UNIVERSITETET
I OSLO

INF1000: Forelesning 4

Mer om arrayer

Metoder

UNIVERSITETET
I OSLO

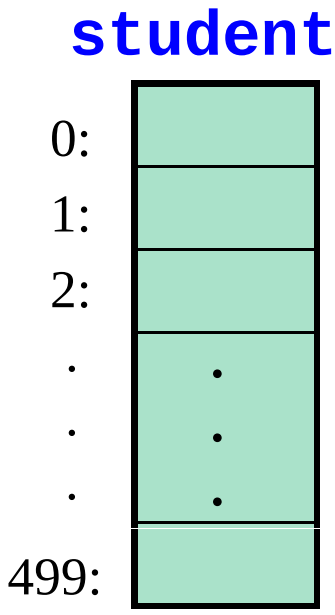
MER OM ARRAYER

Array som en samling verdier

- Anta at vi ønsker å lagre en liste med navnene på alle INF1000-studentene:

```
String[] studenter = new String[500];
```

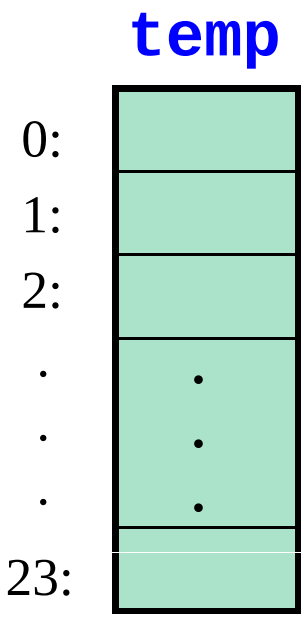
- Resultatet kan visualiseres (tegnes) slik:



Array som indeksert liste

- Anta at vi ønsker å lagre temperaturen for hver hele time gjennom ett døgn:

```
double[] temp = new double[24];
```



Eksempel på traversering av arrayer

- Beregning av gjennomsnittstemperatur gjennom døgnet:

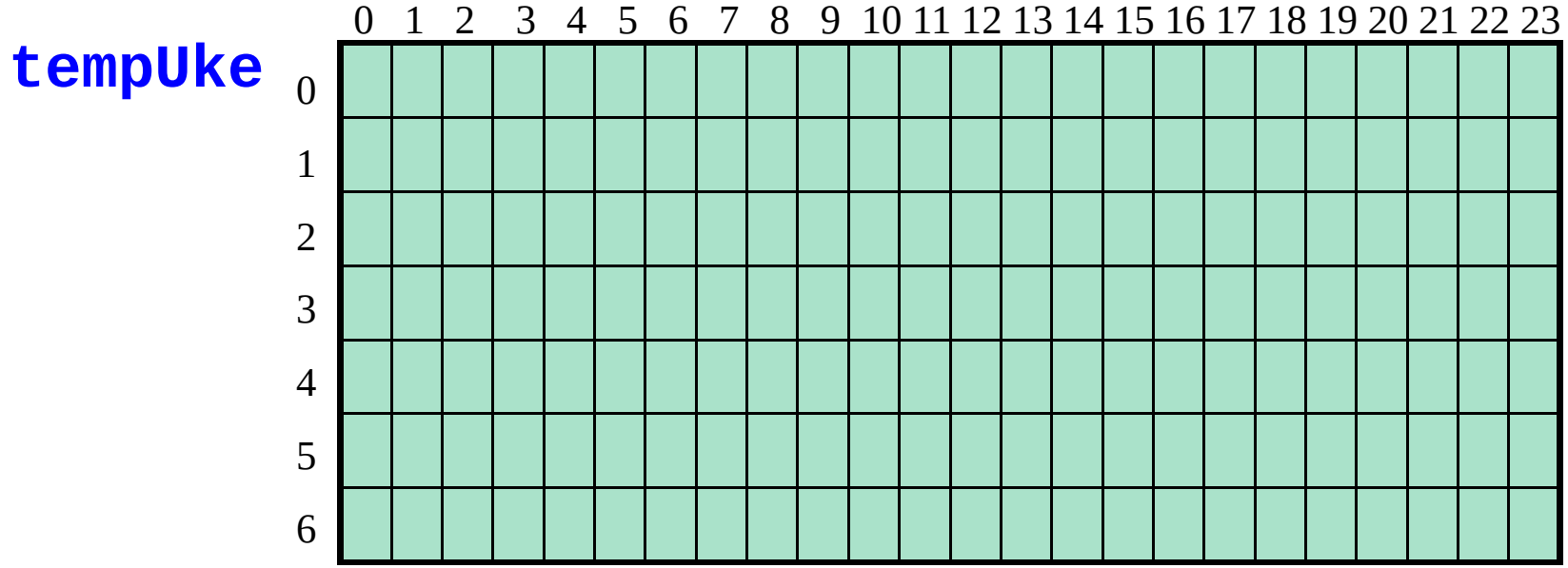
```
double sum = 0;

for (int i = 0; i < temp.length; i++) {
    sum = sum + temp[i];
}

System.out.println("Gjennomsnittstemperatur: ”
    + sum/temp.length);
```

Todimensjonale arrayer

- Anta at vi ønsker å lagre times-temperaturen ikke bare for et døgn, men for en hel uke:

```
final int ANT_DAGER = 7;
final int ANT_TIMER = 24;
double[][] tempUke = new double[ANT_DAGER][ANT_TIMER];
```

- Dette kan vi tenke på som en tabell med 7 rader og 24 kolonner:



Traversering av todimensjonal array

- Beregning av gjennomsnittstemperatur gjennom en hel uke:

```
double sum = 0;

< for hver dag: >
  < for hver time denne dagen: >
      < øk sum med registrert temperatur >



int totalTimer = ANT_DAGER * ANT_TIMER;

System.out.println("Gjennomsnittstemperatur: " +
  sum/totalTimer);
```

Traversering av todimensjonal array

- Beregning av gjennomsnittstemperatur gjennom en hel uke:

```
double sum = 0;

for (int dag = 0; dag < ANT_DAGER; dag++) {
  for (int time = 0; time < ANT_TIMER; time++) {
    sum = sum + tempUke[dag][time];
  }
}

int totalTimer = ANT_DAGER * ANT_TIMER;

System.out.println("Gjennomsnittstemperatur: " +
  sum/totalTimer);
```

Flerdimensjonale arrayer

- Arrayer kan ha så mange dimensjoner vi vil, for eksempel om vi ønsker å lagre times-temperaturen for hver uke i et helt år:

```
final int ANT_UKER = 52;
final int ANT_DAGER = 7;
final int ANT_TIMER = 24;

double[][][] tempAar =
  new double[ANT_UKER][ANT_DAGER][ANT_TIMER];
```

Vanlige feil ved bruk av arrayer

TO EKSEMPLER FRA FØRSTE FORELESNING

TemperaturKonvertering

UNIVERSITETET I OSLO

```
class TemperaturKonvertering {
    public static void main (String[] args) {
        double tempCelcius, tempFahrenheit;
        System.out.println("Celcius Fahrenheit");

        tempCelcius = -10;
        tempFahrenheit = 9 * tempCelcius / 5 + 32;
        System.out.println(tempCelcius + " " + tempFahrenheit);

        tempCelcius = 0;
        tempFahrenheit = 9 * tempCelcius / 5 + 32;
        System.out.println(tempCelcius + " " + tempFahrenheit);

        tempCelcius = 37;
        tempFahrenheit = 9 * tempCelcius/ 5 + 32;
        System.out.println(tempCelcius + " " + tempFahrenheit);

        tempCelcius = 100;
        tempFahrenheit = 9 * tempCelcius / 5 + 32;
        System.out.println(tempCelcius + " " + tempFahrenheit);
    }
}
```

- Her gjøres nesten det samme fire ganger – for fire ulike Celcius-verdier. Kan dette gjøres på en bedre måte?

Gangetabell

UNIVERSITETET I OSLO

```
class Gangetabell {
    public static void main (String [] args) {
        System.out.println(1 * 8);
        System.out.println(2 * 8);
        System.out.println(3 * 8);
        System.out.println(4 * 8);
        System.out.println(5 * 8);
    }
}
```

- Hvordan kan vi bruke samme program til å regne ut ikke bare 8-gangen, men en hvilken som helst n-gange?
- Hvordan kan vi angi – og variere – hvilken del av tabellen som skal skrives ut?

Løsningen:

METODER

Metoder vi har brukt allerede

- Vi har brukt en del metoder allerede, for eksempel

```
System.out.println("  *  ");
double d = tastatur.inDouble();
String s = tastatur.inWord();
int i = (int) Math.round(d);
char c = s.charAt(0);
```

- Også

```
public static void main(String[] args) {
    …
}
```

  er en metode.

- Vi skal nå se nærmere på hva metoder egentlig er, hvordan de brukes og hvordan vi kan lage våre egne metoder.

Hva er en metode?

- En metode er en navngitt blokk med instruksjoner som vi kan få utført ved å angi metodens navn.
- Eksempel på (deklarasjon av) metode:

```
void skrivPyramide() {
    System.out.println("  *  ");
    System.out.println(" *** ");
    System.out.println("*****");
}
```

- Bruk av (kall på) metoden:

```
skrivPyramide();
```

Plassering i programmet

- Foreløpig samler vi alle metoder i en egen klasse:

Filen PyramideProgram.java:

```
class PyramideProgram {
    public static void main(String[] args) {
        Pyramide p = new Pyramide();
        p.skrivPyramide();
    }
}

class Pyramide {
    void skrivPyramide() {
        System.out.println("  *  ");
        System.out.println(" *** ");
        System.out.println("*****");
    }
}
```

Returverdier

- Ofte ønsker vi at resultatet fra en metode skal kunne brukes videre i resten av programmet, som for eksempel:

```
In tastatur = new In();

int k = tastatur.inInt();
```

- Her er **`inInt()`** en metode i klassen **`In`**, som vi får tilgang til via variabelen **`tastatur`**.
- Metoden **`inInt()`** **returnerer** et heltall (en **int**).
- Dette heltallet tar vi vare på i variabelen **`k`**.

Metode med returverdi

- Her er en metode som leser et heltall og returnerer det dobbelte:

typen til returverdien

```
int lesOgDoble() {
    In tastatur = new In();

    System.out.print("Skriv et tall: ");
    int tall = tastatur.inInt();

    return 2*tall;
}
```

verdien som returneres

Eksempel: metoden lesPositivtHeltall()

- Metoden **`inInt()`** leser inn et vilkårlig heltall. Noen ganger ønsker vi å sikre oss at vi får et **positivt** heltall. Vi kan da lage en egen metode for dette:

```
int lesPositivtHeltall() {
    In tastatur = new In();
    int tall;

    do {
        System.out.print("Gi et positivt tall: ");
        tall = tastatur.inInt();
    } while (tall <= 0);

    return tall;
}
```

Eksempel:
bruk av lesPositivtHeltall()


```
import easyIO.*;

class LesOgSkrivTall {
    public static void main(String[] args) {
        Innlesing il = new Innlesing();
        int i = il.lesPositivtHeltall();
        System.out.println(i + " er et positivt heltall");
    }
}

class Innlesing {
    int lesPositivtHeltall() {
        In tastatur = new In();
        int tall;
        do {
            System.out.print("Gi et positivt tall: ");
            tall = tastatur.inInt();
        } while (tall <= 0);
        return tall;
    }
}
```

Parametre

- Ofte ønsker vi at samme metode skal kunne brukes for litt ulike input-verdier, som for eksempel:

```
System.out.println("  *  ");
System.out.println("Hei verden");
```

- Her er **`println`** en metode som tar en tekst som input (**parameter**).

UNIVERSITETET I OSLO

Eksempel: metode som konverterer fra Celcius til Fahrenheit

- På første forelesning så vi omregningsformelen

  TF = 9 * TC / 5 + 32

- Vi lager en egen metode som gjør beregningen for en gitt Celcius-verdi:

```
void konverterTilFahrenheit(double tempCelcius) {
    double tempFahrenheit;
    tempFahrenheit = 9 * tempCelcius / 5 + 32;
    System.out.println(tempCelcius + " "
                                   + tempFahrenheit);
}
```

Bruk av metoden

```
class TemperaturKonvertering {
    public static void main(String[] args) {
        Konverterer konv = new Konverterer();
        konv.konverterTilFahrenheit(-10);
        konv.konverterTilFahrenheit(0);
        konv.konverterTilFahrenheit(37);
        konv.konverterTilFahrenheit(100);
    }
}

class Konverterer {
    void konverterTilFahrenheit(double tempCelcius) {
        double tempFahrenheit;
        tempFahrenheit = 9 * tempCelcius / 5 + 32;
        System.out.println(tempCelcius + " " + tempFahrenheit);
    }
}
```

Metode med returverdi og parametre

- I stedet for å skrive ut temperaturen, kan vi la `konverterTilFahrenheit`-metoden returnere temperaturen i Fahrenheit:

```
double konverterTilFahrenheit(double tempCelcius) {
    double tempFahrenheit;
    tempFahrenheit = 9 * tempCelcius / 5 + 32;
    return tempFahrenheit;
}
```

Eksempel: gangetabell-metode

- Metode som skriver ut n-gangen fra 1 til 10:

```
void gangeTabell(int n) {
    for (int i = 1; i <= 10; i++) {
        System.out.println(i * n);
    }
}
```

- Eksempel på kall på metoden:

```
gangeTabell(2);
gangeTabell(15);
```

Bruk av metoden

UNIVERSITETET I OSLO

```
import easyIO.*;

class GangeProgram {
    public static void main(String[] args) {
       In tastatur = new In();
       Utregning utr = new Utregning();

       System.out.print("Hvilken gangetabell vil du skrive? ");
       int tall = tastatur.inInt();

       utr.gangeTabell(tall);
    }
}

class Utregning {
    void gangeTabell(int n) {
       for (int i = 1; i <= 10; i++) {
           System.out.println(i * n);
       }
    }
}
```

Eksempel med flere parametre

- Vi kan utvide gangeTabell-metoden med en parameter som angir hvor mye av tabellen som skal skrives:

```
void gangeTabell(int n, int slutt) {
    for (int i = 1; i <= slutt; i++) {
        System.out.println(i * n);
    }
}
```

- Eksempel på kall på metoden:

```
gangeTabell(2,20);
gangeTabell(15,10);
```

Oppsummering: Deklarasjon av metoder

- De metode-deklarasjonene vi har sett på så langt har følgende form:

hva slags output metoden gir, for eksempel void, int, double, char, …

et navn som vi velger – bør beskrive hva metoden gjør

```
returverditype metodenavn (parametre) {
    setning 1;
    setning 2;
    ....
    setning n;
}
```

hva slags input metoden skal ha – gis i form av variabel-deklarasjoner separert av komma

- Hvis returverditypen er noe annet enn void, må metoden inneholde minst en return-setning (typisk som den siste i metoden)

Metodekall

- Når vi benytter en metode sier vi at vi **kaller på metoden**.

- Kall på metode uten parametre – eksempel:

```
minMetode();
```

- Kall på metode med parametre – eksempel:

```
minMetode2(2,"Hei");
```

- Kall på metode som returnerer en verdi – eksempel:

```
int i = minMetode3(10);
```

Parametre og argumenter

```
class MittProgram {
    public static void main(String[] args) {
        Kalkulator k = new Kalkulator();
        double pris = 100.0;
        double nyPris = k.trekkFraRabatt(pris);
        System.out.println("Utsalgspris: " + nyPris);
    }
}

class Kalkulator {
    double trekkFraRabatt(double x) {
        return x * 0.8;
    }
}
```

argument

parameter

- Merk: argumenter kalles også for **aktuelle parametre**, mens parametre da kalles **formelle parametre**.

Parametre og argumenter

- **Parametre**:
  Deklareres mellom parentesene i toppen (= den første linjen) av metode-deklarasjonen. De er ”vanlige variable” som bare eksisterer inne i metoden og så lenge denne eksekverer.
- **Argumenter**:
  Verdier som oppgis mellom parentesene når vi **kaller på** en metode.
- **Antall argumenter**:
  MÅ samsvare med antall parametre i metoden, ellers får vi komplieringsfeil.
- **Argumentenes datatyper**:
  MÅ samsvare med datatypen til tilsvarende parameter.