Ohjelmoinnin perusteet
R0027

5 op

Reviewing concepts from last class

- Understanding code with:
  - Variables
  - Data types: String and int
  - System.out.print()
  - Concatenation

C: Tulostamista, lukemista ja operaattoreita

Viopen luku 5

Luokka ja Metodi lyhyesti

- Luokka
  - Olemme kirjoittaneet jokaisen ohjelmamme johonkin luokkaan
  - Luokkaan kirjoitetaan metodeita
  - Esimerkkejä luokista
    - HenkilotietoTulostaja
    - System
    - String
- Metodi
  - Olemme kirjoittaneet jokaisen ohjelmamme main-nimiseen päämetodiin
  - Olemme käyttäneet muitakin metodeita apunamme
  - Metodi sisältää toiminnallisuuden, joka voidaan suorittaa ”kutsumalla metodia”
  - Esimerkkejä metodeista
    - main()
    - print()
    - println()

Vilkaisu Javan peruskirjastoon

- Java tarjoaa ohjelmoijalle valmiin kirjaston (Java API), joka sisältää valmiita luokkia ja metodeita
- Ohjelmoijan ei tarvitse siis kirjoittaa kaikkea itse
- Java API on kuvattu osoitteessa https://docs.oracle.com/javase/8/docs/api/
- Kaikki **java.lang**-paketista löytyvät luokat ovat automaattisesti käytettävissä (kuten String)
- Muiden pakkausten osalta java-tiedoston alkuun on määriteltävä tuontilauseet

```
import java.util.Scanner;
import java.awt.*;

public class Ohjelmani {
   //…
```

Tulostaminen komentoriville

- System.out.print();
- System.out.println();

- Muista erikoismerkkejä:
  - \n
  - \t
  - \'
  - \"
  - \\

Printing to the command line – Java API

System.out is the so called "**Standard output stream**". Text written to the standard output stream appears on the command line or terminal window. To write text to the standard output stream you need to use methods from the class PrintStream. This works because the standard output stream derive from the class PrintStream. The System class belongs to the **java.lang** package. This class is imported by default in every java program.

`System.out.print(String s);` Prints the string passed as argument

`.println(String s);` Prints the string passed as argument and then a newline character '\n'

`.printf(String format, Object... args);` Prints a formatted string to the output stream using the specified format string and arguments

Methods of class PrintStream
This class resides in the **java.io** package

These two lines of code produce the same output!

```
System.out.print("My line of output\n");
System.out.println("My line of output");
```

Browse the java API to verify all this! → https://docs.oracle.com/javase/8/docs/api/

Formatoitu tulostus

Formatoitu tulostus mahdollistaa C-kielestä tuttujen **printf()**-metodien käytön Javassa. **java.io.PrintStream**-luokassa määritelty metodi on muotoa

```
public PrintStream printf(String format, Object… args)
```

Metodia käytettäessä tulostuslause voi näyttää esimerkiksi seuraavalta:

```
System.out.printf("Nimesi on %s ja ikäsi %d%n",nimi,ika);
```

Ensimmäisessä **format**-parametrissa määritellään tulostettava merkkijono ja merkkijonon sisältämien tietokenttien tulostusmuodot **konversiomerkkien** avulla (edellä **%s** ja **%d** ja **%n**). Sen jälkeen tulee **argumenttilista**, joka sisältää parametrit, jotka tulostetaan konversiomerkkien osoittamiin tietokenttiin. Esimerkiksi edellä muotoilumerkkijonossa kerrotaan, että ensin tulostetaan teksti *Nimesi on*, sen jälkeen **nimi**-parametrin arvo **merkkijonona** (**%s**), jota seuraa teksti *ja ikäsi*. Lopuksi tulostetaan **ika**-muuttujan arvo **10-järjestelmän kokonaislukuna** (**%d**). Lopuksi tulostetaan rivinvaihtomerkki (**%n**).

Muotoilumerkkijonon (**format**-parametri) **konversiomerkit** ja niiden eteen kirjoitetut **ohjausmerkit** kirjoitetaan yleisesti muotoon

```
%[indeksi$][liput][leveys][.tarkkuus]konversio
```

Merkkijono alkaa aina **%**-merkillä. Valinnainen indeksi on kokonaisluku, joka ilmaisee, monesko **args**-listassa oleva argumentti tulostetaan tässä kohdassa. Ensimmäiseen argumenttiin viitataan **1$**, toiseen **2$** jne. Yleensä tätä ei tarvita, vaan argumentit tulostetaan siinä järjestyksessä kuin ne ovat **args**-listassa.

Source: Java 2 - Ohjelmoinnin peruskirja, page 630, author: Pekka Kosonen

Formatoitu tulostus, jatko…

Valinnaisesti voidaan myös esittää joukko **lippuja** (*flags*), joilla ohjataan tulostusta. Tavallisin lippu on miinus-merkki (-), joka ilmaisee, että tulostetaan kentän vasempaan reunaan, kun oletuksena on aina oikean reunan tasaus.

Leveys ilmaisee tulostuskentän leveyden ja **tarkkuus** desimaaliluvuille desimaalien lukumäärän.

Ainoa pakollinen **konversio**-merkki ilmaisee, missä muodossa muuttujan arvo tulostetaan.

Tavallisimmat **konversiomerkit** ovat:

- **d** – kokonaisluvun tulostaminen 10-järjestelmän lukuna
- **f** – liukuluvun tulostaminen desimaalilukuna
- **e** – liukuluvun tulostaminen eksponenttimuodossa
- **n** – rivinvaihtomerkin tulostaminen
- **x** – kokonaisluvun tulostaminen heksadesimaalilukuna
- **s** – merkkijonon tulostaminen
- **c** – Unicode-merkin tulostaminen
- **%** – %-merkin tulostaminen.

Tavallisimmat liput ovat:

- **-** – tulostetaan kentän vasempaan reunaan (oletus on oikea reuna)
- **+** – numeerisissa arvoissa on aina etumerkki
- **0** – numeerisen kentän alkuun tulee etunollia täytteeksi.

Source: Java 2 - Ohjelmoinnin peruskirja, page 631, author: Pekka Kosonen

String manipulation methods – indexOf, lastIndexOf

Indexes start always at 0 (zero)!!

Indexes in a String	0	1	2	3	4	5	6	7	8	9	10	11	12
String mytext	**J**	**a**	**v**	**a**		**i**	**s**		**g**	**r**	**e**	**a**	**t**

Browse the java API to verify the specs of methods indexOf and lastIndexOf
https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

```
myText.indexOf(´J´) → 0

myText.indexOf(´t´) → 12

myText.indexOf(´a´) → 1
myText.indexOf(´a´, 2) → 3

myText.lastIndexOf(´a´) → 11

myText.indexOf(“is“) → 5
```

Copy the java program ExampleStringSearch.java from Optima, run it on Eclipse and check the results!

Make sure you understand the code.

Näppäimistösyötteiden lukeminen

- Näppäimistösyötteitä voidaan lukea Scanner-luokan metodeilla
- Java-tiedoston alkuun on kirjoitettava seuraava lause, koska Scanner-luokka ei ole java.lang-pakkauksessa:

```
import java.util.Scanner;
```

- Päämetodin alkuun pitää kirjoittaa Scanner-lukijan objektin alustus, jotta sitä voidaan käyttää

```
Scanner lukija = new Scanner(System.in);
```

- Tämän jälkeen voidaan kutsua metodeita, jotka lukevat näppäimistösyötteitä

```
luku1 = lukija.nextInt();
```

Scanner – code explanation

```
import java.util.Scanner;
```

Importing the Scanner class to our program. This class belongs to the java.util package

"lukija" is just a name chosen by the programmer. It could be anything else, like "keyboardReader"

```
Scanner lukija = new Scanner(System.in);
```

Same as …

```
Scanner lukija;
lukija = new Scanner(System.in);
```

Creates a new instance ("object") of the Scanner class to be used in our program. A Scanner object as its name says acts as "scanner" that can parse data types and strings from an input stream

System.in is the so called "**Standard input stream**". Typically this stream corresponds to keyboard input or another input source specified by the host environment

```
lukija.close();
```

The close() method terminates or "closes" the scanner object used in the program

Näppäimistösyötteiden lukeminen

```
import java.util.Scanner;

public class NimenKysyminenJaTulostus {
        public static void main(String[] args) {

                String kayttajanNimi;
                Scanner lukija = new Scanner(System.in);

                System.out.print("Anna nimesi: ");
                kayttajanNimi = lukija.nextLine();

                System.out.print("Hei vaan “ +kayttajanNimi);

        }
}
```

Näppäimistösyötteiden lukeminen

Metodi	Palautus-tyyppi	Selitys
`nextBoolean()`	boolean	Skannaa syötteestä boolean-tyyppisen arvon ja palauttaa sen
`nextByte()`	byte	Skannaa syötteestä byte-tyyppisen arvon ja palauttaa sen
`nextDouble()`	double	Skannaa syötteestä double-tyyppisen arvon ja palauttaa sen
`nextFloat()`	float	Skannaa syötteestä float-tyyppisen arvon ja palauttaa sen
`nextInt()`	int	Skannaa syötteestä int-tyyppisen arvon ja palauttaa sen
`nextLong()`	long	Skannaa syötteestä long-tyyppisen arvon ja palauttaa sen
`nextShort()`	short	Skannaa syötteestä short-tyyppisen arvon ja palauttaa sen
`nextLine()`	String	Palauttaa syötteenä annetun rivin
`next()`	String	Palauttaa ensimmäisen merkkijonon ennen erotinmerkkiä. Erotinmerkki on oletuksena välilyönti.

Näppäimistösyötteiden lukeminen

```
import java.util.Scanner;

public class LukujenSumma {
	public static void main(String[] args) {
		int luku1 = 0, luku2 = 0;

		Scanner lukija = new Scanner(System.in);

		System.out.print("Anna ensimmäinen luku: ");
		luku1 = lukija.nextInt();

		System.out.print("Anna toinen luku: ");
		luku2 = lukija.nextInt();

		System.out.print(luku1 + " + " + luku2
			+ " = " + (luku1 + luku2));
	}
}
```

Operaattoreita

- Sijoitusoperaattori

```
x = 4;
```

- Aritmeettiset operaattorit

```
4+2
```

- Vertailuoperaattorit

```
luku == 3
```

- Loogiset operaattorit

```
rahaa && nakkikioskiAuki
```

Javan aritmeettiset operaattorit

Operaattori	Selitys
+ +	Unaarinen kasvatusoperaattori
- -	Unaarinen vähennysoperaattori
+	Yhteenlasku
-	Vähennyslasku
*	Kertolasku
/	Jakolasku
%	Jakojäännös (modulo)
+ =	Lisäysoperaatio
- =	Vähennysoperaatio
* =	Kertolaskuoperaatio
/ =	Jakolaskuoperaatio
% =	Jakojäännösoperaatio

Vertailuoperaattorit

Operaattori	Selitys	Esimerkki käytöstä
==	Yhtä suuri	`123 == 456` ***(ei toimi Stringillä!)*** *`nimi1.equals(nimi2)`*
!=	Eri suuri	`123 != 456` *`!nimi1.equals(nimi2)`*
<	Pienempi kuin	`12 < 13`
>	Suurempi kuin	`12 > 13`
<=	Pienempi tai yhtä suuri kuin	`12 <= 13`
>=	Suurempi tai yhtä suuri kuin	`12 >= 12`

Vertailuoperaattorit - esimerkki

```
boolean yhtaSuuria;
int luku1 = 3;
int luku2 = 4;

yhtaSuuria = (luku1 == luku2); //false eli epätosi
System.out.println(”Yhtäsuuria: ” +yhtaSuuria);
//tulostaa ”Yhtäsuuria: false”

boolean onSuurempi;
boolean onPienempi;

onSuurempi = luku1 > luku2; //false
onPienempi = luku1 < luku2; //true eli tosi
```

Loogiset operaattorit

Operaattori	Selitys	Esimerkki käytöstä
!	Ei (NOT)	`!(1 <= 1)`
&&	JA (AND)	`(1 == 1) && (2 <= 3)`
||	TAI (OR)	`(0 > 1) || (0 < 1)`

Loogiset operaattorit - esimerkki

```
boolean nakkikioskiAuki = true;
boolean rahaaTaskussa = false;

boolean menenNakkiKioskille = nakkikioskiAuki && rahaaTaskussa; // false
```

```
boolean jaakaapissaRuokaa = true;
boolean pakastimessaRuokaa = false;
boolean jauhokaapissaRuokaa = false;

boolean syotavaaKeittiossa = jaakaapissaRuokaa || pakastimessaRuokaa ||
    jauhokaapissaRuokaa; // true
```

```
boolean ravintolaTaynna = true;
boolean ravintolaanMahtuu = !ravintolaTaynna; // false

boolean eriNumeroita = (3 != 4); // true
```

Totuusarvotaulukko

A	B	A && B	A || B	! A
TRUE	**TRUE**	TRUE	TRUE	FALSE
TRUE	**FALSE**	FALSE	TRUE	FALSE
FALSE	**TRUE**	FALSE	TRUE	TRUE
FALSE	**FALSE**	FALSE	FALSE	TRUE

Ehdollinen operaattori

- ***lauseke1 ? lauseke2 : lauseke3***
  - *Ensimmäisen osan (lauseke1)* arvo tulee olla tyyppiä boolean
  - Mikäli *lauseke1* on true, saa koko ehdollinen lauseke arvokseen *lauseke2*
  - *Mikäli lauseke1 on false, saa koko ehdollinen lauseke arvokseen lauseke3*

Esimerkki:

```
System.out.prinln("Are you in Finland? (true/false): ");
Scanner lukija = new Scanner(System.in);
boolean ollaankoSuomessa = lukija.nextBoolean();
String tervehdys = ollaankoSuomessa ? "Moi" : "Hello";
System.out.println(tervehdys);
```

Suoritusjärjestys

Operaattorien suoritusjärjestys (ylhäältä alas)
++ -- !
* / %
+ -
< > <= >=
== !=
&&
||
?:
= op=

- Sulkujen käytöllä voi selkeyttää silti luettavuutta
- Alla olevat rivit tekevät saman asian, mutta luettavuudessa on eroa

```
1 < 2 && 3 < 2;
(1 < 2) && (3 < 2);
```

Rakenteiset lauseet

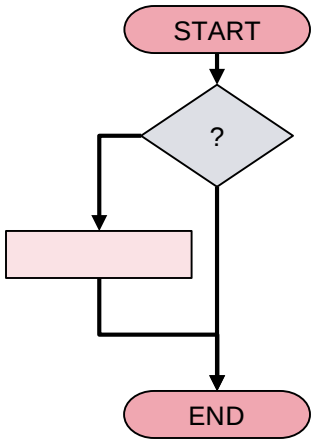
- Rakenteisilla lauseilla voidaan vaikuttaa lähdekoodirivien suoritusjärjestykseen
- Rakenteiset lauseet määrittävät metodin sisälle uusia lohkoja, jotka alkavat ja loppuvat aaltosulkuihin **{ }**
- **HUOM! Lohkon sisällä määritelty muuttuja ei ole näkyvissä lohkon ulkopuolelle**

Valintalause - if

- Jos tilillä on rahaa, tilaa pitsa.

```
boolean tilillaOnRahaa = true;

if (tilillaOnRahaa) {
   System.out.println("Tilaa pitsa.");
}
```

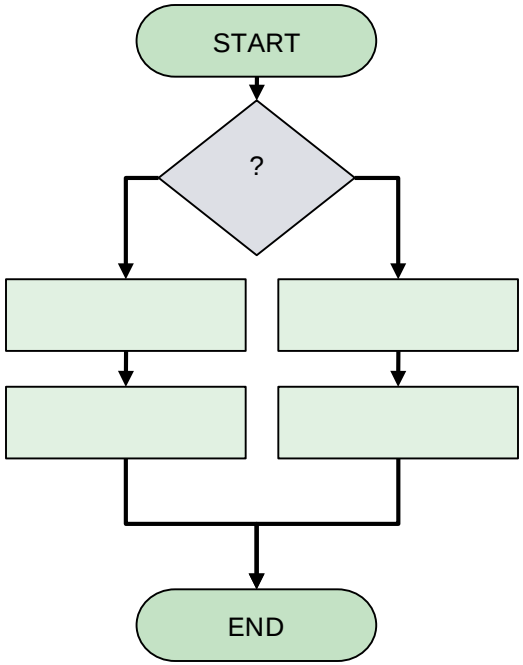



```
if (luku1 > luku2) {
   System.out.println("Luku1 on suurempi kuin luku2");
}
```

Valintalause - else

- Jos tilillä on rahaa, tilaa pitsa. Muuten mene lenkille.

```
if (tilinSaldo > 0) {
   System.out.println("Tilaa pitsa.");
   System.out.println("Katso leffa.");
}
else {
   System.out.println("Mene lenkille");
   System.out.println("Käy suihkussa");
}
```



Check the example code

Copy the java program EmailChecking.java from Optima, run it on Eclipse and check the results!

The code contain:

- If/else blocks (valintalauset)
- Comparison operators (vertailuoperaattorit)
- Logical/Boolean operators (loogiset operaattorit)

Make sure you understand the code.