Harjutustund 2

Eclipse. Aritmaatirilised operaatorid. Tingimusdirektiivid. Võrdlusoperaatorid. Loogilised operaatorid. Tsüklidirektiivid.

Eclipse

- Arvutiklassis peaks *Eclipse*'i ikoon olema töölaual. (Kui pole, siis leidke kataloog, kus on uusim *Eclipse*.) Sülearvutites peaks *Eclipse* olema `C` kettal.
- Töökataloogiks määrake nt. alamkataloog `oop`.
- Tervitusaknad võite sulgeda, aga võite ka neid soovi korral lugeda.
- Valige Window --> Open Perspective --> Others ...--> Java(default) või ärge valige midagi, sest Java ongi vaikimisi.
- Looge uus *Java* projekt File --> New --> Java Project või File --> New --> Project ... --> Java Project
- Pange projektile mõistlik nimi. Esialgu võiks aktsepteerida vaikimisi vastuseid loomisdialoogis.
- Nüüd peaks *Package Explorer* aknas olema teie loodud projekt olemas. Kui sellel hiire parema klahviga klõpsata, siis avaneb menüü.
- Looge uus klass: New --> Class, nimi alaku suurtähega. Klassi loomise dialoogis

    - jätke *Package* tühjaks,
    - *Modifiers* olgu *public*,
    - *Superclass* olgu *java.lang.Object*,
    - loodavatest meetoditest märgime ära *public static void main(String[] args).*

- Toimetiaknas on nüüd teksti toorik.
- Kirjutage *main*-meetodisse midagi mõistlikku, mis ka midagi ekraanile väljastab, ja salvestage.
- Käivitage Run --> Run või Run --> Run As --> Java Application
- Tulemus peaks ilmuma konsooliaknasse.

Aritmeetilised operaatorid

```
Operaator       Tulemus
--------------------
   +               Liitmine
   -               Lahutamine
   *               Korrutamine
   /               Jagamine
   %               Täisarvulise jagamise jääk
   ++              Inkrement (suurendamine ühe võrra)
   +=              Omistamine liitmisega
   -=              Omistamine lahutamisega
   *=              Omistamine korrutamisega
   /=              Omistamine jagamisega
   %=              Jagatise jäägi omistamine
```

```
   --              Dekrement (vähendamine ühe võrra)
```

Tingimusdirektiivid

Tingimusdirektiivi üldine kuju *Javas* on

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

```
if (loogiline avaldis)
    {
     direktiivid1
    }
else
    {
     direktiivid2
    }
```

Võrlusoperaatorid

```
Operaator       Tulemus
-------------------------
 ==              Võrdne
 !=              Mittevõrdne
 >               Suurem
 <               Väiksem
 >=              Suurem-võrdne
 <=              Väiksem-võrdne
```

Loogilised operaatorid

```
Operaator       Tulemus
----------------------------------------
  &               Loogiline JA
  |               Loogiline VÕI
  ^               Loogiline MITTESAMAVÄÄRSUS
  ||              VÕI arvutamine lühiskeemi kohaselt
  &&              JA arvutamine lühiskeemi kohaselt
  !               Loogiline unaarne eitus
  &=              Loogiline JA ning omistamine
  |=              Loogiline VÕI ning omistamine
  ==              Võrdne
  !=              Mittevõrdne
  ?:              Valikuoperaator
```

Ülesanne 1

Koostada programm, mis teeb kindlaks, kas on tegemist paarisarvuga. Programmi töö lõpeb, kui kasutaja sisestab arvu null.

Ülesanne 2

Koostada programm soodustuse arvutamiseks. Kui ostusumma on suurem kui 1000 eur, siis soodus 10%.

Ülesanne 3

Koostada programm soodustuse arvutamiseks. Kui ostusumma on suurem kui 500 eur, siis soodus 3%, kui summa on suurem kui 1000 eur, siis 5%.

Tsüklidirektiivid

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 lokaalmuutujad `i` ja `summa` ning omistatakse neile algväärtused `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.

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.