LUKU 5

MUUTTUJAT JA MUUT JAVA-OHJELMIEN PERUSELEMENTIT

Nyt, lopultakin, aloitamme tietokoneohjelmoinnin opiskelun Java-ohjelmointikielellä. Muuttujat ovat oleellisia elementtejä tietokoneohjelmissa. Niitä tarvitaan käytännössä lähes kaikissa ohjemissa. Tässä luvussa tutustumme erityyppisiin muuttujiin ja käytämme niitä yksinkertaisissa aritmeettisissa laskutoimituksissa. Tulemme myös tutustumaan muihin ohjelmien peruselementteihin kuten nimiin ja varattuihin sanoihin.

Tämä luku, kuten myös tätä seuraavat luvut, esittelee sinulle joukon Java-kielisiä tietokoneohjelmia. Tietokoneohjelma on tekstuaalinen kuvaus siitä miten tietokoneen tulee toimia kun ohjelman käännetty versio suoritetaan tietokoneella. Tässä luvussa esiteltävät ohjelmat sisältävät ensin muuttujien esittelyjä ja näitä esittelylauseita seuraa joukko toimintolauseita (action statements) jotka tekevät jotain aiemmin esitellyille (määritellyille) muuttujielle. Näiden ohjelmien yleisrakenne on seuraavanlainen::

```
import java.util.* ;

class LuokanNimi
{
   public static void main( String[] ei_kaytossa )
   {
      Scanner nappaimisto = new Scanner( System.in ) ;

      Muuttujien esittelyt (määrittelyt).

      Toimintolauseet jotka muuntelevat esiteltyjen
      muuttujien arvoja ja tulostavat
      niitä sekä tekstejä näytölle.
   }
}
```

Nämä rivit esiintyvät vain ohjelmissa jotka lukevat tietoja näppäimistöltä.

Kuten jo luvussa 2 todettiin, ensimmäisten esimerkkieohjelmien rakenne on sellainen että niissä on staattinen metodi nimeltä **`main()`**, joka sisältää ohjelman oleelliset lauseet, ja tuo **`main()`**-metodi on kirjoitettu luokkamäärittelyn sisään. Tässä opiskelun vaiheessa emme yritä ymmärtää tuon luokkamäärittelyn merkitystä, vaan keskitymme **`main()`**-metodin sisään kirjoitettujen lauseiden ymmärtämiseen.

5.1 Integer-muuttujat (int, short, long, byte, char)

Muuttuja on lähdekielisissä ohjelmissa hyvin yleinen ohjelman peruselementti. Muuttujaan voidaan tallettaa numeerinen arvo. Yleensä muuttujan arvo muuttuu ohjelman suorituksen aikana. Kun ohjelmaan määritellään muuttuja, se tarkoittaa käytännössä että tietokoneen keskusmuistista varataan muutamia tavuja tiedon talletukseen. Seuraava on esimerkki muuttujan esittelystä ohjelmaan:

```
int  luku_nappaimistolta ;
```

Tällaista ohjelman lausetta voidaan kutsua joko muuttujan esittelyksi (declaration) tai muuttujan määrittelyksi (definition). Yllä oleva ohjelmarivi tarkoittaa että neljä tavua (32 bittiä) varataan keskusmuistista näppäimistöltä luettavan kokonaisluvun talletusta varten, ja näihin neljään tavuun voidaan viitata nimellä **`luku_nappaimistolta`**. Yllä määritellyn muuttujan tyyppi on **`int`**, joka on lyhennys englannin kielen sanasta "integer". Tyyppiä **`int`** olevat muuttujat voivat tallettaa sekä positiivisia että negatiivisia kokonaislukuja, mutta desimaalipisteellisiä lukuja niihin ei voida tallettaa.

Muuttujalla on aina jokin tyyppi kuten esim. **`int`**. Ohjelman kirjoittajan tulee antaa yksikäsitteinen nimi jokaiselle ohjelmaan määritellylle muuttujalle. Yllä olevassa muuttujan esittelylauseessa muuttujan nimi on **`luku_nappaimistolta`**. Muuttujien esittelylauseet, kuten kaikki Java-kielen lauseet, tulee päättää puolipisteeseen.

Ohjelma **Peli.java**, joka on esitelty seuraavalla aukeamalla, on esimerkkiohjelma jossa käytetään kahta **`int`**-tyypin muuttujaa. Kyseessä oleva ohjelma on erittäin yksinkertainen tietokonepeli. Pelaajan kannalta kyseessä on hiukan huono peli, koska tietokone voittaa pelissä aina. Ohjelma voittaa käyttäjänsä esittämällä aina yhtä suuremman luvun kuin käyttäjä.

Lähdekielinen ohjelma kuten **Peli.java** on tekstitiedosto tietokoneen kovalevyllä. Tuo tekstitiedosto voidaan kääntää Java-kääntäjällä suoritettavaksi ohjelmaksi, joka käännösprosessin aikana sijoitetaan omaan tiedostoonsa. Kääntäjä on itsekin tietokoneohjelma, ja se voi tuottaa uusia tietokoneohjelmia. Kääntäjä käsittelee lähdekielisen ohjelmatiedoston samassa järjestyksessä kuin tiedosto on kirjoitettu. Kun Java-kääntäjä prosessoi tiedostoa **Peli.java**, se "näkee" ensin muuttujamäärittelyt joiden perusteella se varaa muistia muutujille. Tämän jälkeen kääntäjä muuntaa ohjelman toimintolauseet numeerisiksi käskyiksi jotka prosessoidaan ohjelman suorituksen aikana.

Lähdekielisen ohjelman toimintolauseet (action statements) kuvaavat aktivitetteja joita tietokone suorittaan silloin kun ohjelman käännettyä versiota ajetaan tietokoneessa. Ohjelman toimintolauset suoritetaan siinä järjestyksessä kun ne on kirjoitettu ohjelman lähdekieliseen versioon. Ohjelman **Peli.java** tapauksessa tietokone suorittaa seuraavat aktiviteetit (toiminnot):

- Ensin se pyytää käyttäjää antamaan kokonaisluvun näppäimistöltä.
- Sitten se lukee kokonaisluvun näppäimistöltä ja tallettaa sen muuttujaan **`luku_nappaimistolta`**.
- Sitten se laskee arvon joka on yhtä suurempi kuin käyttäjän antama kokonaisluku-arvo, ja tallettaa tuon arvon muuttujaan **`yhta_suurempi_luku`**.
- Lopuksi se näyttää molempien muuttujien arvot ja informoi käyttäjää siitä että tietokone on voittanut pelin.

Peli.java-ohjelmassa on neljä toimintolausetta. Toimintolauseita edeltävät muuttujien esittelylauseet (variable declarations) ovat myös ohjelman lauseita, mutta ne eivät saa ohjelmassa mitään toimintaa aikaiseksi. Muuttujien esittelylauseet käytännössä vain varaavat muistia tiedon talletukseen. Kaikille ohejlman lauseille on yhteistä se että ne päättyvät puolipisteeseen.

Vaikka **int**-muuttujalle varattava muistitila on melkoisen suuri, 4 tavua, on olemassa tarkat rajat sille kuinka suuria arvoja **int**-muuttujaan voidaan tallettaa. Neljä tavua muistia omaava **int**-tyypin muuttuja voi tallettaa kokonaislukuarvoja seuraavilla arvoalueilla:

-2.147.483.648, ... , -1, 0, 1, ... , 2.147.483.647 (kymmenjärjestelmä)

-80000000H, ..., -1, 0, 1, ... , 7FFFFFFH (heksadesimaalijärjestelmä)

Neljätavuinen **int**-muuttuja voi siis tallettaa 4.294.967.296 (100000000H) erilaista numeroarvoa. Laskutoimituksissa voidaan muuttujan tallennuskapasiteetti ylittää. Tämän demonstroimiseksi ohjelma **Peli.java** on suoritettu ohjelmakuvauksessa myös siten että sille on annettu liian suuri input-arvo. Kun ohjelma yrittää kasvattaa yhdellä arvoa 2.147.483.647 joka on talletettu **int**-tyypin muuttujaan, tulokseksi saadaan negatiivinen arvo -2.147.483.648 eikä arvoa 2.147.483.648. Tämän ohjelman "kummallisen" käyttäytymisen selittämiseksi on muistettava että tietokoneiden muistit pystyvät tallettamaan vain ei-negatiivisia binaarilukuja. Negatiiviset kokonaisluvut esitetään tietokoneiden sisällä siten että tietyt positiiviset arvot käsitetään negatiivisina lukuina. Esimerkiksi lukuarvoa 2.147.483.648 pidetään negatiivisena lukuna -2.147.483.648. Seuraava lukuluettelo esittää arvot jotka käsitetään negatiivisiksi luvuiksi neljätavuisen (32-bittisen) **int**-muuttujan tapauksessa:

```
MUISTIIN TALLETETTU ARVO          ARVON MERKITYS OHJELMASSA

2.147.483.648 (80000000H)        -2.147.483.648
2.147.483.649 (80000001H)        -2.147.483.647
2.147.483.650 (80000002H)        -2.147.483.646
.                                 .
.                                 .
4.294.967.294 (FFFFFFFEH)        -2
4.294.967.295 (FFFFFFFFH)        -1
0                                 0
1                                 1
.                                 .
.                                 .
2.147.483.646 (7FFFFFFEH)         2.147.483.646
2.147.483.647 (7FFFFFFFH)         2.147.483.647
```

Kuva 5-1 esittää miltä ohjelman **Peli.java** muuttujat näyttävät tietokoneen keskusmuistissa, ja miten muuttujien arvot muuttuvat kun ohjelma suoritetaan syöttämällä sille arvo 1234. Muttuujan esittelylause kuten esimerkiksi

```
int  luku_nappaimistolta ;
```

varaa neljä tavua muistia perättäisistä muistiosoitteista keskusmuistissa. Ohjelmoijahan ei yleensä tiedä mistä kohtaa keskusmuistia tila varataan. Heti muuttujan esittelyn jälkeen noiden neljän tavun sisältö ei ole tiedossa. Kun esimerkiksi sijoituslause

```
luku_nappaimistolta  =  nappaimisto.nextInt() ;
```

suoritetaan, noille neljälle tavulle tulee arvot jotka edustavat näppäimistöltä syötettyä kokonaislukua.

Kuva 5-1 esittää yleistä periaatetta jonka mukaan muistia varataan muuttujille. Ohjelman kääntäjä taikka ohjelman ajoympäristö saattaa kuitenkin tehdä toimenpiteitä, joiden avulla muistin käyttöä optimoidaan eli muistia säästetään.

Tämä rivi ei varsinaisesti kuulu ohjelmaan. Tämä on ns. kommenttirivi joka antaa tietoa ohjelman lukijalle. Merkintä // (double slash) aloittaa kommenttirivin. Kääntäjä toimii siten että se jättää huomiotta merkit // ja sitä seuraavat merkit kyseisellä rivillä.

Tässä määritellään (esitellään) kaksi kokonaislukumuuttujaa joille on annettu nimet **luku_nappaimistolta** ja **yhta_suurempi_luku**. Näihin muuttujiin viitataan näillä nimillä myöhemmin ohjelmassa.

Kuten luvussa 2 mainittiin, nämä ohjelmarivit tarvitaan niissä ohjelmissa jotka lukevat jotain tietoa näppäimistöltä.

```
//  Peli.java  (c) Kari Laitinen

import java.util.* ;

class Peli
{
   public static void main( String[] ei_kaytossa )
   {
      Scanner nappaimisto = new Scanner( System.in ) ;

      int  luku_nappaimistolta ;
      int  yhta_suurempi_luku ;

      System.out.print(
          "\n Tama ohjelma on tietokonepeli. Anna kokonaisluku "
        + "\n lukualueelta  1 ... 2147483646 : " ) ;

      luku_nappaimistolta  =  nappaimisto.nextInt() ;

      yhta_suurempi_luku   =  luku_nappaimistolta  +  1 ;

      System.out.print( "\n Annoit luvun  "  +  luku_nappaimistolta  +  "."
                      +  "\n Minun luku on "  +  yhta_suurempi_luku   +  "."
                      +  "\n Valitan. Havisit. Peli on lopussa.\n") ;
   }
}
```

Tämä ohjelmarivi lukee näppäimistöltä kokonaisluvun ja tallettaa luetun luvun muuttujaan **luku_nappaimistolta**. Ohjelman suoritus pysyy tällä rivillä niin kauan kunnes käyttäjä on antanut kokonaisluvun. Kokonaisluvun lukee näppäimistöltä metodi **nextInt()** jota kutsutaan aiemmin määritellyn "näppäimistöolion" suhteen. Kutsuminen tapahtuu siten että näppäimistöolioon viittaavaan nimeen **nappaimisto** liitetään metodinimi **nextInt()** pisteoperaattorin **.** avulla.

Tuplaheittomerkkien " " sisällä olevat tekstit ovat näytölle tulostettavia merkkijonoja. Tekstiin kuuluva merkintä \n tarkoittaa että teksti alkaa uudelta riviltä. \n on nimeltään newline-merkki.

Peli.java - 1.+ Ohjelma joka toteuttaa yksinkertaisen tietokonepelin.

Kirjoittamalla **System.out** voidaan viitata tietokoneen näyttöön. **print()** on puolestaan metodi jonka avulla voidaan tulostaa tekstiä näytölle kun sitä kutsutaan **System.out.print(...)**. Tässä tulostettava teksti koostuu kahdesta merkkijonosta jotka on liitetty toisiinsa operaattorilla + (string concatenation operator). Tulostettava teksti annetaan sulkujen sisässä **print()**-metodille.

Kun tämä käskylause on suoritettu, muuttujan **yhta_suurempi_luku** sisältämä arvo on yhtä suurempi kuin muuttujaan **luku_nappaimistolta** talletettu arvo.

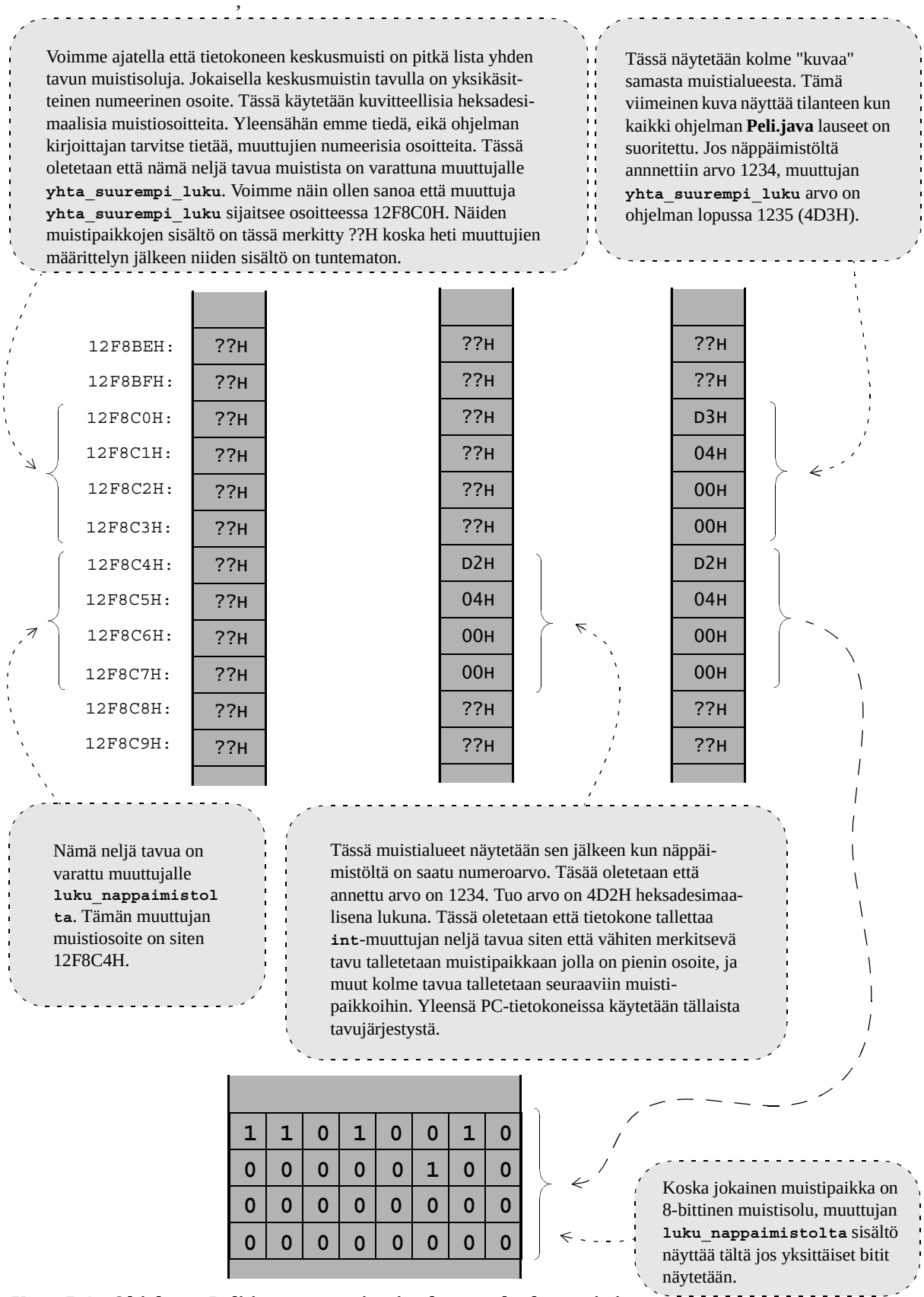
```
System.out.print(
    "\n Tama ohjelma on tietokonepeli. Anna kokonaisluku "
  + "\n lukualueelta  1 ... 2147483646 : " ) ;

luku_nappaimistolta  =  nappaimisto.nextInt( ) ;

yhta_suurempi_luku   =  luku_nappaimistolta  +  1 ;

System.out.print( "\n Annoit luvun  "  +  luku_nappaimistolta  +  "."
                +  "\n Minun luku on "  +  yhta_suurempi_luku   +  "."
                +  "\n Valitan. Havisit. Peli on lopussa.\n") ;
```

Yhdellä **System.out.print()**-metodikutsulla on mahdollista tulostaa monentyyppistä tietoa. Tässä kokonaislukumuuttujien arvot tulostetaan tuplaheittomerkkien sisällä annettujen merkkijonojen väliin. Operaattori + on aina sijoitettuna erityyppisten tietoelementtien väliin. Operaattori + muuntaa muuttujiin talletetut numeroarvot merkkijonoiksi, ja liittää näin syntyvät merkkijonot tuplaheittomerkkien sisällä annettuihin merkkijonoihin. Puolipiste (;) päättää tämänkin käskylauseen.

Peli.java - 1 - 1. Ohjelman toimintolauseet.

```
D:\javaohjelmat2>java Peli

 Tama ohjelma on tietokonepeli. Anna kokonaisluku
 lukualueelta  1 ... 2147483646 : 1234

 Annoit luvun  1234.
 Minun luku on 1235.
 Valitan. Havisit. Peli on lopussa.

D:\javaohjelmat2>java Peli

 Tama ohjelma on tietokonepeli. Anna kokonaisluku
 lukualueelta  1 ... 2147483646 : 2147483647

 Annoit luvun  2147483647.
 Minun luku on -2147483648.
 Valitan. Havisit. Peli on lopussa.
```

Peli.java - X. Ohjelman jälkimmäisessä suorituksessa sille annetaan liian suuri lukuarvo.



Kuva 5-1. Ohjelman Peli.java muuttujat tietokoneen keskusmuistissa.

Tyypin **int** muuttujien lisäksi Javassa on myös muuntyyppisiä kokonaislukumuuttujia. Näitä muuttujatyyppejä kutsutaan englanniksi termeillä "integer types" tai "integral types". Tyyppi **long** on eräs kokonaislukutyyppi. Tyypin **long** muuttujat käyttävät 8 tavua (64 bittiä) muistia. Ne ovat siten "pidempiä" kokonaislukumuuttujia kuin **int**-tyyppiset muuttujat. Jos ohjelman **Peli.java** muuttujat esiteltäisiin seuraavilla lauseilla

```
long  luku_nappaimistolta ;
long  yhta_suurempi_luku ;
```

ohjelma toimisi oikein huomattavasti suuremmilla numeroarvoilla kuin se toimii nykyisellään.

Kolmas kokonaislukutyyppi on **short** joka on kaksitavuinen, 16-bittinen, muuttuja. Tämä muuuttujatyyppi voi olla hyödyllinen jos ohjelma käsittelee ainoastaan kokonaislukuja jotka ovat pienempiä kuin 32767. Jos on tarve tallettaa useita tällaisia pienehköjä arvoja, voidaan säästää muistia kun käytetään **short**-tyyppiä **int**-tyypin sijaan. Pienissä ohjelmissa ei kuitenkaan ole varsinaista syytä käyttää **short**-tyypin muuttujia koska nykyisissä tietokoneissa on yllin kyllin muistia tarjolla.

Tyyppiä **byte** olevia muuttujia voidaan käyttää tallettamaan 8-bittisiä arvoja alueelta -128 ... 127. Tyypin **byte** muuttuja siis käyttää yhden tavun verran muistia.

Tyyppiä **char** muuttujat ovat kaksitavuisia 16-bittisiä muuttujia, jotka voivat tallettaa jonkin merkin Unicode-merkkikoodin. Ensimmäiset 256 Unicode-koodausjärjestelmän koodia ovat samat kuin vanhemmassa ASCII-koodausjärjestelmässä. **char**-tyypin muuttujat ovat periaatteessa kokonaislukumuuttujia koska merkkien koodit ovat kokonaislukuarvoja. On kuitenkin parempi olla käyttämättä **char**-muuttujia tilanteissa joissa käsitellään jotain muuta kokonaislukutietoa kuin merkkien koodeja.

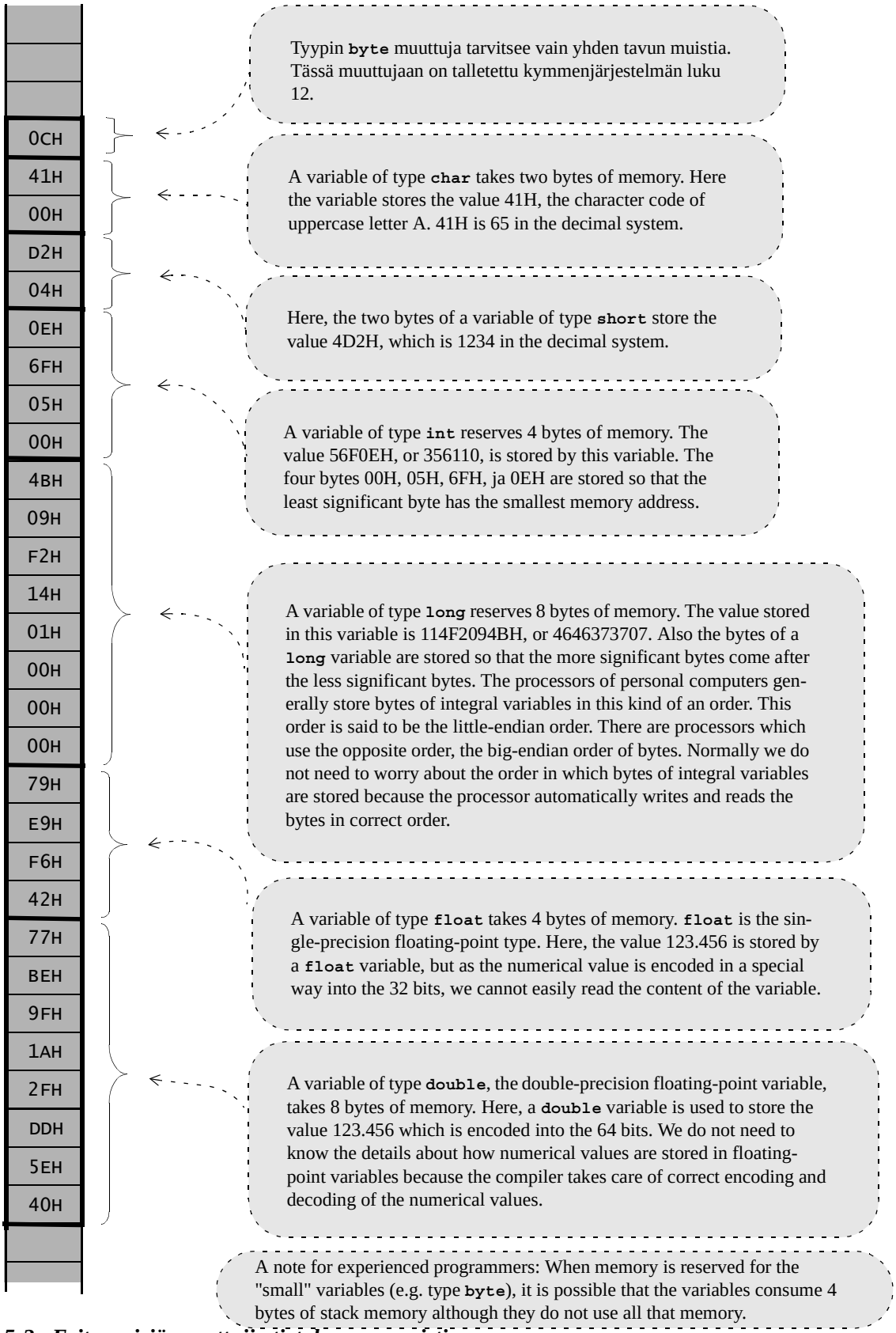
Kokonaislukutyyppejä kuten **int**, **short** ja **long** käytetään kun halutaan käsitellä kokonaislukuja tietokoneohjelmissa. Yleisimmin käytetty kokonaislukutyyppi on **int**. Kokonaislukumuuttujat ovat tarpeellisia kun haluamme laskea joidenkin asioiden lukumääriä ohjelmissa. Kokonaislukumuuttujien ja myös muiden Javan muuttujien ominaisuudet on lyhyesti kerrottu taulukossa 5-1. Kuva 5-2 näyttää miltä muuttujat näyttävät tietokoneen keskusmuistissa. Muita muuttujatyyppejä käsitellään myöhemmin tässä luvussa. Tyyppi **boolean** on ns. booleaaninen muuttujatyyppi joka voi saada vain kaksi erilaista arvoa: **true** ja **false**. Näitä muuttujia tutkitaan tarkemmin kun ne tulevat vastaan esimerkkiohjelmissa.

Joihinkin muihin ohjelmointikieliin (esim. C++ ja C#) kuuluu ns. etumerkittömät (unsigned) kokonaislukumuuttujat. Etumerkittömillä muuttujatyypeillä tarkoitetaan kokonaislukumuuttujia jotka voivat saada vain ei-negatiivisia arvoja. Näiden muuttujatyyppien tallennuskapasiteetti "positiivisessa" suunnassa on laajempi, koska muuttujien arvoaluetta ei tarvitse käyttää negatiivisten lukujen esittämiseen. Javassa näitä etumerkittömiä kokonaislukumuuttujia ei kuitenkaan ole tarjolla, oletettavasti siitä syystä että tällä tapaa Java-kieli on saatu pysymään mahdollisimman yksinkertaisena ohjelmointikielenä.

Harjoituksia ohjelmalla Peli.java

Harjoitus 5-1. Muuta ohjelmaa siten että se aina häviää pelin käyttäjän kanssa. Tarkoitus on että peliohjelman luku on aina yhtä pienempi kuin käyttäjän luku. Vähennyslaskuoperaatio suoritetaan Javassa operaattorilla -.

Harjoitus 5-2. Tee ohjelmasta versio joka tulostaa kolme lukua jotka ovat juuri ja juuri suurempia kuin käyttäjän antama luku. Esimerkiksi, jos ohjelmalle annetaan luku 144, ohjelma tulostaa luvut 145, 146 ja 147.



Kuva 5-2. Erityyppisiä muuttujia tietokoneen muistissa.

Table 5-1. Java-ohjelmointikielen muuttujatyypit.

Tyyppi	Muistin määrä	Lukualue
int	4 tavua 32 bittiä	-2,147,483,648 ... 2,147,483,647 -80000000H ... 7FFFFFFFH
short	2 tavua 16 bittiä	-32,768 ... 32,767 -8000H ... 7FFFH
long	8 tavua 64 bittiä	-9,223,372,036,854,775,808 ... 9,223,372,036,854,775,807 -8000000000000000H ... 7FFFFFFFFFFFFFFFH
byte	1 tavu 8 bittiä	-128 ... 127 -80H ... 7FH
boolean	1 tavu 8 bittiä	false tai true
char	2 tavua 16 bittiä	Unicode 0 ... 65535
float	4 tavua 32 bittiä	Tarkkuus: 7 desimaalista numeroa Alue: +/- 1.5e-45 ... 3.4e38[a]
double	8 tavua 64 bittiä	Tarkkuus: 15 desimaalista numeroa Alue: +/- 5.0e-324 ... 1.7e308
muistiosoite[b]	4 tavua 32 bittiä	Riittää osoittamaan 4,294,967,296 tavua (4 gigatavua) keskusmuistia.

a. Tässä annetut lukualueet ovat likimääräisiä.

b. "muistiosoite" ei ole Javan muuttujatyyppi. Tämä taulukon rivi kertoo kuinka paljon muistia varataan (allokoidaan) silloin kun niin sanottu olioviittaaja esitellään. Olioviittaajat ovat eräänlaisia muuttujia joihin talletetaan heap-muistiin luotujen olioiden osoitteet.

5.2 Varatut sanat, nimet, välilyönnit ja newline-merkit

Java-kieliset tietokoneohjelmat koostuvat erityismerkeistä ja merkkipareista kuten {, }, (,), =, ;, ", \n, ja +; sekä luonnolliseen kieleen kuuluvista ilmauksista kuten **`static`**, **`void`**, **`main`**, **`int`**, **`System`**, ja **`luku_nappaimistolta`**. Sekä erikoismerkit että luonnolliseen kieleen kuuluvat ilmaukset ovat merkityksellisiä Java-kääntäjälle. Kun Java-kääntäjä aloittaa ohjelman kääntämisen, se lukee tekstitiedoston joka sisältää laaditun ohjelman merkkikoodeina. Riippuen siitä mitä kääntäjä lukee lähdekielisen ohjelman sisältävästä tiedostosta, se tuottaa vastaavan suoritettavan ohjelman, joka on käännöksen tulos. Ohjelmaan kirjoitettavilla erikoismerkeillä ja luonnollisen kielen ilmauksilla voimme kertoa Java-kääntäjälle millaisen suoritettavan ohjelman haluamme sen tuottavan.

Ohjelmointikielen varatut sanat (reserved words tai keywords) ovat luonnollisesta kielestä (siis käytännössä englannista) lainattuja sanoja ja lyhenteitä joilla on erityismerkitys kyseisellä kielellä laadituissa ohjelmissa. Kääntäjä osaa varattujen sanojen avulla tunnistaa ohjelman syntaktisia rakenteita. Varattuja sanoja ei voida käyttää ohjelmissa niminä. Java-kielessä on seuraavat varatut sanat:

```
abstract   boolean  break     byte      case         catch
char       class    const     continue  default      do
double     else     enum      extends   final        finally
float      for      goto      if        implements   import
instanceof int      interface long      native       new
null       package  private   protected public       return
short      static   super     switch    synchronized this
throw      throws   transient try       void         volatile
while
```

Varattujen sanojen merkitys esitellään tulevissa luvuissa sitä mukaa kun kohtaamme niitä esimerkkiohjelmissa. Kaikki varatut sanat on lueteltu ja selitetty lyhyesti liitteessä B.

Tietokoneohjelmassa oleva nimi (name) on ohjelman kirjoittajan keksimä tai valitsema ilmaus. Nimiä kutsutaan myös symboleisksi. Myös englanninkielinen sana identifier tarkoittaa nimeä. Ohjelmassa **Peli.java** on käytössä muuttujien nimet **`luku_nappaimistolta`** ja **`yhta_suurempi_luku`**. Ohjelman päämetodin nimi on **`main()`**. **`print()`** on nimenä metodilla jolla voimme kirjoittaa tekstiä näytölle, ja **`nextInt()`**-metodilla voimme lukea kokonaisluvun näppäimistöltä. Vaikka **`main`**, **`print`**, ja **`nextInt`** eivät ole virallisia Javan varattuja sanoja, niitä voisi lähes pitää sellaisina koska Java-ohjelman päämetodin nimi tulee olla **`main()`**, ja **`print()`** ja **`nextInt()`** ovat Javan standardiluokissa määriteltyjä nimiä. (Metodinimien loppuun on usein tapana kirjoittaa tyhjät kaarisulkeet (), vaikka nuo kaarisulkeet eivät varsinaisesti kuulu nimeen. Niillä pyritään tämän kirjan tekstissä erottamaan metodien nimet muista nimistä kuten esimerkiksi muuttujien nimistä.)

Tämän kirjan esimerkkiohjelmissa käytetään ns. luonnollisia nimiä. Luonnolliset nimet koostuvat luonnollisen kielen sanoista. Muuttujan nimen pitäisi kuvata kyseisen muuttujan käyttötarkoitusta, ja huolellisesti valituilla sanoilla kirjoitettu luonnollinen nimi yleensä toimii tällä tavalla. Luonnollisten nimien käytöllä tämän kirjan esimerkkiohjelmat on pyritty tekemään mahdollisimman helppolukuisiksi. Koska tietokoneohjelmat ovat luonnostaan monimutkaisia, niitä ei kannata tehdä entistä vaikeammiksi huonosti ymmärrettäviä nimiä käyttämällä.

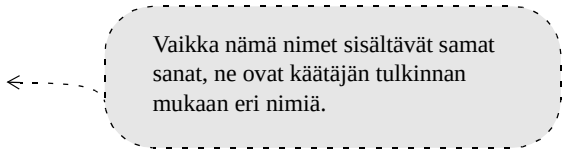
Luonnollisen nimen erilliset sanat voidaan yhdistää alaviivamerkkiä (_) tai ns. kapitalisoinnilla voidaan merkitä uuden sanan alkaminen nimessä. Tässä kirjassa muuttujien nimet on kirjoitettu siten että nimen sanat on yhdistetty alaviivoilla. Ns. standardit Java-nimet kuten **`nextInt`** on puolestaan kirjoitettu "kapitalisoidusti" siten että uusi nimen sana alkaa isolla kirjaimella (capital letter). Näin standardit Java-nimet on helppo erottaa tämän kirjan kirjoittajan keksimistä nimistä.

Luonnolliset nimet ovat tärkeitä ohjelman lukijalle, mutta kääntäjä käsittelee nimiä hyvin teknisellä tavalla. Kääntäjän näkökulmasta nimien ei tarvitse olla luonnollisia. Seuraavat ovat säännöt joiden mukaan nimet ovat hyväksyttäviä Java-ohjelmissa:

- Nimen tulee olla katkeamaton sarja merkkejä. Nimeen ei saa sisällyttää välilyöntimerkkejä, rivinvaihtomerkkejä ja muita erikoismerkkejä. (Tämän kirjan tekstissä nimi voi joskus jatkua seuraavalla rivillä, ts. nimi sisältää väliviivan. Tällainen nimen jatkaminen seuraavalla rivilla ei ole kuitenkaan mahdollista ohjelmatekstissä.)
- Nimen ensimmäinen merkki tulee olla kirjain tai alaviivamerkki (_).
- Ensimmäistä merkkiä seuraavat merkit tulee olla kirjaimia, numeroita tai alaviivamerkkejä. Tämä tarkoittaa käytännössä että numero ei voi aloittaa nimeä, mutta numeroita voidaan muuten käyttää nimissä.

Näiden sääntöjen mukaan kaikki seuraavat muuttujamäärittelyt olisivat hyväksyttäviä Java-kääntäjälle:

```
int  i ;
int  __xxx ;
int  joku_luku ;
int  JOKU_LUKU ;
int  jokuLuku ;
int  x2 ;
```



Java-kääntäjä hyväksyy lyhyet ja lyhennetyt nimet, mutta on parempi käyttää pidempiä luonnollisia nimiä, koska niillä voidaan lisätä ohjelmien luettavuutta.

Kun kääntäjä löytää ohjelmasta jotain ohjelmointikielen sääntöjen vastaista, se tulostaa virheilmoituksia näytölle. Myös nimiä kirjoitettaessa on mahdollista tehdä virheitä. Esimerkiksi muuttujamäärittely

```
int  1_suurempi_luku ;
```

on virheellinen koska numero ei voi aloittaa nimeä. Määrittelyt

```
int  joku luku nappaimistolta ;
int  joku-toinen-luku ;
```

eivät ole hyväksyttäviä koska nimiin ei voi sisällyttää merkkejä ’ ’ ja ’-’.

Välilyöntimerkkien käyttö on tärkeää Java-ohjelmissa. Varatun sanan ja nimen välissä täytyy olla ainakin yksi välilyöntimerkki. Esimerkiksi muuttujamäärittely

```
intluku_nappaimistolta ;
```

johtaisi käännösvirheeseen koska kääntäjä ei osaisi erottaa varattua sanaa **`int`** muuttujan nimestä. Yleensä ottaen on niin että erikoismerkkien ja nimien välissä ei tarvita välilyöntejä, mutta yleensä on parempi pistää myös noihin paikkoihin välilyöntejä jotta ohjelmakoodista saadaan luettavampaa. Seuraavat kaksi lausetta

```
yhta_suurempi_luku=luku_nappaimistolta+1;
yhta_suurempi_luku  =  luku_nappaimistolta  +  1 ;
```

ovat samoja lauseita teknisessä mielessä mutta jälkimmäinen lause on helpompi lukea.

Halutessasi voit kirjoittaa enemmän kuin yhden välilyöntimerkin niihin ohjelman kohtiin johon välilyönti ylipäätään voidaan kirjoittaa. Niissä ohjelman kohdissa joissa välilyöntimerkki on sallittu on myös mahdollista käyttää rivinvaihtomerkkiä tai tabulointimerkkiä. Rivinvaihtomerkki on jokaisen Java-ohjelman fyysisen rivin lopussa. Kun ohjelmaan lisätään tyhjiä rivejä, käytännössä ohjelman fyysiseen tiedostoon lisätään rivinvaihtomerkkejä.

Usein ohjelman lauseet ovat niin pitkiä että ne on parasta kirjoittaa usealle fyysiselle ohjelmariville. Yleinen sääntö lauseen jatkamiseen on että rivinvaihtomerkin saa kirjoittaa kaikkiin niihin kohtiin joissa välilyöntimerkki on sallittu. Tämä tarkoittaa käytännössä että nimet täytyy säilyä ehjinä kun lause jatkuu seuraavalla rivillä.

5.3 Liukulukumuuttujat (floating-point variables)

Kokonaislukumuuttujat ovat käteviä kun haluamme esimerkiksi laskea lukumääriä. Toisaalta kun haluamme suorittaa tähän todelliseen maailmaan liittyviä laskutoimituksia, kokonaisluvut eivät useikan ole riittäviä. Kun halutaan tallettaa sellaisia arvoja kuin 3.1416 (piin arvo) tai 6,378.16 (maan säde kilometreinä päiväntasaajalla), kannattaa hyödyntää ns. liukulukumuuttujia. Näitä muuttujia voidaan esitellä varattujen sanojen **`float`** ja **`double`** avulla.

Ohjelma **Mailit.java** on esimerkki jossa käytetään liukulukumuuttujaa jonka tyyppi on **`float`**. Ohjelma pyytää käyttäjältä etäisyyden maileina, ja muuttaa sitten annetun etäisyyden kilometreiksi. Koska yksi maili on 1.6093 kilometriä, ohjelman täytyy käyttää liukulukumuuttujaa tarkean laskentatuoksen aikaansaamiseksi. Kokonaislukumuuttujaan ei voi tallettaa arvoa johon kuuluu desimaalipiste.

Liukulukumuuttujat perustuvat ideaan että jokainen luku voidaan esittää ns. exponentiaalisella notaatiolla. Tätä notaatiota käyttävät usein mm. tiedeihmiset jotka suorittavat laskentaa hyvin suurilla tai hyvin pienillä luvuilla. Seuraavat ovat esimerkkilukuja esitettynä eksponentiaalisella notaatiolla (* tarkoittaa kertolaskua, piste on desimaalipiste ja pilkku toimii luvun numeroiden ryhmittelijänä)

"NORMAALI" NOTAATIO	EXPONENTIAALISIA NOTAATIOITA	
2,000,000,000,000	$2.0 * 10^{12}$	2.0e12
299,800,000	$2.998 * 10^{8}$	2.998e8
0.000,000,123	$1.23 * 10^{-7}$	1.23e-7
0.000,000,000,002	$2.0 * 10^{-12}$	2.0e-12

Eksponentiaalisessa notaatiossa luku esitetään kahdessa osassa jotka ovat mantissa ja eksponentti. Ensimmäisessä yllä annetussa luvussa mantissa on 2.0 ja eksponentti on 12. Mantissa kertoo "paljonko" luku tarkoittaa ja eksponentti ilmoittaa monestiko mantissa on kerrottava kymmenellä, tai jaettava kymmenellä negatiivisten eksponenttien tapauksessa, jotta saadaan luku "normaalissa" notaatiossa. Eksponentiaalinen notaatio perustuu siis tosiasiaan että jokainen luku voidaan esittää muodossa "mantissa kertaa 10 potenssiin eksponentti". Tässä on muistettava että kymmenen potenssiin nolla on 1.

Liukulukumuuttujat tallettavat lukuja eksponentiaalisessa muodossa. Mantissa ja eksponentti talletataan erikseen binaarisina lukuina. Tyyppiä **`float`** oleva liukulukumuuttuja käyttää 4 tavua muistia ja tyyppiä **`double`** oleva muuttuja käyttä 8 tavua muistia.

Liukulukumuuttujien sisäinen rakenne on melmoisen monimutkainen. Esimerkiksi **`float`**-tyyppisen muuttujan varaamat 32 bittiä käytetään siten että 8 bittiä käytetään eksponentin talletukseen, 23 bittiä käytetään mantissan talletukseen ja jäljellejäävä 1 bitti käytetään luvun etumerkin (plus tai miinus) talletukseen. Tarkempi liukulukutyyppi **`double`** käyttää enemmän bittejä mantissan ja eksponentin talletukseen.

Yleensä ohjelmien kirjoittajien ei tarvitse tuntea liukulukumuuttujien sisäistä rakennetta. Ohjelmointikielen kääntäjä ja itse tietokone pitävät huolta siitä että laskennat liukuluvuilla suoritetaan oikein. Kun esittelemme liukulukumuuttujan

```
float  joku_liukuluku ;
```

kääntäjä huolehtii siitä että 4 tavua muistia varataan tälle muuttujalle. Kun annamme muuttujalle arvon sijoituslauseella

```
joku_liukuluku  =  12.34F ;
```

kääntäjä pitää huolen että literaalivakio 12.34F muunnetaan automaatisesti siihen eksponentiaaliseen esitysmuotoon jota käytetään **`float`**-tyypin muuttujissa, ja arvo tallettuu oikein muuttujalle varattuun neljään tavuun. Huomaa että kirjain F tulee olla lisättynä **`float`**-tyyppisen literaalivakion perään.