Enkel oversikt over terminologi og kodesnutter i INF1000 uke 1 og 2

Gard Inge Rosvold

September 12, 2016

Contents

1 Sjekkliste obligatorisk innlevering

- Sjekk at programmet kompilerer.
- Kjør programmet ditt med forskjellige typer input og se at det fungerer som du vil.
- Svar på oppgaven, bruk det du blir bedt om!
- Ikke bruk Æ, Ø eller Å. Heller ikke i kommentarer.
- MinOppgave skal alltid ha en kommentar øverst som forteller hva programmet gjør.
- Bruk forklarende variabelnavn. Skal du lagre alder kall den "int alder" og ikke "int a".
- Hold deg til pensumet som er gjennomgått, hvis du kan mer avanserte ting i java så bruk det i MinOppgave, men pass på at du fortsatt svarer på oppgaven.
- Legg ved README.txt, svar på spørsmålene dere blir bedt om. Dersom programmet ditt IKKE kompilerer eller kjører sånn du hadde tenkt - skriv at du vet om det, men at du ikke har klart å finne ut hvorfor.
- Sitter du sammen med andre og jobber på obligen eller får hjelp så skriv det i README.txt, men ikke skriv av andre eller kopier andres kode (dette er fusk). Feks. kan du skrive "jeg fikk hjelp av Arne, *brukernavn*, til å løse oppgave 3.4" eller "jeg og Kari, *brukernavn*, satt sammen og løste oblig 2".

2 Uke01

2.1 Template

Template for filen TestKlasse.java

```
public class TestKlasse {
    public static void main(String[] args) {
        // din kode her
    }
}
```

- `public` sikrer at du har samme navn på klassen som filnavnet
- `TestKlasse` er navnet på klassen – og skal alltid ha stor forbokstav og være lik filnavnet før filtype endelsen (.java)
- `public static void main(String[] args)` er den magiske metoden hvor programmet "starter". Legg merke til at den er indentert 4 spaces AKA 1 tab, og at den avsluttende krøllparentesen har samme indentering
- De to skråstrekene `//` er kommentartegn og vil kun vises i kildekodefila – veldig greit for å forklare en mindre selvforklarende kodedel

2.2 Skriv til terminal

For å skrive ut til terminal, bruk:
`System.out.println("Det du vil skrive ut");`

2.3 Variabler

- En variabel *deklareres* med *type* slik: `int myndighetsalder;` eller `String brukersNavn;` der hhv. `int` og `String` er typer, mens `myndighetsalder` og `brukersNavn` er variabelnavn.
- En variabel får verdi gjennom *initiering*: `myndighetsalder = 18;` & `brukersNavn = "Gard";`. Dette kan skje samtidig som *deklarering*: `int myndighetsalder = 18;` & `String brukersNavn = Gard;`.

2.4 Scanner og innlesning fra terminal/bruker

- For å lese inn fra bruker trenger du `Scanner`, og da er det tre ting som trengs:

  1. Man må legge til følgende linje *i toppen* av fila:
     `import java.util.Scanner;`
  2. Det må deklareres og initeres en variabel av type `Scanner`, eksempel:
     `Scanner tastatur = new Scanner(System.in);`, kun variabelnavn (her `tastatur`) er valgfritt
  3. Hente neste input/linje fra bruker som `String` med `tastatur.nextLine();`, eksempel:
     `String fraBruker = tastatur.nextLine();`
     (**OBS:** Husk å gi beskjed til bruker med `System.out.println("Beskjeden");` om hva som forventes at skal skrives inn.)

2.5 Uttrykkstest if/else if/else

Uttrykkstestene `if`, `else if` og `else` må komme i denne rekkefølgen, og delene i `if` og `else if` skjer kun hvis uttrykket de tester er `true` (det motsatte er `false`, som er de to eneste `boolean` typene). På tester er det veldig viktig og lurt å indentere riktig for å være sikker på at krøllparentesene er korrekt, så ikke en `else` plutselig havner "inni" en `if`-test. Et større eksempel er på slutten av dette dokumentet.

2.5.1 Beslutningstest symboler

```
// == eksakt likhet
4 == 4 // er sann
4 == 5 // er usann
// != ulikhet
4 != 4 // er usann
4 != 5 // er sann
// < strengt mindre enn
4 < 4 // er usann
4 < 5 // er sann
// <= mindre eller lik
4 <= 4 // er sann
4 <= 5 // er sann
// > strengt større enn
4 > 4 // er usann
4 > 5 // er usann
```

```
// >= større eller lik
4 >= 4 // er sann
4 >= 5 // er usann

// de neste to kan ikke brukes mellom to variabler, med mindre de er
             av type 'boolean',
// de brukes derfor oftest mellom to sjekker, for å koble flere
             sammen
// && er logisk AND, som betyr at begge må være sann
4 == 4 && 4 == 5 // dette blir SANN AND USANN, og totalt blir det da
             USANN
4 == 4 && 4 != 5 // dette blir SANN AND SANN, totalt SANN altså
// derfor kan du tenke at begge sider må være sann, for at AND skal
             bli sann
// || er logisk OR, som betyr en må være sann
4 == 4 || 4 == 5 // dette blir SANN OR USANN, som totalt blir sann
             siden den første er sann
4 > 4 || 4 <= 5  // dette blir USANN OR SANN, som totalt blir sann,
             siden den siste er sann
// man trenger altså bare en sann for å få totalt sann med OR
```

2.6 Type konvertering (String til int)

- Hvis du har en verdi av type `String`, men vil ha den til `int` må du benytte `Integer.parseInt(strengen)`.
- Det vanligste eksempelet er alder: `int alder = Integer.parseInt("19");`. I dette eksempelet er verdien (`"19"`) gitt direkte, men den kunne også vært lagret i en variabel først:

```
String tekstAlder = "19";
int alder = Integer.parseInt(tekstAlder);
```

2.6.1 int til String

De to vanligste metodene er:

- Å bare legge tallet sammen med en tom String `""`:
  `String tilString = 2 + "";`
- Eller aa bruke `new String`:
  `String tilString = new String(2);`

2.7 void metoder uten parametre

Void metoder uten parametre har formen:

```
public static void metodeNavn() { // metodenavn har alltid liten forbokstav
    //din kode her indentert et hakk
}
```

- Der man velger et passende selvvalgt metodeNavn som representerer hva metoden gjør.
  Husk: Alltid liten forbokstav på metodenavn og en indentering mellom krøllparentes start/slutt.
- Du kan ikke sette metoder inni andre metoder, men de kan være under eller over andre metoder - se eksempelkoden.

2.8 Avsluttende komplett eksempel uke01

```
import java.util.Scanner;

public class TestKlasse {

        public static void main(String[] args) {
                System.out.println("Velkommen til INF1000");

                // Oppsett av variabler
                int brukersAlder;
                int myndighetsAlder = 18;
                String brukersNavn;
                Scanner tastatur = new Scanner(System.in);

                // Hent informasjon fra bruker
                // Foerst navn
                System.out.println("Skriv inn navnet ditt");
                brukersNavn = tastatur.nextLine();

                // Deretter alder
                System.out.println("Skriv inn din alder");
                String brukersAlderSomString = tastatur.nextLine();
                // Konverterer til typen 'int'
                brukersAlder =
              Integer.parseInt(brukersAlderSomString);

                // Skriver ut beskjed til bruker - navnet foerst og er
              likt uansett
                System.out.println("Hei " + brukersNavn + "!");
                // Deretter om han er myndig eller ei, som er resultat
              av alderen bruker skrev
                if (brukersAlder < myndighetsAlder) {
                        System.out.println("Med alder paa " +
              brukersAlder + " er du nok ikke myndig (enda)");
                } else if (brukersAlder == myndighetsAlder) {
                        System.out.println("Joess - du er akkurat
              myndig i aar med " + brukersAlder + " som alder!");
                } else {
                        // Kan regne med 'int' typene, men ikke med
              String
                        int aarSidenMyndig = brukersAlder -
              myndighetsAlder;
                        System.out.println("Det er " + aarSidenMyndig
              + " aar siden du var myndig" );
                }
        }

        public static void linjeskift() {
                // Metoden min 'linjeskift' kaller paa en tom println
              for aa kun faa et linjeskift
                System.out.println();
        }

}
```

3 Uke02

3.1 Metoder med parametre

- Alltid angi hvilken `type (String, int, double)` man forventer at hver parameter skal ha
- Husk å indenter riktig her også
- Etter at metoden er blitt "ferdig" vil variablene i hovedmetoden fortsatt ha samme verdi som før
- For å finne ut når en parameter metode er lurt og hvordan, se etter mønster på likheter med små variasjoner:
  - Se for dere at du en main-metode som ser noe sånn her ut:

```
public static void main(String[] args) {
    String navn = "Gard";
    System.out.println("Hei " + navn); // skriver ut en velkomstmelding
    navn = "Odin";
    System.out.println("Hei " + navn); // skriver ut samme velkomstmeldin
}
```

  - Siden programmerere er late, og ser et mønster i utskriften der `"Hei "` er likt, og `navn` varierer, kan man opprette en metode som har det like som innhold, og får det varierende som parameter:

```
public static void siHei(String navnet) {
    // bruker annet navn på parameteren for aa vise at at parameternavnet
    System.out.println("Hei " + navnet);
}
```

  - Med metoden laga over kan alt endres til:

```
public static void main(String[] args) {
    String navn = "Gard";
    siHei(navn);   // Bruker metoden, og gir med verdien i 'navn' som par
    navn = "Odin"; // Har byttet verdien i navn til "Odin"
    siHei(navn);   // det er da den nye verdien som blir med
}
```

```
public static void siHei(String navnet) {
    // bruker annet navn på parameteren for aa vise at at parameternavnet
    System.out.println("Hei " + navnet);
}
```

3.2 Løkker med while

- En while løkke ser ut som en if-test, forskjellen er at når den er ferdig vil den sjekke om uttrykket fortsatt er sant.
- Siden en while løkke vil sjekke om den skal kjøre igjen når den er ferdig er det viktig å sikre en "slutt kriterie", for eksempel når brukeren taster en spesifikk input eller en teller har nådd sin grense.

```
Scanner tastatur = new Scanner(System.in);
System.out.println("Skriv inn teksten du vil gjenta, eller q for aa stoppe");
String input = tastatur.nextLine();

while(input.equals("q") == false) { // saa lenge input ikke er lik 'q'
    System.out.println(input);
    input = tastatur.nextLine();
}
```

3.3 Arrays

- En array brukes for å slippe å måtte lage en variabel for hver verdi man vil lagre.
- Den kan bare lagre verdier av samme type, f.eks `int` eller `String`.
- Alle arrayer har `[ ]` på typen for å markere at det er en array, og ikke en enkel variabel.
- Arrayene begynner å telle elementer på posisjon 0, så det første elementet har indeks = '0'.
- Når man lager en tom array må man bestemme hvor stor arrayen skal være: `int[] array = new int[5];`

- Det er også mulig å lage arrayer automatisk med innhold uten å bestemme lengden ved å ha "{}" rundt innholdet og et komma mellom hver verdi: `int[] array = {1, 2, 3, 2, 1};`

```
int[] array = {1, 2, 3, -4, 1};
array[0] = 3; // dette sier at verdien på den første posisjonen (indeks = 0) skal
System.out.println(array[3]); // dette skriver ut den fjerde verdien, altsaa -4
```

3.4 Arrays og while løkker

- En sterk fordel med løkker er at de automatisk kan jobbe seg gjennom et array av verdier
- For eksempel kan man bruke en teller for å akksessere hvert element i arrayen

```
int[] array = {1, 2, 3, -4, 1};
int lengde = array.length; // alle arrayer har .length for aa hente lengde
int teller = 0; // starte
while (teller < lengde) { // saa lenge telleren ikke har naadd lengden av array
    // Skriv ut verdien paa teller-posisjonen i arrayet
    System.out.println(array[teller]);
    // deretter oek teller verdien, saa vi til slutt faar stoppet loekka
    teller++;
}
```

3.5 Ressurser

Se github for kodeeksempler fra seminartimen.