ja viiemärgilise sümbolirea ekraanile trükkimiseks. Valige vabalt sobiv kujund ja kirjutage programm selle kujundi väljastamiseks, kasutades neid meetodeid.

Teema 3. Lokaalmuutujad. Naasmisdirektiiv. Tagastustüüp

Kujundi joonistamise ülesandes saime välja eraldada alamülesande „väljastada üks kujund“ ning seejärel kasutasime seda alamülesannet lahendavat meetodit `kujund` viis korda. Ka arvutusülesannete ja üldse programmeerimisülesannete lahendusi (algoritme, programme) võib samal viisil jaotada alamülesanneteks. Seejuures peavad programm ja tema meetodid arvutuste käigus vahetama andmeid (arvutuste lähteandmeid ja arvutuste käigus saadud tulemusi). Kuidas lähteandmeid ette anda, seda nägime juba ülesande 3 lahendamisel. Vaatleme nüüd, milles üldse seisneb probleem väljaarvutatud tulemuste tagastamisel ja kuidas seda lahendatakse.

Ülesanne 5. Kolmnurga ümbermõõt

Olgu antud kolmnurga tippude koordinaadid. Leida nende tippudega määratud kolmnurga ümbermõõt.

Ülesande lahendamiseks paneme kõigepealt kirja avaldise kahe punkti vahelise kauguse arvutamiseks. Nii punktide koordinaadid kui ka kaugus olgu reaalarvud st `double`-tüüpi muutujad. Et punktidevaheline kaugus võrdub ruutjuurega koordinaatide vahede ruutude summast ehk valemiga $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$, on otsitav omistamiskäsk järgmine:

```
double d = Math.sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
```

Siin tähendab `Math.sqrt` ruutjuurt. Vormistame nüüd selle valemi meetodina, et teda saaks kasutada kahe etteantud punkti kauguse arvutamiseks. Parameetriteks, mis asuvad meetodi nime `vahemaa` järel ümarsulgudes, on kummagi punkti koordinaadid (kokku neli reaalarvu):

```
static double vahemaa(double x1, double y1, double x2, double y2) {
    double d = Math.sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
    return d;
}
```

Kõik meetodi sisus kirjeldatud muutujad (praegusel juhul muutuja `d`, mis tähendab kahe punkti vahelist kaugust) on Java jaoks kättesaadavad vaid selles meetodis, aga mitte väljaspool, nagu näiteks peameetodis. Seetõttu on vajalik mingi mehhanism, mille abil saab meetodis arvutatud väärtuse tagastada põhiprogrammile. Niisugust mehhanismi kujutab valemile järgnevas reas kirjapandud *naasmisdirektiiv*

```
return d;
```

kus `d` on muutuja nimi, mille väärtus tagastatakse.

Lisaks naasmisdirektiivile peab meetodi esimeses reas olema määratud tagastatava väärtuse tüüp. Seda teeb meetodi nime `vahemaa` ees asuv võtmesõna `double`, mis ütleb, et meetod tagastab väärtusena reaalarvu. Tagastatava väärtuse tüübi järgi määratletakse ka meetodit ennast, näiteks on `vahemaa` reaalarvuline meetod. Meetodi tüüp ja naasmisdirektiivis esineva väärtuse (muutuja) tüüp peavad olema omavahel kooskõlas.

Olgu meil kolmnurga külje `a` otspunktideks punktid (2; 5) ja (2; 8). Külje `a` pikkuse saab siis arvutada meetodi `vahemaa` järgmise rakendamisega:

```
double a = vahemaa(2.0, 5.0, 2.0, 8.0);
```

Külje `b` otspunktid olgu (2; 8) ja (6; 8) ning külje `b` pikkuse arvutab korraldus

```
double b = vahemaa(2.0, 8.0, 6.0, 8.0);
```

Külje `c` otspunktid on nüüd (6; 8) ja (2; 5) ning külje `c` pikkuse leiame käsuga

```
double c = vahemaa(6.0, 8.0, 2.0, 5.0);
```

Kui soovime arvutada külgede pikkusi mingis teises kolmnurgas, siis oleks ebamugav muuta koordinaate kolmes kohas. Selleks, et teha seda vaid ühes kohas, tasub peameetodi alguses kirjeldada muutujad `p1x` (mis tähendab punkti P1 esimest, x-koordinaati), `p1y`, `p2x`,`p2y`,`p3x` ja `p3y`:

```
double p1x, p1y, p2x, p2y, p3x, p3y;
```

Siis saab selle külje pikkuse, mille otspunktid on `p1` ja `p2`, arvutada järgmiselt:

```
double a = vahemaa(p1x, p1y, p2x, p2y);
```

Kirjutage programm, mis arvutab ja väljastab kolmnurga ümbermõõdu. Seejuures

- klassis peab olema kaks meetodit: meetod `vahemaa` ja peameetod, mis meetodit `vahemaa` kasutab;
- peameetodi alguses tuleb punktide koordinaate esitavatele muutujatele omistada konkreetsed väärtused;
- ekraanile tuleb väljastada kolmnurga tippude koordinaadid, külgede pikkused ja ümbermõõt koos selgitava tekstiga.

Teema 4. Klassi `Math` meetodid

Ruutjuure leidmise meetodit `Math.sqrt` kasutasime juba eespool. Klassi `Math` paljude meetodite hulgas leidub ka näiteks meetod `Math.random`, mis tagastab (pseudo)juhusliku `double`-tüüpi arvu poollõigust [0,0; 1,0) (st 0 on kaasa arvatud, 1 aga mitte). Omistamisdirektiivi

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

täitmisel omandab muutuja `juhuarv` 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()*10.0+20` väärtus ei ole väiksem kui 20.0 ja on väiksem kui 30.0.

Selleks, et teada saada, millised meetodid klassis `Math` üldse olemas on, avage Java API veebileht ja otsige raamist All classes üles link Math.

Iseseisev töö

Ülesanne 6. Kehamassiindeks

Koostage meetod (nimega näiteks `kmindeks`), mis arvutab inimese massi ja pikkuse järgi kehamassiindeksi valemist

$$\text{indeks} = \frac{\text{mass}}{\text{pikkus}^2}$$

Mass on antud kilogrammides, pikkus meetrites (valem on saadud Terviselehest).

Peameetodis peaksid mass ja pikkus saama juhuslikud väärtused (kasutage meetodit `Math.random`), kuid sellised, mis võiksid täiskasvanud inimeste puhul võimalikud olla. Peameetod peab rakendama kirjeldatud meetodit (`kmindeks`) ning koos seletava tekstiga väljastama inimese pikkuse, massi ja kehamassiindeksi. Ümardada ei ole vaja, aga võib (reaalarvulise muutuja `x` sajandikeni ümardatud väärtuse esitab avaldis `Math.round(x*100)/100.0`). Programm varustada kommentaaridega.

Viited samateemalistele materjalidele

- J. Kiho. Java programmeerimise aabits. Tartu, 2002, vihik 1.

IV klass

Tingimusdirektiiv. Silumine

Teemad. *Tingimusdirektiiv. Loogiline tüüp. Kontrollimismeetod ja selle rakendamine. Programmi testimine ja silumine.*

Pärast selle klassi läbimist üliõpilane

- oskab kirja panna (lihtsamaid) loogilisi avaldisi ja kujutab ette, kuidas arvutatakse nende väärtusi;
- tunneb loogilist tüüpi;
- oskab eristada lahendustes ja algoritmides alternatiive;
- oskab alternatiive kirja panna programmeerimiskeeles Java;
- oskab kirja panna meetodeid, mis kontrollivad alternatiive ja tagastavad loogilise väärtuse;
- teab, mis on programmi algandmed ja testandmed;
- oskab koostada algandmeid kõigi programmis realiseeritud alternatiivide testimiseks ja määrata nende andmete puhul programmi töö oodatavat tulemust.

Praktikumijuhend

Teema 1. Tingimusdirektiiv

Kõikides eelnevates praktikumides koostasime programme, mille direktiivid täideti üksteise järel, alates programmi algusest (esimesena kirja pandud direktiivist) kuni lõpuni (vahepeal võis küll toimuda pöördumisi meetodite poole, aga ka meetodi sisu täideti iga kord ridahaaval). On olemas palju ülesandeid, mida sellisel viisil pole võimalik lahendada, käesolevas praktikumis vaatlemegi niisuguseid näiteid. Tingimusdirektiivi abil saab vastavalt andmete konkreetsetele väärtustele programmi täitmise kulgu suunata. Tingimusdirektiivi üldine kuju on

```
if (loogiline avaldis) direktiiv1 else direktiiv2
```

Tingimusdirektiiv algab sõnaga `if`, millele järgneb ümarsulgudes loogiline avaldis (avaldis, mille väärtus saab olla kas tõene või väär) ja sellele omakorda direktiiv, mis täidetakse siis, kui avaldis on tõene. Sõnale `else` aga järgneb direktiiv, mis täidetakse siis, kui loogiline avaldis on väär. Ühe direktiivi rollis võib olla ka plokk (programmilõik looksulgude `{}` vahel). Tingimusdirektiivi `else`-osa võib ka üldse puududa:

```
if (loogiline avaldis) direktiiv1
```

Ülesanne 1. Kehamassiindeks

Alustamegi variandist, kus `else`-osa pole. Täiendage eelmise praktikumi kehamassiindeksi arvutamise programmi nii, et juhul, kui indeks on suurem kui 27, väljastatakse näiteks selline teade: „Teie pikkus on liiga väike!“.

Ülesanne 2. Absoluutväärtus

Koostame meetodi etteantud täisarvu absoluutväärtuse arvutamiseks. Meetodi parameetriks olgu üks täisarv, meetod tagastab selle arvu absoluutväärtuse.

Kasutades eelmises praktikumis omandatud teadmisi ja tingimusdirektiivi, võime meetodi kirja panna järgmiselt:

```
static int absväärtus(int x) {
    if (x < 0) return -x;
    return x;
}
```

Kui avaldis `x < 0` on tõene, siis tagastatakse parameetri `x` vastandväärtus ja väljutakse meetodist, st tingimusdirektiivile järgnevat direktiivi ei täideta. Kui vaadeldav loogiline avaldis on väär, siis täidetakse tingimusdirektiivile järgnev direktiiv, st tagastatakse parameetri `x` väärtus ja väljutakse meetodist.

Praegusel juhul võib viimase naasmisdirektiivi esitada ka sellele eelneva tingimusdirektiivi `else`-osana:

```
static int absväärtus(int x) {
    if (x < 0) return -x;
    else return x;
}
```

Järgnevas on toodud mõned näited selle meetodi rakendamisest. Enne kui programm tööle panna, tuleks mõtiskleda, milline on muutuja `x` väärtus pärast iga omistamisdirektiivi.

```
x = absväärtus(-5);
x = absväärtus(10);
x = absväärtus(0);
```

Kopeerige meetod `absväärtus` oma programmi ja koostage peameetod, mis kontrollib meetodi `absväärtus` tööd arvude -5, 10 ja 0 korral. Kas tulemused olid sellised nagu ootasite?

Lisage programmi analoogiline meetod `absväärtusDouble` reaalarvu absoluutväärtuse leidmiseks ja kontrollige ka selle töötamist erinevate argumentide puhul.

Tegelikult ei pea nende kahe meetodi nimed olema erinevad, ka teisele meetodile võib panna nimeks `absväärtus`, küll aga peab erinevus olema formaalsete parameetrite tüübis (antud juhul `int` ja `double` ongi erinevad). Sellist olukorda nimetatakse *üledefineerimiseks*. Vaadake Java APIst, mitu absoluutväärtuse arvutamiseks mõeldud meetodit on olemas klassis `Math`.

Teema 2. Tingimusdirektiivid üksteise sees

Vahel on vaja eristada programmi täitmisel rohkem kui kahte haru. Seda saab realiseerida mitme tingimusdirektiiviga üksteise sees, näiteks

```
if (tingimus1)
    if (tingimus2)
        // tingimus1 ja tingimus2 on mõlemad täidetud
        direktiivid
    else
        // tingimus1 on täidetud, aga tingimus mitte
        direktiivid
else
    // tingimus1 ei ole täidetud (tingimus2 ei mõju)
    direktiivid
```

Sellisel juhul tuleb silmas pidada, et `else` kuulub alati viimase sellise tingimusdirektiivi juurde, millel veel `else`-osa pole. Selguse mõttes on kasulik programmiteksti treppida.

Ülesanne 3. Kehamassiindeks kolme variandiga

Täiendage kehamassiindeksi arvutamise programmi nii, et vastavalt indeksi väärtusele väljastatakse vähemalt kolme liiki teateid (vt Terviseleht).

Ülesanne 4. Veerandid

Olgu antud punkti koordinaadid x ja y. Koostage meetod, mis vastavalt etteantud punktile tagastab sõne kujul kirjelduse, millises koordinaattasandi veerandis antud punkt paikneb. Kui punkt paikneb teljel, siis tagastatakse tekst „paikneb teljel“. Pange tähele, et see meetod ei pea väljastama teksti ekraanile, vaid tagastama sõnetüüpi väärtuse.

Alljärgnevas on toodud osa vastavast meetodist:

```
static String punktiPaiknemine(double x, double y) {
    if (x == 0 || y == 0) return "on teljel";
        else if (x>0 && y>0) return "I veerandis";
            else if (x<0 && y>0) return "II veerandis";
    return "Kusagil mujal"; // kui meetod on õigesti tehtud,
                            // siis siia kunagi ei jõuta
}
```

Antud meetodi kirjapanekul on kasutatud *liittingimusi* (nt esimeses tingimuslauses `x>0 && y>0`), mille puhul kontrollitakse korraga kahe tingimuse täidetust (vt Java programmeerimise aabits, vihik 2, lk 8). Samuti kasutatakse selles meetodis tingimusdirektiivi sisus omakorda tingimusdirektiive.

See lahendus pole aga veel täielik, sest ta ei arvesta juhtu, kui punkt asub kolmandas või neljandas veerandis. Lõpetada meetod `punktiPaiknemine`. Kirjutada kahe reaalarvulise parameetriga meetod `väljasta`, mis pöördub selle meetodi poole ning väljastab tema töö tulemuse koos selgitava tekstiga. Näiteks

```
Punkt koordinaatidega 2.0 ja 5.0 asub I veerandis.
```

Koostage peameetod, mille alguses väärtustatakse muutujad vähemalt viie punkti koordinaatidega nii, et programm väljastaks erinevad teated (kokku siis vähemalt kümme muutujat).

Teema 3. Loogiline tüüp. Kontrollimismeetod ja selle rakendamine

Eelmistes praktikumides rääkisime muutujatest ja nende väärtustest. Käesolevas praktikumis oleme käsitlenud uut liiki, *loogilist väärtust*, mis võib olla tõene või väär. Selle väärtuse säilitamiseks kasutatakse programmeerimiskeeles Java vastavalt *loogilist tüüpi* muutujat. Analoogiliselt reaalarvulise või täisarvulise meetodiga saame rääkida ka *loogilisest meetodist* ehk *kontrollimismeetodist.*

Ülesanne 5. Paarsus

Koostame meetodi, mis etteantud arvu korral kontrollib, kas on tegemist paarisarvuga või paaritu arvuga, ja tagastab vastava tõeväärtuse:

```
static boolean onPaaris(int x) {
    boolean b;
    if (x%2 == 0)
        b = true;
    else
        b = false;
    return b;
}
```

Paneme tähele, et võrdsust väljendab programmeerimiskeele Java loogilises avaldises kaks järjestikust võrdusmärki, analoogiliselt mittevõrdsus on `!=` , võrratus `<=` jne. Loogilist tüüpi näitab võtmesõna `boolean`. Võtmesõnad `true` ja `false` tähendavad vastavalt tõeväärtusi „tõene“ ja „väär“.

Seega, kui avaldise `x&2 == 0` väärtus on „tõene“, siis `b` väärtuseks saab „tõene“. Kui aga avaldise `x&2 == 0` väärtus on „väär“, siis `b` väärtuseks saab „väär“. Kokkuvõttes: muutuja `b` väärtuseks saab avaldise `x&2 == 0` väärtus (`true` või `false`). Programmilõik

```
if (loogiline avaldis)
    b = true;
else
    b = false;
```

on seega sisuliselt samaväärne omistamisega

```
b = loogiline avaldis;
```

Järelikult võime eelneva meetodi kirja panna kujul

```
static boolean onPaaris(int x) {
    boolean b;
    b = (x%2 == 0);
    return b;
}
```

või isegi kujul

```
static boolean onPaaris(int x) {
    return x%2 == 0;
}
```

Seda kontrollimismeetodit saab rakendada näiteks järgmiselt:

```
int a = 4;
if (onPaaris(a))
    System.out.println(a + " on paarisarv");
else
    System.out.println(a + " on paaritu arv");
```

Kirjutage peameetod, mis rakendab kontrollimeetodit `onPaaris` erinevate täisarvude puhul.

Teema 4. Programmi testimine ja silumine

Mitmes käesoleva praktikumi ülesandes tuli jälgida programmi tööd erinevatel algandmetel. Sellist tegevust, programmi täitmist erinevate algandmete korral, nimetatakse programmi *testimiseks*. Testimist, testimistulemuste analüüsi ning selle põhjal programmist vigade otsimist ja nende parandamist nimetatakse programmi *silumiseks*. Silumine on sama tähtis (või isegi tähtsam) kui programmi esialgne kirjapanek, sest kindluse, et kirjapandu on õige, saame ainult programmi tööd jälgides.

Ülesanne 6. Punkti paiknemine

Koostame testandmete komplekti meetodi `punktiPaiknemine` testimiseks.

Antud meetod võib anda kokku viit erinevat liiki vastuseid. Neid nimetatakse ka meetodi või programmi *alternatiivideks*. Seega saame antud meetodi puhul rääkida viiest alternatiivist, milleks on punkti paiknemine esimeses, teises, kolmandas ja neljandas veerandis ning paiknemine teljel. Järelikult koosneb antud meetodi testandmete minimaalne komplekt viie erineva punkti koordinaatidest, näiteks

1. A(1,5) – punkt paikneb I veerandis
2. B(-2,6) – punkt paikneb II veerandis
3. C(-7,-11) – punkt paikneb III veerandis
4. D(2, -6) – punkt paikneb IV veerandis
5. E(0,7) – punkt paikneb teljel (täpsemalt, y-teljel)

Lisaks testandmetele tuleb programmi testimisel ette valmistada (läbi mõelda) ka meetodi oodatavad tulemused. Näiteks

- kui punkti A puhul on programmi töö tulemuseks vastus „I veerandis“, siis meetod töötab õigesti, mistahes muu vastuse korral töötab meetod valesti;
- kui punkti B puhul on programmi töö tulemuseks vastus „II veerandis“, siis meetod töötab õigesti, mistahes muu vastuse korral töötab meetod valesti jne.

Kui ka programmi töö tulemused on testidest lähtuvalt läbi mõeldud, võib anda testandmed programmile ette ja võrrelda saadud tulemusi plaanitud (ettevalmistatud) vastustega. Juhul, kui programmi käivitamisel saadud tulemused langevad oodatavatega kokku, siis on programm testid läbinud. Vastasel juhul tuleb vead välja selgitada, need parandada ja testimist korrata.

Kui meie meetod töötab õigesti kõigi ettevalmistatud testide korral, kas siis võib tema silumise lugeda lõppenuks? Veel mitte. Asi on selles, et meie testid on koostatud lähtuvalt meetodi struktuurist st. alternatiividest, mis on olnud meetodi kirjapanemise aluseks. Selline lähenemine on otstarbekas silumise algul, sest testis saadud vale tulemus aitab kindlaks määrata meetodi osa (näiteks tingimusdirektiivi haru), mida tuleb parandada või muuta. Kui aga meetodi struktuurist lähtuvad testid on kõik õige tulemuse andnud, tuleb siiski lisaks koostada veel mõned testid.

Oletame, et testija ei tea midagi meetodi struktuurist. Sel juhul kontrollib ta näiteks lisaks veel seda, kas meetod töötab õigesti järgmiste punktide korral:

6. F(0,0)
7. G(7,0)

Punkt F on koordinaatide alguspunkt ja G paikneb x-teljel (y-teljel paiknemist kontrollib punkt E). Üldiselt peaks silumisel kontrollima meetodi tööd võimalikult paljude eri liiki algandmetega. Ideaaljuhul võiks läbi kontrollida algandmete kõikvõimalikud väärtused. See pole aga enamasti võimalik. Seetõttu püütakse kõikvõimalikud algandmed jagada klassideks (paikneb koordinaatide alguspunktis, paikneb telgedel, paikneb kindlas veerandis jne) ja testandmete komplekti valida igast klassist üks esindaja.

Millised võimalused on veel antud meetodi testandmete komplekti täiendamiseks?

Siluge ja testige meetodit `punktiPaiknemine` eespool kirjeldatud testikomplektiga. Täiendage testandmeid ja sõnastage programmi eeldatavad täitmistulemused.

Iseseisev töö

Ülesanne 7. Täringuvise

Koostage programm, mis modelleerib täringu viskamist ja väljastab silmade arvu koos märkusega, kas see on paaris või paaritu.

- Kasutage meetodit `onPaaris`
- Koostage meetod `täringuvise`, mis tagastab `int`-tüüpi juhusliku täisarvu 1, 2, 3, 4, 5 või 6.

Juhusliku reaalarvu leidmiseks võib kasutada meetodit `Math.random` ja ümardamiseks meetodit `Math.round` (vt. API veebileht). Tähele tuleb panna, et `Math.random` tagastab `double`-tüüpi arvu, kuid `Math.round` tagastab `double`-tüüpi argumendi puhul `long`-tüüpi täisarvu. Aitab tüübiteisendus `int a = (int)Math.round(...);`

- Koostage peameetod.

Viited samateemalistele materjalidele

- J. Kiho. Java programmeerimise aabits. Tartu, 2002, vihik 2, lk. 9.

V klass

Tsükkel

Teemad. *Korduvate tegevuste programmeerimine. Tsüklid. For-tsükkel, while-tsükkel. Arvutuslikud meetodid.*

Pärast selle klassi läbimist üliõpilane

- tunneb ära ülesanded, mille programmeerimisel läheb vaja tsüklit;
- oskab valida sobivat tsüklidirektiivi, määrata tsükli eeltegevusi, jätkamistingimust ja sammu järeltegevusi;
- oskab tsükli tööd jälgida tsükli sisusse lisatud väljastuskäsu abil;
- on teadlik lõpmatu tsükli tekkimise võimalusest ja oskab seda vältida.

Praktikumijuhend

Teema 1. Tsüklidirektiivid programmeerimiskeeles Java

Sageli on vaja, et mingeid direktiive täidetaks korduvalt. Seda saab realiseerida tsüklite abil. Tsüklite programmeerimiseks on keeles Java kolm erinevat konstruktsiooni: kolmikpäisega tsükkel, eelkontrolliga tsükkel ja järelkontrolliga tsükkel.

Üldtsüklidirektiiv ehk kolmikpäisega tsükkel ehk for-tsükkel:

```
for (eeltegevused; jätkamistingimus; sammu järeltegevused) {
    // käsud, mida tuleb täita niikaua,
    // kui jätkamistingimus kehtib
}
```

Tüüpiliselt on *eeltegevusteks* mingitele muutujatele algväärtuste omistamised. Võimalikud eeltegevused on näiteks järgmised.

```
int i = 0;
```

Eeltegevused: kirjeldatakse täisarvutüüpi muutuja `i`, mis on selle tsükli lokaalne muutuja (st muutuja `i` väärtus ei ole väljaspool tsüklit kasutatav) ja omistatakse sellele algväärtus `0`.

```
kordaja = 0.5;
```

Eeltegevus: omistatakse reaalarvutüüpi muutujale `kordaja` algväärtus `0.5`. Siin ei ole kordaja tsükli lokaalne muutuja, vaid eeldatakse, et see muutuja on kirjeldatud juba enne tsükli täitmist (tüüp on `double` või `float`).

```
int i = 0, summa = 0;
```

Eeltegevused: kirjeldatakse täisarvutüüpi tsükli lokaalmuutuja `i` ja omistatakse sellele algväärtus `0`; varemkirjeldatud muutujale `summa` omistatakse algväärtus `0`.

Jätkamistingimus tuleb seada nii, et tsüklit täidetaks täpselt vajalik arv kordi. Näiteks kui soovime, et tsüklit täidetaks kolm korda, võib tsükli eeltegevusena omistada mingile lokaalmuutujale (tsükliloendajale) algväärtuseks nulli ja igal sammul liita tsükliloendajale ühe. Pärast tsükli esimest sammu on tsükliloendaja väärtus siis 1, pärast teist sammu 2 ja pärast kolmandat sammu 3. Neljandat sammu me enam lubada ei tohi, seega peaks tsükli lõpetama niipea, kui tsükliloendaja saab võrdseks 3-ga või 3-st suuremaks. Jätkamistingimus on lõpetamistingimuse vastandtingimus, ehk antud juhul võib tsüklit jätkata nii kaua, kui tsükliloendaja on veel väiksem kolmest.

Olukorda kirjeldava for-tsükli päis on järgmine:

```
for (int i = 0; i < 3; i = i + 1)
```

Tsükli *sammu järeltegevuseks* on sageli mingi muutuja väärtuse suurendamine või vähendamine teatud arvu võrra. Näiteks

```
i = i + 1;
j = j + 2;
k = k - 1;
```

Kõiki neid omistamisi saab Java-keeles ka lühemalt kirja panna:

```
i++;
j+=2;
k--;
```

Kirjutame näiteks for-tsükli, mis väljastab ekraanile viis korda teksti `Tere!`.

```
for (int i = 0; i < 5; i++) {
    System.out.println("Tere!");
}
```

Kui tsükli sisu koosneb ühest käsust, siis võib loogelised sulud ka ära jätta.

Ülesanne 1. Ruudud

Kopeerige eeltoodud tsükkel peameetodisse ning proovige programmi tööd. Muutke programmi nii, et ta väljastaks ekraanile arvude -10, -9, -8, ..., 8, 9, 10 ruudud.

Eelkontrolliga tsükkel ehk while-tsükkel:

```
while (jätkamistingimus) {
    // Siia kirjutame käsud, mida tuleb täita niikaua,
    // kui tsükli päises olev jätkamistingimus kehtib.
}
```

See while-tsükkel on samaväärne for-tsükliga, millel puudub nii eeltegevus kui ka sammu järeltegevus, s.o

```
for ( ; jätkamistingimus; ) {
    // käsud
}
```

Järelkontrolliga tsükkel ehk do-while-tsükkel:

```
do {
    // Siia kirjutame käsud, mida tuleb täita niikaua,
    // kui jätkamistingimus kehtib.
}
while (jätkamistingimus)
```

Pange tähele, et erinevalt eelkontrolliga tsüklist täidetakse järelkontrolliga tsüklit alati vähemalt üks kord, sest jätkamistingimuse kontroll toimub alles pärast tsükli sisu täitmist.

Arvatavasti olete juba märganud, et programmeerimisülesannetel võib sageli olla mitu erinevat õiget lahendust. Sarnaselt saab kordust sisaldavaid ülesandeid lahendada mitmel viisil, kasutades erinevaid tsüklitüüpe.

Näide ühe ülesande lahendamisest erinevatel viisidel

Kirjutada tsükkel, mis leiaks arvude 1, ..., 10 summa. *Lahendus 1.* For-tsükkel:

```
int summa = 0;
for (int arv = 1; arv <= 10; arv++)
    summa = summa + arv;
    // Tsükli sisu koosneb ainult ühest käsust, {} on ära jäetud
System.out.println("Arvude 1..10 summa on " + summa);
```

Lahendus 2. While-tsükkel. Pange tähele, et tsükliloendaja suurendamine toimub tsükli kehas (for-tsükli puhul on vastav käsk tsükli päises):

```
int summa = 0;
int arv = 1;
while (arv <= 10) {
    summa = summa + arv;
    arv++;
}
System.out.println("Arvude 1..10 summa on " + summa);
```

Lahendus 3. Do-while-tsükkel. Pange tähele käskude järjekorda tsüklis:

```
int summa = 0;
int arv = 0;
do {
    arv++;
    summa = summa + arv;
}
while (arv < 10)
System.out.println("Arvude 1..10 summa on " + summa);
```

Lahendus 4. Teine variant for-tsükli abil. Tsükliloendaja kahaneb nüüd 10-st 1-ni (esimeses lahenduses kasvas 1-st 10-ni):

```
int summa = 0;
for (int arv = 10; arv > 0; arv--)
    summa = summa + arv;
System.out.println("Arvude 1..10 summa on " + summa);
```

Teema 2. Lõpmatud tsüklid. Käsud `break;` ja `continue;`

Tsükli kirjutamisel varitseb programmeerijat oht kirjutada tsükkel, mis töötab lõpmatult. Põhjus on selles, et kirjutati jätkamistingimus, mis jääb alati tõeseks, seda omakorda võib põhjustada sammu järeltegevuse ununemine (for-tsükli puhul on seda raskem unustada, sest järeltegevus kirjutatakse tsükli päisesse).

Näiteks eelmise ülesande lahendusest 2 saame lõpmatu tsükli, kui kustutame (või muudame kommentaariks) rea, milles suurendatakse muutujat `arv` ühe võrra:

```
int summa = 0;
int arv = 1;
while (arv <= 10) {
    summa = summa + arv;
    // arv++;
}
```

Siis jääb muutuja `arv` väärtuseks alati `1` ja jätkamistingimus alati tõeseks.

Tsükli töö saab „pikema jututa“ katkestada käsu `break;` abil, mis suunab programmi täitmisjärje antud tsüklile järgnevale käsule. Näiteks arvude 1, ..., 10 summat võiks leida ka nii:

```
int summa = 0;
int arv = 0;
while (true) {    // alati tõene, võib ka: for (;;) {
    arv++;
    if (arv > 10)
        break;
    summa = summa + arv;
}
```

On olemas ka käsk `continue;`, mis suunab programmi täitmisjärje tsükli järgmisele sammule, jättes antud sammul täitmata need käsud, mis jäid `continue;` ja tsüklit lõpetava looksulu vahele. Üldiselt aga ei peeta programmeerimises keset tsüklit välja hüppamist heaks stiiliks. Tegelikult saab käskude `break;` ja `continue;` kasutamist alati vältida, kirjutades tsükli ümber, s.o muutes jätkamistingimust ja sisu.

Teema 3. Kahekordsed tsüklid

Tsükli sisuks võib olla ka teine tsükkel. Olgu meil näiteks vaja väljastada arvude 1, ..., 10 korrutustabel. Oletame, et tühi tabel rea- ja veerunumbritega on juba olemas:

×	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

Mõtleme, kuidas me tavaliselt toimime, kui hakkame tabelit täitma. Tõenäoliselt teeme seda reakaupa (või veerukaupa). Niisiis, võtame ette 1. rea ja hakkame selle lahtritesse arvutuste tulemusi kirjutama.

Vaatame 1. veeru kohal olevat arvu (1), korrutame reanumbri (1) sellega läbi ja kirjutame tulemuse (1) 1. rea 1. lahtrisse. Samuti toimime 2., 3., ..., 10. veeruga:

×	1	2	3	4	5	6	7	8	9	10
1	1	2	3	...						

Paneme tähele, et reanumber jäi kogu aeg samaks, aga veerunumber muutus ühest kümneni. See viib mõttele, et meil võiks olla „aeglane“ tsükkel, mis teeb oma järgmise sammu alles siis, kui selle sees olev „kiire“ tsükkel on kõik oma kümme sammu teinud. Keeles Java võib olukorra kirja panna järgmiselt:

```
for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 10; j++) {
        System.out.print(i*j + "\t");
    }
    System.out.println();
}
```

Ülesanne 2. Korrutustabel

Kopeerige see programmiosa tekstiredaktorisse, täiendage, kompileerige ja käivitage.

Mis rolli mängib sõne `"\t"`?

Mõelge/proovige, mis juhtuks siis, kui sisemises tsüklis olev väljastuskäsk oleks

- `System.out.println(i*j);`
- `System.out.print(i*j);`

Ülesanne 3. Paprikasupp

Kausitäis paprikasuppi jahtub minuti jooksul 19% võrra supi ja ruumi temperatuuride vahest. Koostage programm, mis väljastab supi temperatuuri minutiliste ajavahemike tagant, kui supi algtemperatuur on 90 kraadi. Ruumi temperatuur on 20 kraadi.

Looge klass `Supp` ja sellesse peameetod `main`, milles asub ka teie poolt kirjutatav tsükkel.

Hoiatus: olge ettevaatlik tsükli jätkamistingimusega! Kui lasete supil jahtuda 20 kraadini, peate väga kaua ootama. Mõelge ja/või proovige järele, miks.

Ülesanne 4. Ruutjuur

Üks lihtsaim praktikas kasutatav arvutusmeetod on meetod positiivse arvu u ruutjuure arvutamiseks. Kõigepealt leiame alglähendi

```
x = (u+1) / 2;
```

ning seejärel arvutame järk-järgult uusi lähendeid eeskirja

```
x = (x + u/x) / 2;
```

kohaselt.

Koostage klass `Ruutjuur`, milles on peameetod `main` ja ühe reaalarvulise parameetriga meetod `leiajuur`. Meetod

```
static double leiajuur(double u)
```

arvutagu ülalkirjeldatud eeskirja kohaselt etteantud arvu u ruutjuurt. Arvutamine lõpetada siis, kui järjekordse lähendi x ruut erineb arvust u vähem kui 0,0001 võrra. Meetod `leiajuur` väljastagu ekraanile ka kõik leitud lähendid.

Peameetodis katsetage meetodit `leiajuur` vähemalt kolme „tuntud“ arvuga (4, 25, 121 vms) ja kolme juhuslikult genereeritud arvuga.

Ülesanne 5. Kujundid

Kirjutage klass `Kujundid`, milles on kaks meetodit. Meetod `tärniraam` väljastab kuvarile sellise kujundi:

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

ja meetod `tärnikolmnurk` kujundi

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

Mõlemad meetodid kutsutakse välja peameetodist `main`. Esialgu võite kirjutada meetodid nii, et kujundi ridade arv on fikseeritud. Pärast muutke neid nii, et kujundi ridade arv on kummagi meetodi sisendparameetriks, s.o kirjutage meetodid

```
static void tärniraam(int ridadeArv) { ... }
static void tärnikolmnurk(int ridadeArv) { ... }
```

Iseseisev töö

Ülesanne 6. Tärniromb

Lisage klassi `Kujundid` veel üks meetod, mis joonistab järgmise kujundi, kasutades mitmekordset for-tsüklit. Ridade arv on meetodi sisendparameetriks (antud juhul on ridade arv 7). Kui ridade arv on paarisarv, siis võiks sellele liita ühe.

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

Järgnevas on antud mõned juhised, kuidas võiks ülesannet lahendada (loomulikult võite ka teistmoodi teha).

Algul võiks koostada sellise meetodi, mis väljastab ekraanile etteantud arvu tühikuid ja nende järel tärni. Äärmiste ridade puhul kasutada seda meetodit üks kord, teiste puhul kaks korda.

Olgu n ridade arv ja olgu ta paarisarv. Sellisel juhul on

- nullindas (i = 0) reas enne tärni n/2 tühikut ja siis tärn;
- esimeses (i = 1) reas n/2-1 (ehk n/2-i) tühikut, tärn, 1 (ehk 2*i-1) tühik, tärn;
- teises (i = 2) reas on n/2-2 (ehk n/2-i) tühikut, tärn, 3 (ehk 2*i-1) tühikut; tärn;
- kolmandas (i = 3) reas n/2-3 (ehk n/2-i) tühikut, tärn, 5 (ehk 2*i-1) tühikut, tärn;
- ...
- n/2-ndas (i = n/2) reas 0 (ehk n/2-i) tühikut, tärn, 2*i-1 tühikut, tärn;
- ja siis tagurpidi tagasi.

Viited samateemalistele materjalidele

- J. Kiho. Java programmeerimise aabits. Tartu, 2002, vihik 3, lk 11 – 12, vihik 4, lk 14.

VI klass

Järjendid I

Teemad. *Arvujärjendi loomine ja väljastamine. Elementide väärtuste muutmine. Järjenditöötlusmeetodi koostamine ja rakendamine.*

Pärast selle klassi läbimist üliõpilane

- tunneb ühemõõtmelise massiivi (järjendi) ülesehitust;
- teab, kuidas luua nullidest koosnev järjend;
- teab, kuidas luua konkreetne järjend massiivialgati abil;
- teab, kuidas omistada järjendi elementidele uusi väärtusi;
- tunneb ja oskab rakendada järjendit genereerivaid meetodeid;
- tunneb ja oskab rakendada meetodeid järjendi väljastamiseks;
- oskab koostada ja rakendada lihtsamaid meetodeid järjendi uurimiseks (ühekordses tsüklis üle järjendi).

Praktikumijuhend

Teema 1. Arvujärjendi loomine

Järjend aitab koondada teatavas mõttes ühetaolisi või ühesuguse tähendusega väärtusi ühte kohta. Niisugusteks väärtusteks võivad olla õhutemperatuurid järjestikustel päevadel, ühes rivis seisvate sõdurite pikkused, entsüklopeedia köidete lehekülgede arvud vms. Näiteks viieliikmeline järjend `a` koosneb Javas elementidest

```
a[0]
a[1]
a[2]
a[3]
a[4]
```

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 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`).

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. Näiteks korraldus

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

väljastab ekraanile järjendi `a` esimese elemendi väärtuse.

Ülesanne 1. Keskmine element

Koostage programm, mis väljastab ekraanile järjendi (asukoha mõttes) keskmise elemendi väärtuse. Selleks kirjeldage vähemalt 10-elemendiline järjend (soovitatavalt paaritu arvu elementidega) ja pange kirja sobiv väljastuskäsk.

Väljastatava elemendi indeks määrata käsitsi või, kes tunneb ennast kindlamalt, arvutada see välja otse programmis järjendi pikkuse (`length`) põhjal.

Teema 2. Arvujärjendi väljastamine

Järjendi kõigi väärtuste väljastamiseks tuleb terve järjend, alates algusest, elementhaaval läbida. Väga sobiv vahend selleks on for-tsükkel. Näiteks programmilõik

```
for (int i = 0; i < a.length; i++) {
    System.out.println(a[i]);
}
```

teeb just vajalikku tegevust – igal sammul, kui tsüklimuutuja `i` saab järjekordse väärtuse, trükitakse ekraanile järjekordse indeksiga element. Muutuja `i` algväärtus on `0` ja lõppväärtus ühe võrra väiksem kui järjendi pikkus. Indeksi nihke tõttu vaadatakse läbi parajasti kõik järjendi elemendid esimesest viimaseni. Näiteks 5-elemendilise järjendi puhul on:

`i` väärtus	ekraanile väljastatakse	märkus
`0`	`10`	`a[0]` väärtus
`1`	`9`	`a[1]` väärtus
`2`	`12`	`a[2]` väärtus
`3`	`11`	`a[3]` väärtus
`4`	`8`	`a[4]` väärtus
`5`		tsükli lõpp

Nagu eelnevast programmitekstist näha, võib elemendi indeksiks olla ka muutuja. Kui muutuja `i` on saanud mingi väärtuse, siis on arvutil lihtne kindlaks teha, millist elementi järjendis tähistab `a[i]`. Samamoodi võib indeksis esineda keerukam aritmeetiline avaldis, sel juhul arvutab arvuti kõigepealt välja avaldise väärtuse ja siis otsib järjendist üles vastava numbriga elemendi.

Ülesanne 2. Kapitalist

Kapitalist on paigutanud raha n firmasse. Nädala lõpus saadab sekretär talle kokkuvõtte, kus on kirjas, kuidas muutus iga firma aktsia hind selle nädala jooksul: kuipalju tõusis või langes. Andmed on salvestatud n-elemendilisse järjendisse, positiivne väärtus tähendab tõusu. Järgmisel nädalal soovib kapitalist suurendada oma positsiooni kõigis neis firmades, mille aktsia sel nädalal odavnes. Kirjutage programm, mis väljastab kõigi selliste firmade järjekorranumbrid ja vastavad aktsiahinna muutused.

Juhis: kasutada tingimusdirektiivi (if-lauset).

Teema 3. Elementide väärtuste muutmine

Järjendi läbivaatamisel võib loomulikult iga elemendiga teha keerukamaid operatsioone kui lihtsalt väärtuse väljastamine. Järgmine programmilõik arvutab järjendi elementide summa, järjendi elementideks on siinkohal kalendrikuude päevade arvud.

```
int[] a = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
int summa = 0;
for (int i = 0; i < a.length; i++) {
    summa += a[i];
}
```

Igal sammul liidetakse muutuja `summa` väärtusele juurde järjekordse elemendi väärtus.

Järjendi elemente võib programmis kasutada nagu tavalisi muutujaid. Ainuke erinevus seisneb selles, et elemente järjest läbi vaadates saab kõigi elementidega teha teatavaid ühtseid tegevusi. Näiteks kui tsükli sisus esineb käsk

```
a[i] = 2*i;
```

siis omistab programm järjendi `i`-nda elemendi väärtuseks tsükli parameetri kahekordse väärtuse, käsuga

```
a[i] = a[i-1] + a[i-2];
```

aga saab `i`-nda elemendi väärtus võrdseks kahe talle eelneva elemendi väärtuste summaga.

Ülesanne 3. Konveier

Mööda konveierit liiguvad detailid. Konveieri kohale on paigutatud kaks läbivalgustusseadet, mis, tõsi küll, pärinevad erinevatest aastatest ja on erineva jõudlusega. Esimene seade vaatab läbi iga teise detaili ning teine seade iga kolmanda detaili. Tööpäeva alguses pannakse konveier käima ja tööpäeva lõpus, pärast n-nda detaili läbilaskmist, peatatakse. Esimese detaili vaatavad läbi mõlemad seadmed. Mitu protsenti kõigist detailidest saab niimoodi läbi valgustatud ja mitu protsenti lipsab valgustamata läbi?

Näide. Oletame, et tööpäeva jooksul vaadati läbi 12 detaili, tähistame neid lihtsuse mõttes arvudega 0 kuni 11:

```
detailid konveierilt 0 1 2 3 4 5 6 7 8 9 10 11
1. seade valgustab  +   +   +   +   +   +
2. seade valgustab  +     +     +     +
```

Seega vaadati 12 detailist läbi 8 ehk 66,7%, vaatamata jäi 4 ehk 33,3%.

Juhis. Seda ülesannet saab lahendada järjendi abil või ilma. Kui teha järjendi abil (aga see on selle teema juures väga soovitatav), siis saab võtta kasutusele järjendi, mille elemendi väärtus 1 tähistab läbi valgustatud ja 0 valgustamata detaili. Järjend läbida kaks korda, kummalgi korral kirjutada sobivates kohtades 0 asemel 1. Lõpuks lugeda tulemusjärjendis kokku väärtuste 1 arv. (Või liita kummalgi korral sobivates kohtades 1 ja lõpuks lugeda kokku nullist erinevate väärtuste arv.)

Teema 4. Järjenditöötlusmeetodi koostamine ja rakendamine

Tihtipeale on kasulik järjendiga sooritatav operatsioon vormistada iseseisva meetodina, mida saab siis kasutada ka muude järjendite puhul. Näiteks summa arvutamiseks vajaminevad tegevused võime koondada meetodisse

```
static int summa(int[] a) {
    . . .
}
```

See meetod saab ette täisarvude järjendi ja annab vastuseks ühe täisarvu, järjendi elementide summa (võtmesõna `static` täpset tähendust vaatleme hiljem). Rakendades põhiprogrammis seda meetodit erinevatele järjenditele, arvutatakse iga kord vastava järjendi summa. Näiteks:

```
int[] a = new int[7];
int[] b = new int[5];
int[] c = {3, -1, 5, 6};
// a elementidele a[0], a[1], ..., a[6]
// ja b elementidele b[0], b[1], ..., b[4]
// omistatakse mingid väärtused:
. . .
// meetodi rakendamine:
int s1 = summa(a);
// nüüd on s1 = a[0] + a[1] + ... + a[6]
int s2 = summa(b);
// s2 = b[0] + b[1] + ... + b[4]
int s3 = summa(c);
// s3 = 13
```

Kui meetodi ülesandeks on lihtsalt teisendada antud järjendi elementide väärtusi, siis on tagastustüübiks harilikult `void`. Näiteks võime kirjutada meetodi, mis pöörab etteantava järjendi ümber:

```
static void vastupidi(int[] a) {
    . . .
}
```

Meetodi parameetriks on täisarvujärjend, meetodi toimel muutub selle elementide järjekord vastupidiseks. Näiteks järjendist

```
-1    6    -5    7    8    3
```

saame järjendi

```
 3    8     7   -5    6   -1
```

Meetodi sisus vahetame lihtsalt elementide väärtused: esimene ja viimane, teine ja eelviimane jne. Programmitekstina näeb see kirjeldus välja järgmiselt:

```
static void vastupidi(int[] a) {
    int n = a.length;
    for (int i = 0; i < n/2; i++) {
        int x = a[i];                    // vahetada a[i] ja a[n-1-i]
        a[i] = a[n-1-i];
        a[n-1-i] = x;
    }
```

Teinekord tuleb koostada ka meetodeid, mis etteantud järjendile vastuseks tagastavad terve uue järjendi. Näiteks võime kirjutada meetodi, mis pöörab etteantud järjendi ümber uueks järjendiks, jättes antud järjendi muutmata:

```
static int[] vastupidi2(int[] a) {
    int[] b = new int[a.length];         // abijärjend
    for (int i = 0; i < a.length; i++) {
        b[i] = a[a.length-1-i];          // kopeerida
    }
    return b;                            // tagastada massiivi viit
  }
```

Erinevus eelmisest meetodist on see, et siin säilib lähtejärjend esialgsel kujul.

Ülesanne 4. Pascali kolmnurk

Pascali kolmnurk on arvutabel

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5  10  10   5   1
    . . . . . . . . . .
```

kus iga arv on kahe tema kohal asuva arvu summa.

Kirjutage programm, mis arvutab Pascali kolmnurga mingi rea põhjal järgmise rea. Programm peab sisaldama kahte meetodit: `uusRida`, mis tagastab argumendiks antud rea järgi arvutatud uue rea, ning `väljasta`, mis väljastab argumentjärjendi.

Juhised. Meetodi `uusRida` puhul looge uus järjend, mis on argumentjärjendist ühe võrra pikem. Nullinda ja viimase elemendi väärtused on ühed. Ülejäänud arvutatakse argumentjärjendi kahe elemendi summana.

Meetodis `väljasta` võib kasutada tsüklit, mis väljastab üksteise järel argumentjärjendi elemendid.

Vabatahtlik lisaülesanne. Täiendada programmi nii, et ta trükiks ekraanile Pascali kolmnurga 10 esimest rida.

Pascali kolmnurga üks tähendus on see, et tema read määravad avaldise x+y astmete kordajad. Näiteks

$$(x+y)^2 = x^2+2xy+y^2$$

$$(x+y)^3 = x^3+3x^2y+3xy^2+y^3$$

Kõrgemate astmete $(x+y)^4$, $(x+y)^5$ jne kordajaid on samuti lihtne välja kirjutada.

Iseseisev töö

Lahendusena esitatud programm peab olema korralikult loetav ja piisavalt kommenteeritud.

Ülesanne 5. Kohtunikud

Mitmel spordialal kasutatakse võistleja esinemise hindamiseks järgmist süsteemi. Kõigepealt hindab võistleja esinemist n kohtunikku. Seejärel eemaldatakse hinnete hulgast madalaim ja kõrgeim ning võistleja hindeks loetakse ülejäänud hinnete aritmeetiline keskmine. Kui madalaima (või kõrgeima) hinde panevad mitu kohtunikku, siis eemaldatakse ainult üks selline hinne. Koostage programm, mis leiab võistleja hinde.

Juhis. Kui kõrgeim ja madalaim hinne on leitud, siis võib need lihtsalt summast lahutada, keskmise leidmisel tuleb samuti hinnete arvu kahe võrra vähendada.

Ülesanne 6. Auto

Punktide A ja B vahelisele teele, mille pikkus on d kilomeetrit, on paigaldatud n kiirust piiravat liiklusmärki. Mööda seda teed sõidab auto, mille suurim kiirus on x kilomeetrit tunnis. Kui suur on minimaalne aeg tundides, millega auto võib jõuda punktist A punkti B ilma, et seejuures oleks vaja liikluseeskirju rikkuda?

Info keelumärkide kohta on järgmine. Võib eeldada, et märgid on loetletud kauguse kasvamise järjekorras:

1. keelumärgi kaugus punktist A, maksimaalne lubatud kiirus
2. keelumärgi kaugus punktist A, maksimaalne lubatud kiirus
3. keelumärgi kaugus punktist A, maksimaalne lubatud kiirus

. . . . . .

n-nda keelumärgi kaugus punktist A, maksimaalne lubatud kiirus

Maksimaalne lubatud kiirus enne esimest keelumärki on 90 kilomeetrit tunnis.

Juhis. Üks võimalus on kasutada kahte järjendit, millest ühes hoida kaugusi ja teises vastavaid maksimaalseid lubatud kiiruseid.

Viited samateemalistele materjalidele

Järjendite ja nende töötlemise kohta võib teavet saada järgmistest allikatest.

- J. Kiho. Java programmeerimise aabits. Tartu, 2002, vihik 4.
- J. Kiho. Väike Java leksikon. Tartu, 2002, märksõnad *massiiv*, *massiivialgati*, *muutujakirjeldus*.

VII klass

Järjendid II

Teemad. *Arvujärjendi teisendamine pikkust muutmata. Arvujärjendi mitmekordne läbivaatus ja sorteerimine. Arvujärjendi(te) põhjal uue järjendi loomine.*

Pärast selle klassi läbimist üliõpilane

- oskab koostada ja rakendada lihtsamaid meetodeid arvujärjendi elementide väärtuste teisendamiseks (ühekordses tsüklis üle arvujärjendi);
- oskab koostada ja rakendada lihtsamaid meetodeid järjendi mitmekordset läbivaatust nõudvate ülesannete lahendamiseks;
- oskab koostada ja rakendada meetodeid arvujärjendi elementide väärtuste ümberpaigutamiseks; tunneb järjendi sorteerimise pistemeetodit;
- oskab koostada ja rakendada järjenditöötlemise meetodeid, kus tulemuseks on (uus) järjend.

Praktikumijuhend

Teema 1. Järjendi teatava elemendi leidmine

Antud on teatava asutuse töötajate palgad. Leida neist maksimaalne.

Hakkame järjest järjendi elemente läbi vaatama, pidades meeles seni läbitud elementide hulgast maksimaalset. Maksimaalse palga väärtust hoidva muutuja algväärtuseks võime võtta arvu 0, sest loomulik on eeldada, et kõik palgad on mittenegatiivsed (ei saa võtta järjendi esimest liiget, sest võib juhtuda, et meetodile antakse ette tühi järjend). Seega võime maksimaalse palga leida järgmise programmilõiguga:

```
int a[] = {1300,6700,2398,10568,2370,2370,4900,4503,2100};
int max = 0;          // palgad ja maksimaalse palga algväärtus
for (int i = 0; i < a.length; i++)
    if (max < a[i])   // kui leiame senisest suurimast suurema,
        max = a[i];   // siis jätame selle meelde
System.out.println("Suurim palk on " + max + " krooni");
```

Teema 2. Kõrvutiasuvate elementide võrdlemine

Ühes reas istub teatav arv inimesi (vähemalt kaks), kelle palkasid kirjeldab järjend `a`. Leida absoluutväärtuselt suurim erinevus selle järjendi kahe naaberelemendi vahel.

Võrreldes eelmise ülesandega, ei otsita siin enam mingi ühe elemendi väärtust, vaid suurust, mis sõltub kahest järjendi liikmest. Kui eelmises ülesandes osutas tsüklimuutuja `i` ainult ühele järjendi liikmele, siis siin peame muutuja `i` iga väärtuse korral käsitlema korraga kahte järjendi liiget `a[i]` ja `a[i+1]`. Muutuja `i` algväärtus on `0`, lõppväärtus aga selline, mille korral `i+1` on võrdne arvuga `a.length-1` ehk siis `a.length-2`. Analoogselt eelmise ülesandega peame meeles senileitud vahedest suurimat. Selle algväärtuseks võime taas valida arvu `0`.

```
int a[] = {1300,6700,2398,10568,2370,2370,4900,4503,2100};
int max = 0;
for (int i = 0; i < a.length-1; i++)  // esimesest eelviimaseni
        // kui leiame senisest suurimast vahest suurema,
        // siis jätame selle meelde
    if (max < Math.abs(a[i]-a[i+1]))
        max = Math.abs(a[i]-a[i+1]);
System.out.println("Suurim vahe on " + max + " krooni");
```

Teema 3. Tingimus sõltub kõigist elementidest

Antud on teatava asutuse töötajate palgad. Leida keskmine palk ja keskmisest väiksema palgaga töötajate arv.

On selge, et keskmine palk sõltub kõigist palkadest. Keskmise leidmiseks tuleb seega kõik järjendi elemendid läbi vaadata, alles siis saame kindlaks teha, millised elemendid on keskmisest väiksemad. Niisiis läbime palkasid väljendavat järjendit kaks korda, esimesel korral leiame elementide summa ja arvutame keskmise palga väärtuse, teisel korral loeme kokku sellest väiksemad elemendid:

```
int a[] = {1300,6700,2398,10568,2370,2370,4900,4503,2100};
int summa = 0;      // palkade summa
int väikesi = 0;    // keskmisest väiksemate arv
for (int i = 0; i < a.length; i++)      // leiame summa
    summa += a[i];
double keskmine = summa*1.0 / a.length; // leiame keskmise
                    // kui 1.0 ära jätta, mis siis juhtub?
for (int i = 0; i < a.length; i++)  // keskmisest väiksemaid
    if (a[i] < keskmine)
        väikesi++;
System.out.println("Keskmisest vaiksemaid palku on " +
                                    väikesi + " inimesel");
```

Ülesanne 1. Teine Eesti

Koostage eelmise programmilõigu järgi programm, mis lisaks keskmisest väiksemate palkade arvule väljastab ka nende väärtused.

Ülesanne 2. Järjestikused naturaalarvud

Antud on n naturaalarvu. Koostage programm, mis selgitab, kas nende seas on olemas kõik arvud 1, 2, ..., n, järjekord ei ole oluline. Iga leitud uue arvu puhul väljastagu programm ka selle väärtuse.

Näiteks järjendis 1, 2, 4, 3 on olemas kõik arvud 1-st 4-ni, järjendis 1, 2, 4, 5 pole kõiki arve 1-st 4-ni, järjendis 1, 1, 2 pole kõiki arve 1-st 3-ni.

Üks võimalus ülesannet lahendada on kontrollida iga arvu 1, 2, ..., n puhul, kas see arv sisaldub järjendis. Kui leidub arv, mida järjendis pole, siis saame kohe öelda, et lõppvastus on eitav. Kui ei leidu arvu, mida järjendis pole, siis on vastus jaatav. Koostame kahekordse tsükli, kus välimises tsüklis vaadatakse läbi kõik arvud 1, 2, ... n, sisemises aga kontrollitakse iga arvu puhul, kas see arv sisaldub järjendis (ehk siis otsitakse järjendist temaga võrdset arvu). Sisemise tsükli asemel võib teha ka eraldi meetodi, mille parameetriks on üks täisarv ning üks täisarvude järjend, meetod tagastab tõeväärtuse `true`, kui arv sisaldub järjendis, ja tõeväärtuse `false`, kui ei sisaldu.

Teine võimalus on kontrollida, kas iga element esineb ainult ühekordselt ning kas kõik elemendid mahuvad 1 ja n vahele.

Teema 4. Kahekordset tsüklit nõudev ülesanne

Antud on tasandi punktide koordinaadid (x_1, y_1), ..., (x_n,y_n). Leida punktid, mis asuvad teineteisest kõige kaugemal. Väljastada ekraanile ka nende punktide koordinaadid.

Siin võib võtta kasutusele ühe järjendi x-koordinaatide jaoks ja teise järjendi y-koordinaatide jaoks. Näiteks `i`-nda punkti koordinaadid on siis `x[i]` ja `y[i]`. Kauguse leidmiseks saame kasutada meetodit kolmandast praktikumist.

Kontrollida tuleb iga punkti kaugust igast teisest punktist. Seda võib teha kahekordse for-tsükliga. Välimises tsüklis võiks indeks `i` muutuda 1-st kuni n-ni, igal välimise tsükli sammul arvutatakse sisemises tsüklis `i`-nda punkti kaugus `j`-ndast punktist, kus `j` on sisemise for-tsükli indeks.

```
for (int i = 0; i < x.length-1; i++)
// sisemises tsüklis vaatame läbi kõik i-le järgnevad punktid
    for (int j = i+1; j < x.length; j++) {
        //  leiame kauguse kahe punkti vahel
        double kaugus = Math.sqrt((x[i]-x[j])*(x[i]-x[j]) +
                                  (y[i]-y[j])*(y[i]-y[j]));
        //  kui võrreldavate punktide kaugus on suurem kui seni
        //  leitud rekord, siis jätame meelde punktid ja kauguse
        if (kaugus > maxkaugus) {
            esimpunkt = i;
            teinpunkt = j;
            maxkaugus = kaugus;
        }
    }
```

Pärast selle programmilõigu täitmist on muutujate `esimpunkt` ja `teinpunkt` väärtusteks kahe kõige kaugema punkti indeksid koordinaatide järjendites. Indeksite järgi leiame punktide koordinaadid ja väljastame need koos punktide vahelise kaugusega, mis on muutuja `maxkaugus` väärtuseks.

Ülesanne 3. Kahe astmed

Antud on täisarvuliste liikmetega järjend `a`. Koostage programm, mis leiab selles järjendis kõige pikema ainult arvu 2 astmetest koosneva lõigu. Näiteks järjendis

2, 4, 3, 1, 2, 4

on otsitavaks lõiguks kolmeelemendiline lõik järjendi lõpus. Programm peab väljastama ka leitud osajärjendi elementide väärtused.

Siin võib olla kasu meetodist, mis teeb kindlaks, kas antud arv on kahe aste või mitte.

Ülesanne 4. Arvutamine ei aita – peab sorteerima

Antud on asutuse kõigi töötajate palgad, kusjuures asutuses on paaritu arv töötajaid. Koostage programm, mis leiab sellise inimese järjekorranumbri, kelle palk on kõigi hulgas keskmine: temast väiksemaid palku on sama palju kui suuremaid.

Sorteerimisest pistemeetodil on juttu Aabitsa vihikus 6. Veebis on saadaval ka vastav fail Jada1.java.

Iseseisev töö

Ülesanne 5. Sõdurid

Järjendisse on salvestatud eriväljaõppe saanud sõdurite pikkused. Eriti tähtsale operatsioonile saadetakse kaks kõige pikemat meest. Millised on nende pikkused?

Juhis. Järjendit pole vaja sorteerida, piisab järjendi ühekordsest läbivaatusest, igal sammul pidada meeles seni vaadeldud elementidest kaks suurimat (selleks võib kasutusele võtta kaks muutujat). Alguses oletada, et kaks suurimat on `a[0]` ja `a[1]`. Kui leidub element, mis on suurem kui vähemalt üks kahest senileitud suurimast elemendist, siis anda leitud väärtus neist kahest muutujast sellele, mille väärtus on väiksem. Miks nii saab?

Ülesanne 6. Indiaanlased

Indiaanlased liiguvad hanereas, nende pikkusi kirjeldab järjend. Mitmendal positsioonil selles reas asub indiaanlane, kelle ees (vahetult) asub kõige rohkem temast lühemaid indiaanlasi?

Juhis. Järjendit läbides peame meeles juba vaadeldud indiaanlaste seast „parima“ järjekorranumbrit ja seda, mitmest vahetult eelnevast inimesest ta pikem on. Leides iga indiaanlase korral lühemate eelkõndijate arvu, tuleb järjendis liikuda näiteks while-tsükliga ettepoole niikaua, kui järjendi liikmete väärtused on vaadeldavast väärtusest väiksemad, ja lugeda kokku selliste väärtuste arv.

Viited samateemalistele materjalidele

- J. Kiho. Java programmeerimise aabits. Tartu, 2002, vihik 6.

VIII klass

Andmed käsurealt

Teemad. *Andmete lugemine käsurealt. Sõne teisendamine arvuks. Käsureaandmete järjendi töötlemine.*

Pärast selle klassi läbimist üliõpilane

- teab, et käsurea argument on tüüpi `String`;
- oskab kasutada peameetodi parameetrit;
- oskab kasutada meetodeid sõnena esitatud andmete teisendamiseks sobivasse algtüüpi (meetodid `parseDouble`,`valueOf`,`doubleValue`).

Praktikumijuhend

Teema 1. Andmed käsurealt

Seni oleme käivitanud oma programme käsuga

```
java KlassiNimi
```

mille toimel hakatakse täitma vastava klassi peameetodit. Tegelikult ei erine peameetod teistest meetoditest kuigivõrd, tal on olemas ka formaalne parameeter, mille tüüp on `String[]` (sõnede järjend). Programminäidetes on sellele antud sageli nimi `args`, aga valida võib ükskõik millise nime, kui see ainult järgib muutujanimedele kehtestatud reegleid. Peameetodi parameetrit kasutades saame soovitud sisendandmeid programmile käsurealt ette anda.

Käivitades programmi näiteks käsuga

```
java KlassiNimi väärtus1
```

saab sõnejärjend `args` pikkuseks ühe ja tema ainus element omandab väärtuse

```
args[0] = "väärtus1";
```

Et järjendi `args` elemendid on sõned, siis on loomulikult kõik käsurealt saadud andmed programmis tüüpi `String`.

Ülesanne 1. Väljastamine

Kolme käsurealt argumendina saadud sõne väljastamise programm võiks välja näha järgmiselt:

```
class Kasurealt {
    public static void main(String[] args) {
        System.out.println("Sisestasid: ");
        for(int i = 0; i < 3; i++) {
            System.out.println(args[i]);
        }
    }
}
```

Kopeerige see programm ja käivitage teda mitu korda, iga kord erineva arvu argumentidega. Milline on tulemus?

Et käsureal eraldab argumente tühik, siis trükib eeltoodud programm ka nelja ja enama argumendi korral ikka esimesed kolm sõnet. Mis aga toimub väiksema argumentide arvu korral?

Kui soovime ette anda argumendi, mis ise sisaldab tühikuid, siis tuleb kogu sisendsõne ümber lisada jutumärgid. Näiteks sisendi

```
java Kasurealt "meil on täna pannkoogipäev!" "eile sai kala"
```

korral on massiiv `args` kaheelemendiline ja tema elementide väärtused on

```
args[0] = "meil on täna pannkoogipäev!";
args[1] = "eile sai kala";
```

Muutke eeltoodud meetodit nii, et ta väljastaks ekraanile kõik käsurealt argumentidena saadud sõned. (Siin võib olla abiks muutuja `args.length`).

Teema 2. Sõnejärjend

Nagu näha, võib järjend koosneda ka sõnedest. Kui on vaja luua sõnejärjendit programmi sees, siis toimub nii järjendi loomine kui ka tema elementidele väärtuste samamoodi kui arvujärjendite korral:

```
String[] sõned = {"Ilus", "punane", "maasikas"};
```

või

```
String[] sõned = new String[pikkus];
```

Teema 3. Sõnest arvuks

Sageli ei piisa käsurea-andmete lihtsast lugemisest/kirjutamisest, vaid neid on vaja kasutada arvutuste tegemisel. Siin aga tekib konflikt erinevate andmetüüpide vahel – nagu juba varasematest praktikumidest teada, ei saa arvutüüpi muutujale omistada sõnetüüpi väärtust. Sellepärast oleks vaja mingeid vahendeid numbritest koosneva sõne teisendamiseks sobivat tüüpi (`int`, `double` jt) arvuks. Uurides Java API veebilehte, leiame, et on olemas klassid, mille nimi sarnaneb enam või vähem vastava algtüübi nimega (`Integer`,`Double` jt). Neid klasse nimetatakse *mähisklassideks* ja nad koosnevad hulgast meetodeist, sealhulgas ka meetodeist sõne teisendamiseks arvuks. Näiteks klassist `Double` leiame meetodi `parseDouble`, mis kontrollib, kas argumendiks antud sõne koosneb sobivaist sümboleist ning jaatava vastuse korral tagastab sõnele vastava `double`-tüüpi arvu. Seega näiteks sõne `"1234"` teisendamise tulemuseks saame arvu `1234.0`.

Proovige, mis juhtub, kui teisendatav sõne sisaldab ka muid sümboleid peale numbrite (ja punkti)?

Analoogilised meetodid eksisteerivad ka teistes mähisklassides. Näiteks sõne kujul antud täisarvu teisendab `int`-tüüpi väärtuseks klassi `Integer` meetod `parseInt`.

Ülesanne 2. Kehamassiindeks

Modifitseerige varemtehtud kehamassiindeksi ülesannet (seda, kus on vähemalt kolm reaktsiooni) selliselt, et käsurea argumentidena antakse ette inimese nimi, kehamass (täisarvuna) ja pikkus, näiteks

```
java kehamass Juhan 90 1.80
```

Pärast käsurea andmete töötlemist on sõnejärjendis nüüd kolm elementi. Neist esimest (indeksiga 0) soovime edaspidi kasutada sõnena, kahte viimast aga arvudena. Vajaliku teisenduse saab teha näiteks käsuga

```
int mass = Integer.parseInt(args[1]);
```

Analoogiliselt teisendame arvuks ka pikkuse:

```
double pikkus = Double.parseDouble(args[2]);
```

Uues programmis muutke väljastamist nii, et programm pöörduks otse konkreetse inimese poole:

```
Juhan, Teie kehakaal on normis.
```

Teema 4. Suurem hulk argumente

Oleme näinud, et käsureal võib argumente olla enam kui üks. Kui argumentide arv on ette teada, siis saab nende arvuks teisendamisel kasutada senist taktikat ehk teisendada kõik ükshaaval:

```
int x1 = Integer.parseInt(args[0]);
int x2 = Integer.parseInt(args[1]);
double z = Double.parseDouble(args[2]);
```

Ent kui argumentide arv on suur, pole see otstarbekohane. Peale selle võib argumentide arv olla fikseerimata. Niisugusel juhul on soodsam luua sobivat tüüpi massiiv ja tsüklis omistada selle igale elemendile üks teisendatud väärtus. Uue massiivi elementide arvuks sobib massiivi `args` pikkus. Näiteks programmilõigus

```
double[] jada = new double[args.length];
for (int i = 0; i < jada.length; i++) {
    jada[i] = Double.parseDouble(args[i]);
}
```

teisendatakse `i`-ndal tsüklisammul massiivi `args` element `args[i]` arvuks ja omistatakse tulemus massiivi `jada` elemendi `jada[i]` väärtuseks.

Ülesanne 3. Aktsiad

Muutke üle-eelmise praktikumi aktsiahindade programmi nii, et hinnamuutused antakse ette käsurealt ja ekraanile väljastatakse langenud hinnaga aktsiate numbrid ning hinnamuutuste suurused.

Ülesanne 4. Kasutusjuhend

Programmi kasutaja reeglina ei tea väga täpselt, kuidas programm töötab. Seepärast on hea, kui programmi koostaja on talle võimalikult üksikasjalised juhtnöörid jätnud. Täiendage üht eelnevatest programmidest nii, et ilma argumentideta või sobimatu arvu argumentidega käivitamisel ilmuks ekraanile programmi kasutusjuhend.

Iseseisev töö

Ülesanne 5. Ise

Koostage üks programm, mis töötab käsurea argumentide järgi ja väljastab töö lõpptulemuse ekraanile. Valige ise ülesanne ning realiseerige see programmina, arvestusega, et Te peaksite selle ülesande puhul programmeerimisele pühendama 30–45 minutit.

Viited samateemalistele materjalidele

- J. Kiho. Java programmeerimise aabits. Tartu, 2002, vihik 3, lk 12-14.