Metoder med parametre, løkker og arrayer

Løse problemer med programmering

INF1000, uke3
Ragnhild Kobro Runde

METODER MED PARAMETRE

Statiske void-metoder med parametre

- Den typen metoder vi så på forrige uke gjorde alltid eksakt det samme når den ble kalt
  - Det er sjelden av nytte!
- For å være nyttig må en slik metode kunne tilpasses
  - Det gjør vi ved å sende inn parametre

Din første metode med parametre

- println er en metode hvor utfallet tilpasses!
- System.out.println(String text):
  skriver verdien i text til skjermen
  - Variabelen text er en parameter
  - Verdien vi gir inn ("hallo verden") når vi kaller println er et argument
- Parameter og argument er to sider av det samme
  - Parameter: variabel i metode som tar i mot verdi
  - Argument: verdi sendt inn når metode kalles

Metoder håndterer også redundans

- Man har ofte behov for (omtrent) samme funksjonalitet ulike steder i en kode
- Man ønsker da ikke å duplisere koden
  - Minsker oversiktlighet av kode
  - Endringer og rettinger må utføres mange steder
- Man samler i stedet funksjonaliteten i en metode og kaller metoden der den trengs
  - Dersom det er noe variasjon i hva man trenger, representerer man det som varierer med en parameter

Metoder med parametre

- {U3VoidMetodeParameter1.java-
  U3VoidMetodeParameter3.java}

Test-caser

- Ved forgreninger: minst en test for hvert alternativ.
  - Barn, voksen, eldre.
- Test spesielt grensetilfeller.
  - 18, 66, …
- Test eventuelt for typiske feilsituasjoner.
  - Hva hvis alder er negativ?

Ulike praktikaliteter (av begrenset betydning)

Praktikaliteter

- Det er greit å kjenne til noen snarveier (forkortelser) når man skal kode.
- Dette er egentlig uviktige teknikaliteter, men uansett greit å lære (for å kunne jobbe raskere i praksis).
- Formålet er igjen å bruke mindre energi på selve kodingen, og ha mer fokus tilgjengelig for problemløsning

Forkortinger rundt variabler

- Flere variabler kan deklareres på samme linje:

```
int tall1, tall2, alder;
```

- Man kan deklarere og gi verdi på samme linje:

```
int alder = 6;
```

- Forenklet skrivemåte for å oppdatere verdi:

```
alder += 5 (samme som: alder = alder+5)

alder++ (samme som: alder = alder+1)
```

Vær oppmerksom på

- Multiplikasjon må alltid angis eksplisitt med *
  - Feil: int prod = 10 a;
  - Riktig: int prod = 10 * a;
- = er noe helt annet enn ==
  - = brukes for å sette verdien til en variabel
  - == brukes for å sammenligne to verdier

Tegnsett

- En datamaskin representerer hver bokstav med en bestemt tallkode
- Det finnes ulike standarder for hvilke tall som representerer hvilke bokstaver
  - En klassisk standard er ASCII som støtter 128 tegn
  - En moderne standard er Unicode (UTF-8) som støtter over 100 000 tegn, og brukes på IFIs maskiner
  - Tegnsett er mer intrikat enn man kunne tro, og kan være kilde til mye krøll

Tegnsett

- Bruk av æ,ø,å i Java kan føre til krøll:
  - Bruk av æ,ø,å i variabelnavn o.l. i Java fører fort til krøll
  - Bruk av æ,ø,å i kommentarer eller tekststrenger burde gå bra, men kan også føre til problemer

Tegnsett

- Vi skal lære programmering,
ikke skrive stil,
og taper ingenting på å være uten æ,ø,å:
  - Skriv "forste aaret laerte jeg mye" i kodefilen,
og fokuser på det viktige i faget!

God programmerings-skikk

- Sørg for at programmet er oversiktlig å lese
- Legg inn blanke linjer der det øker lesbarheten
- Legg inn kommentarer der noe bør forklares (//)

God programmerings-skikk (forts.)

- Velg beskrivende navn fremfor korte navn:
  - antallStudenter er oftest bedre enn as
  - og i hvertfall ikke minVariabel for samme formål

God programmerings-skikk (forts.)

- Bruk store bokstaver for å skille de ulike delene av variabelnavnet (camelCase)
  - ikke: antallstudenter
  - men: antallStudenter

LØKKER

Et eksempel på repetert kjøring

- {U3RiktigPlussing1.java-U3RiktigPlussing2.java}

Repetert kjøring (løkke): while

- Syntaks:
  while (condition) {do1; do2; ...}

- Eksempel:

```
int tall = 0;
while (tall<100) {
   System.out.println(tall);
   tall = tall + 5;
}
```

- En slags if med tilbakekobling – kjører innholdet mange ganger, helt til condition ikke lenger er sann (true)

Merk den presise rekkefølgen ting blir gjort!

- while (condition) {do1; do2; ...}
  - Man sjekker,
    kjører hele blokka,
    og går så tilbake til sjekken igjen
  - Sjekk condition,
    do1,
    do2,
    sjekk condition igjen,
    do1,
    do2 ...

Merk den presise rekkefølgen ting blir gjort!

- while (condition) {do1; do2; ...}
- Det blir altså **ikke** sjekket noe mellom do1 og do2
  - Det som sjekkes er oppfylt når man starter å kjøre kodeblokka
  - .. men det kan slutte å være oppfylt underveis i blokka

En liten oppgave

- Skriv kode (det essensielle) som regner ut summen av tallene fra 1 til 100 (1+2+3...+100)
  - Prøv alene med blyant og papir i 3 minutt
  - Diskutér med nabo i 3 minutt
- {U3SumTallrekkeVhaWhile.java}

Eksempel

- {u3LokkerMedMetodekall.java}: repetert spørring frem til brukeren oppgir negativ alder

ARRAYER

Eksempel

- U3HoydeGittAlder1.java

Vi kan slå opp verdien vi trenger direkte!

- Det vi ønsket:
  - hoydeAaralder
- Syntaks i java:
  - hoydeAar[alder];
- Og før dette må vi definere hoydeAar som en array:
  - int[] hoydeAar = {50, 76, 87, 96};

Array

- Definere array:
  - int[] hoydeAar = {50, 76, 87, 96};
  - int[] hoydeAar = new int[4];
- Sette en enkeltverdi:
  - hoydeAar[0] = 5;
  - hoydeAar[2] = 8;
- Bruke enkeltverdi
  - System.out.println(hoydeAar[0]);

Array - merk

- Størrelsen på en array kan bestemmes av en annen variabel
  - int hvorMangeHoyder = Integer.parseInt(in.nextLine());
  - int[] hoydeAar = new int[hvorMangeHoyder];
- Alle verdier må være samme type, men man kan bestemme hvilken
  - String[] hoydeAar = {"uhyre liten", "bitteliten", "liten"}
- En tom array fylles med default-verdier
  - int[] hoydeAar = new int[4];
  - er samme som: int[] hoydeAar = {0,0,0,0}

LØKKER OG ARRAYER

Løkker og arrayer

- Løkker og arrayer og arbeider ofte godt sammen!
  - Løkker gjør at kodelinjer kan kjøres flere ganger
  - Arrayer gjør det mulig å jobbe med mange verdier
- {U3HuskFemOrd1.java-U3HuskFemOrd2.java}

Oppsummering

- Metoder med parametre gjør at koden kan skreddersys lignende situasjoner
- Løkker gjør at kodelinjer kan kjøres flere ganger
  - F.eks. gjøre lignende type utregning på ulike verdier
- Arrayer gjør det mulig å jobbe med mange verdier
  - Kan slå opp direkte på verdien i en bestemt posisjon
- Løkker og arrayer jobber godt sammen
  - Kan generere og oppsummere store mengder verdier
- Dere har nå verktøykassen for å løse skikkelige problemstillinger!