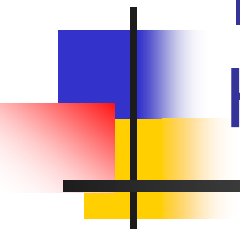


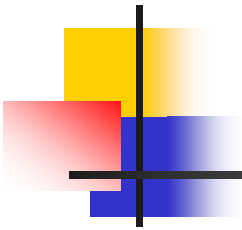
Inf1000 (Uke 10)
HashMap og ArrayList

Grunnkurs i programmering

Institutt for Informatikk

Universitetet i Oslo

Are Magnus Bruaset og Anja Bråthen Kristoffersen



Bibliotekpakker i Java

Mange pakker i java, som hver inneholder:

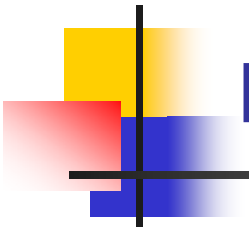
- klasser (In, Out, StringTokenizer, ...)
- metoder (Math.random, outln, endOfFile, nextToken, ...)
- evt. konstanter (Math.PI)

Vi har hittil brukt metoder fra pakkene:

easyIO.*

java.util.*

Math.*



Bibliotek

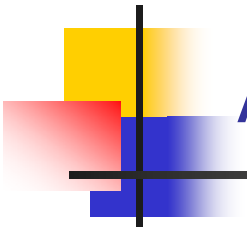
- Sun har laget en oversikt over alle tilgjengelige klasser og pakker i biblioteket:

  http://java.sun.com/j2se/1.4.2/docs/api/index.html

- For å hente inn de delene man trenger i programkoden fra biblioteket brukes import

  ```
  import java.util.*
  ```

- Vi skal i dag se på to av pakkene i java.util.*
  - ArrayList
  - HashMap



ArrayList

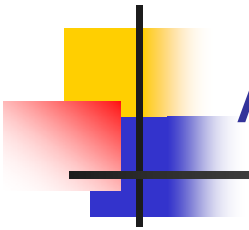
Klassen ArrayList fungerer *nesten* som vanlige array-er.

Vi må hente inn klassen fra biblioteket:

```
import java.util.ArrayList;
```

Objekter opprettes med new:

```
ArrayList a = new ArrayList();
```

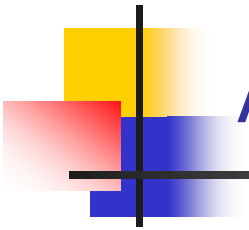


ArrayList: innsetting av objekter

- Vi setter inn objekter med metoden add:

```
a.add(p);      //objektet p settes inn sist i listen
a.add(n, p);   //objektet p settes inn på indexplass n
```

- Vi får bare sette inn i posisjoner som er i bruk, eller én høyere!



ArrayList: oppslag i objekter

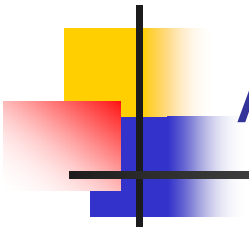
- Vi henter elementer med metoden get:

```
p = (MyClass) a.get(n);
```

- Vi må angi klassen til det objektet vi henter ut her indikert med (MyClass) som kan være f.eks. (String)

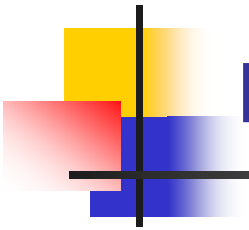
- Antall elementer i ArrayList a får man med metoden size:

```
int lengde = a.size();
```



ArrayList sammenliknet med arrayer

- **Fordeler**
  - Man behøver ikke anslå noen størrelse når man oppretter strukturen. ArrayList-objektet vil utvide seg selv automatisk
  - ArrayList har noen flere operatorer, for eksempel for søking
- **Ulemper**
  - ArrayList er mer kronglete å bruke



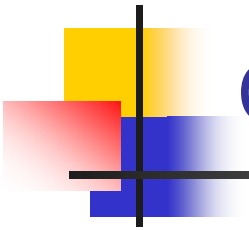
Problem

- Anta at vi ønsker å lage et program som leser inn linje for linje fra fil. Deretter skal programmet skrive ut linjene i motsatt rekkefølge.

- Vi lager først programmet ved bruk av vanlig array så ved bruk av ArrayList.

```
import easyIO.*;
class Array1 {
  public static void main (String arg[]) {
    String line[] = new String[1000];
    In file = new In(arg[0]);
    int nLines = 0;
    while (! file.endOfFile()){
      line[nLines++] = file.inLine();
    }
    file.close();
    Out skjerm = new Out();
    while (nLines > 0){
      skjerm.outln(line[--nLines]);
    }
  }
}
```

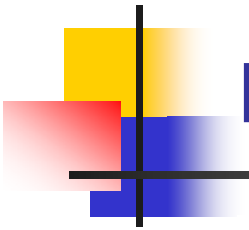
```
import easyIO.*;
import java.util.ArrayList;

class Array2 {
  public static void main (String arg[]) {
    ArrayList a = new ArrayList();
    In file = new In(arg[0]);
    while (! file.endOfFile()){
      a.add(file.inLine());
    }
    file.close();
    Out skjerm = new Out();
    for (int i = a.size()-1; i >= 0; --i){
      skjerm.outln((String) a.get(i));
    }
  }
}
```



Oppsummering ArrayList

- Lengden av listen er ikke bestemt ved oppretting
- Objekter blir satt inn i listen sekvensielt, evt. skriver vi over andre objekter i listen
- For å hente ut objekter fra listen må klassen til objektet være kjent, samt posisjonen til objektet i listen

  Eksempel: **(String) a.get(i);**
- Listen kan bestå av forskjellige objekter



HashMap

- En HashMap er en form for tabell (i likhet med arrayer og ArrayLister)
- HashMap kan brukes til å holde orden på mange objekter.

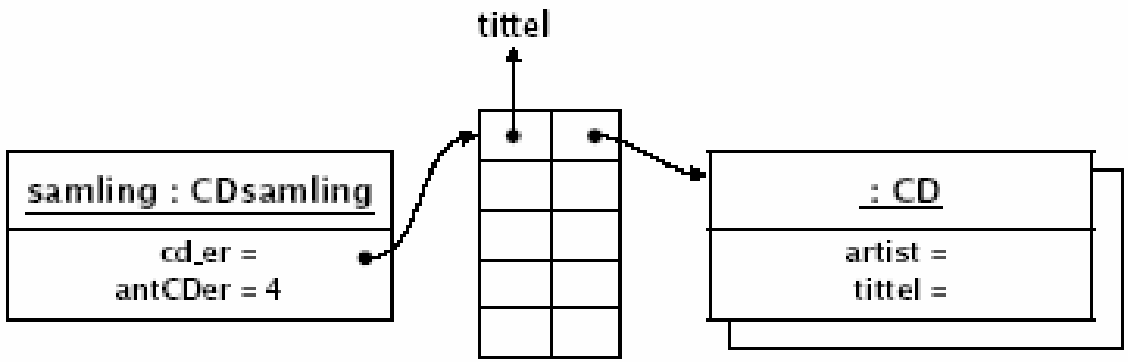
Forskjellen mellom array/ArrayList og HashMap

- I array/ArrayList:
  - Legger vi inn objekter i en bestemt posisjon
  - Må bruke denne posisjonen ved senere uthenting
  - Indeksen er et heltall mellom 0 og (length-1)

- I HashMap:
  - Må oppgi en bestemt *nøkkel* (vanligvis en tekststreng) når vi legger inn et nytt objekt
  - Må oppgi samme nøkkel ved senere uthenting
  - Indeksen er en tekststreng

Med en HashMap kan man:

- legge inn nye objekter
- finne tilbake til et objekt som er lagt inn
- fjerne et objekt som er lagt inn
- løpe gjennom alle objektene i tabellen



Å opprette en HashMap

- I starten av programmet:

```
import java.util.*;
```

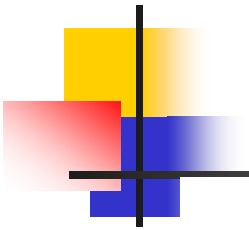
  Da importeres pakken java.util hvor bl.a. klassen HashMap ligger.

- I klassen eller metoden som skal bruke HashMap'en opprettes HashMap'en med:

```
HashMap tabell = new HashMap();
```

  Som for andre variable, hvis tabellen skal brukes av flere metoder i en klasse, deklareres variabelen ovenfor i starten av klassen (som en objektvariabel).

  Hvis tabellen kun skal brukes av en enkelt metode, er det naturlig å deklarere variabelen øverst (eller nesten øverst) inni denne metoden.



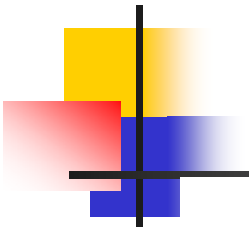
Å legge inn et objekt i en HashMap

- Et hvilket som helst objekt i Java kan legges inn i en HashMap
- Når vi legger et objekt inn i HashMap'en, må vi samtidig oppgi en nøkkel (en tekststreng) som entydig identifiserer objektet.

```
tabell.put(nøkkel, objekt);
```

- Vi trenger denne nøkkelen dersom vi senere skal finne (eller fjerne) objektet i HashMap'en.

```
MyClass x = (MyClass) tabell.get(nøkkel);
```



Å legge inn et objekt i en HashMap

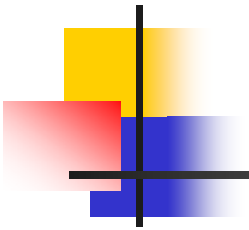
- Det kan være fint å kunne bruke en tekst (en String) som nøkkel, f.eks. når:
  - Finne data om en student ut fra hans eller hennes navn
  - Finne data om en kommune ut fra kommunens navn.
  - Finne en bil i et register ut fra bilnummeret
  - Finne en CD i CD-arkivet ut fra tittelen på CD-en

Konstruksjon av nøkkel

- Noen ganger konstrueres en nøkkel ut fra flere variable:

```
String lengdegrad = "67.3";
String breddegrad = "53.3";
String posisjon = lengdegrad + ";" + breddegrad;
tabell.put(posisjon, målestasjon);
```

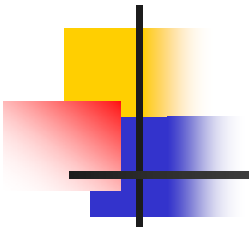
Å hente et objekt fra en HashMap



- For å hente et objekt med utgangspunkt i nøkkelen:

```
Person p = (Person) tabell.get("Jens");
```

- Legg merke til at vi i starten må skrive i parentes navnet på klassen som objektet tilhører - i dette tilfellet klassen Person.



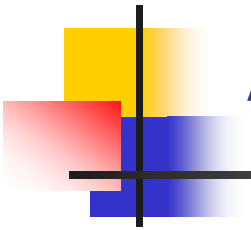
Å hente et objekt fra en HashMap

- HashMap'en ikke holder rede på hvilken klasse objektene som legges inn har
- Når objektene hentes ut må vi derfor fortelle Java hvilken klasse objektet har (egentlig et møte med en avansert mekanisme i objektorienterte språk som kalles *arv*, se INF1010).
- Merk: å hente et objekt fra en HashMap slik som over medfører *ikke* at objektet fjernes fra HashMap'en.

Overskriving av et objekt i en HashMap

- Dersom vi legger inn flere objekter med samme nøkkel, er det bare det sist innlagte objektet som blir liggende i tabellen (de andre overskrives):

```
Person p1 = new Person(...);
Person p2 = new Person(...);
Person p3 = new Person(...);
String navn = "Jens";
tabell.put(navn, p1); // p1 legges inn
tabell.put(navn, p2); // p2 legges inn, p1 overskrives
tabell.put(navn, p3); // p3 legges inn, p2 overskrives
```



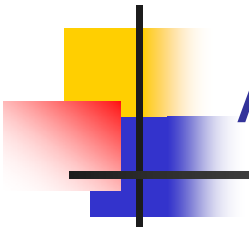
Å fjerne et objekt fra en HashMap

- For å fjerne et objekt med utgangspunkt i nøkkelen:

```
tabell.remove("Jens");
```

- Dersom det ligger et objekt i HashMap'en med den gitte nøkkelen, blir objektet fjernet og setningen ovenfor returnerer med en peker til objektet som fjernes.

- Dersom det ikke ligger et objekt i HashMap'en med den gitte nøkkelen, returnerer setningen ovenfor med verdien **null**.



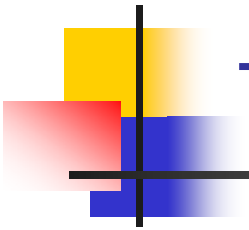
Å få fatt i alle objektene i en HashMap

- For å få fatt i alle objektene i en HashMap på en gang, lager vi en egen *oppramsing* av alle objektene i HashMap'en:

```
Iterator it = tabell.values().iterator();
```

- Deretter kan vi se på hvert enkelt objekt i HashMap'en ved å gå i løkke:

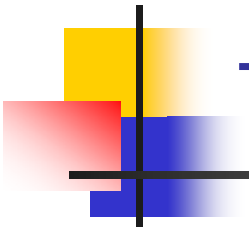
```
while (it.hasNext()) {
    Person p = (Person) it.next();
    <gjør noe med Person-objektet>
}
```



To måter å løpe gjennom en HashMap

- Løpe gjennom verdiene (som på forrige foil):

```
Iterator it = tabell.values().iterator();

while (it.hasNext()) {
  Person p = (Person) it.next();
  <gjør noe med objektet>
}
```



To måter å løpe gjennom en HashMap

- Løpe gjennom nøklene:

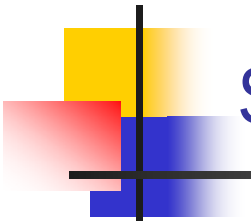
```
Iterator it = tabell.keySet().iterator();

while (it.hasNext()) {
  String nøkkel = (String) it.next();
  <gjør noe med nøkkelen>
}
```

Metoder i HashMap

Metode	Eksempel	Beskrivelse
put	`tabell.put(id, obj);`	Legg inn objekt
get	`(Person) tabell.get(id);`	Finn objekt
remove	`tabell.remove(id);`	Fjern objekt

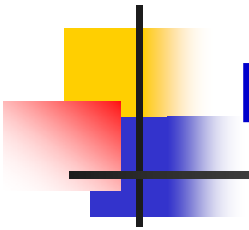
containsKey	`if (tabell.containsKey(id)){`
	`  // gjør et eller annet`
	`}`
		Sjekk om nøkkel finnes i tabell

values	`Iterator it =` `tabell.values().iterator();`	Lag oppramsing av objektene
keySet	`Iterator it =` `tabell.keySet().iterator();`	Lag oppramsing av nøklene



Skal vi bruke array, ArrayList eller HashMap?

- Array:
  - kan lagre et på forhånd spesifisert antall verdier eller objekter
  - hver tabell kan bare lagre verdier av én type
- ArrayList:
  - kan lagre et uspesifisert antall objekter
  - kan lagre objekter av ulik klasse
  - kan sette inn og fjerne objekter
- HashMap:
  - kan lagre et uspesifisert antall objekter
  - hver tabell kan lagre objekter av ulike klasser
  - kan sette inn og fjerne objekter i tabellen
  - kan gjøre oppslag ved hjelp av en nøkkel



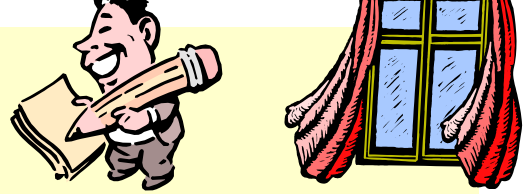
Et fullstendig eksempel

Vi lager et program som består av to klasser:

- class TestHashMap (inneholder bl.a. main-metoden)
- class Person (med objektvariablene navn og fnr)

Programmet skal lage en HashMap og et objekt av klassen Person, og skal legge sistnevnte objekt inn i HashMap'en, med "Petter" som nøkkel.

Deretter skal programmet - ut fra nøkkelverdien "Petter" - finne fram Person-objektet igjen i HashMap'en.



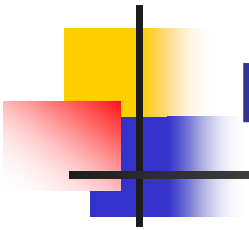
```
import java.util.*;

class TestHashMap {
  public static void main (String [] args) {
    TestHashMap test = new TestHashMap();
  }

  TestHashMap() {
    HashMap h = new HashMap();
    Person p1 = new Person("Petter", "12128646733");
    h.put("Petter", p1);
    Person p2 = (Person) h.get("Petter");
    p2.skrivUt();
  }
}

class Person {
    String navn;
    String fnr;

    Person (String navn, String fnr) {
      this.navn = navn;
      this.fnr = fnr;
    }

    void skrivUt() {
      System.out.println("Navn: " + navn);
      System.out.println("Fnr: " + fnr);
    }
}
```

Eksempel: ELIZA - den elektroniske psykiater

ELIZA: program skrevet av Joseph Weizenbaum i 1966 for studier av språklig kommunikasjon mellom mennesker og datamaskiner.

Programmet imiterer en psykiater, slik som i dette eksemplet:

Person : *Men are all alike.*
ELIZA : In what way?
Person : *They're always bugging us about something or other.*
ELIZA : Can you think of a specific example?
Person : *Well, my boyfriend made me come here.*
ELIZA : Your boyfriend made you come here?
Person : *He says I'm depressed much of the time.*
ELIZA : I am sorry to hear that you are depressed.
Person : *It's true. I am unhappy.*
ELIZA : Do you think coming here will help you not to be unhappy?
Person : *I need some help, that much seems certain.*
ELIZA : What would it mean to you if you got some help?
Person : *Perhaps I could learn to get along with my mother.*
ELIZA : Tell me more about your family



En samtalepartner

Vi skal nå lage et program som basert på en eksisterende assosiasjonsliste gjør det mulig å "snakke med maskinen", slik at hver gang maskinen gjenkjenner et ord gir den et bestemt (og tilnærmet fornuftig) svar.

- Det første programmet gjør er å gå i løkke for å lese inn assosiasjonslisten ”assosiasjoner.txt”. Den blir lagt inn i en HashMap med assosiasjonsordet som nøkkel.

- Deretter skal programmet gå i løkke og be brukeren skrive noe, hvoretter programmet skriver ut tilhørende svar (hvis et av ordene brukeren skriver er registrert i assosiasjoner.txt).

assosiasjoner.txt

```
mat     Liker du pizza?
pizza   Jeg liker ikke pizza.
sulten  Det er viktig å få i seg nok mat.
hallo   Heisan.
hei     Hei du. Fortell hvorfor du er kommet til meg.
morn    Morn du. Hva kan jeg hjelpe deg med.
heisan  God formiddag, og hva er ditt problem
…
```

Program skisse

```
import easyIO.*
import java.util.*

class Eliza1 {
   main()
}

class Psykiater {
   HashMap h = new HashMap();
   In tast = new In();
   Out skjerm = new Out();

   void lesFraFil() {
     //her skal vi lese dataene fra filen assosiasjoner.txt og lagre dem i en HashMap
   }
   void samtale() {
     // her skal vi lese inn tekst fra skjermen og lete gjennom HashMap-en etter respons
    }
}
```

Metoden lesFraFil()

```
void lesFraFil() {
   In innfil = new In("assosiasjoner.txt"); // åpner filen for lesing
   while (!innfil.lastItem()) {   // sjekker om det er mer igjen å lese på filen
         String s1 = innfil.inWord();   // leser inn assosiasjonsordet (nøkkelen)
         String s2 = innfil.inLine(); // leser inn responsen
         h.put(s1, s2); // lagrer responsen med assosiasjonsordet som nøkkel
   }
   innfil.close(); // lukker assosiasjoner.txt
}
```

Skisse til metoden samtale()

```
void samtale() {
   // be om respons fra bruker og les inn fra skjerm
   while(!avslutt){
      // Hvis ikke brukeren vil avslutte les inn setningen
      // For hver setning trenger vi å dele den opp i ord, teste om ordet er en nøkkel i h
      // Vi lager en klasse Setning med metodene flereOrd() og nesteOrd().
      // Gå i løkke gjennom hvert ord i setningen, for første ord i setningen som er en
      // nøkkel i h les responsen og skriv den ut på skjermen

      // Hvis ingen ord i setningen er nøkkel i h, be brukeren utdype svaret

      // Les inn nytt svar
   }
}
```

Klassen Setning

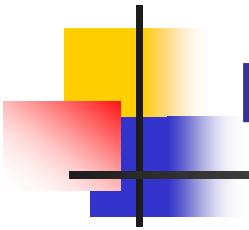
```
class Setning {
  String s;
  Setning(String t) {
     s = t + " ";  // legger et ekstra mellomrom etter setningen
  }

  boolean flereOrd() {
     while (s.length() > 0 && s.charAt(0) == ' ') {
       s = s.substring(1);  // leser over mellomrom
     }
     return (s.length() > 0);  // returnerer false når det ikke er flere ord i setningen
  }

  String nesteOrd() {
     int k = s.indexOf(' ');  // leser lengden til neste mellomrom
     String t = s.substring(0, k);  // leser neste ord
     s = s.substring(k);  // tar vare på resten av setningen i s
     return t;
 }
}
```

Metoden samtale()

```
void samtale() {
  skjerm.outln("Hei, jeg er din psykiater. Jeg skal prøve å hjelpe deg.");
  skjerm.outln("For å avslutte, skriv 'avslutt'");
  skjerm.out(">");
  String svar = tast.inWord("\n"); // leser inn fra skjerm til linjeskift
  while (!svar.equals("avslutt")) {
      Setning set = new Setning(svar); // lager objekt av klassen Setning
      boolean funnet = false; // initialiserer den boolske variabelen funnet
      while(set.flereOrd() && !funnet) {
        String ord = set.nesteOrd(); // leser neste ord i setningen
        if (h.containsKey(ord)) { // sjekker om ordet er en nøkkel i h
           String respons = (String) h.get(ord); // hvis nøkkel leser respons
           funnet = true;
           skjerm.outln(respons);
        }
      }
      if (!funnet) {
         skjerm.outln("Fortell mer!");
      }

      skjerm.out(">");
      svar = tast.inWord("\n");
   }
```

Klassen Eliza1 med main metoden.

```
import easyIO.*;
import java.util.*;

class Eliza1 {
   public static void main (String[] args) {
        Psykiater psy = new Psykiater();
        psy.lesFraFil();
        psy.samtale();
   }
}
```



Programeksempler på nettsiden

- På kursets hjemmeside finner du et større programeksempel som illustrerer bruk av HashMap