Repetisjon – Lub. samarbeid mellom obj.

- kan beskrive samarbeid mellom objekter ved hjelp av modelleringsspråket UML (sekvensdiagram)
- kan redegjøre for hvordan objekter kan samarbeide
- kan redegjøre for forskjellen mellom komposisjon og aggregering og avgjøre i hvilke tilfeller disse to teknikkene skal brukes
- kan beskrive komposisjon og aggregering ved hjelp av modelleringsspråket UML (klassediagram)
- kan forklare hvorfor flere referanser til samme mutable objekt kan være problematisk
- kan utvikle og anvende equals- og compareTo-metoder for å sammenligne objekter

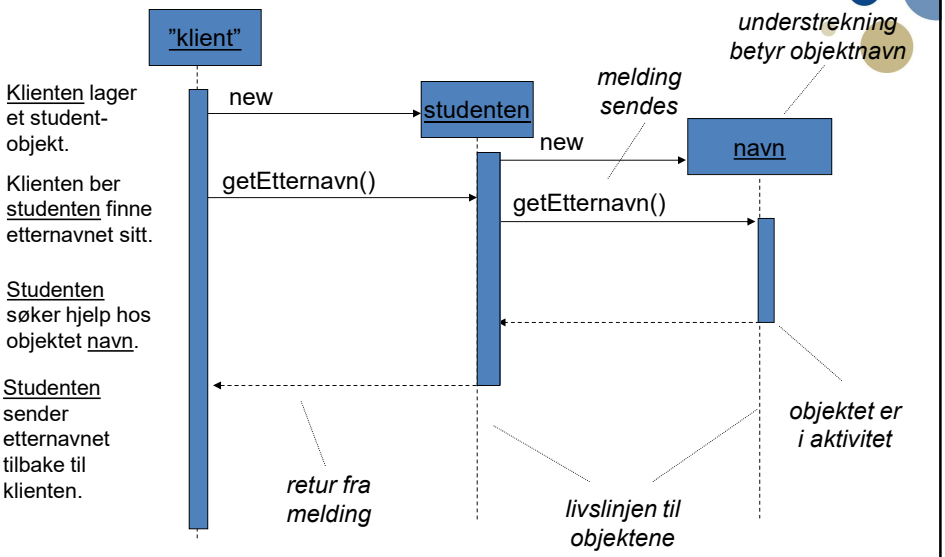
1

Repetisjon - Samarbeid mellom objekter

- Objekter samarbeider ved å sende meldinger
- Et objekt er bygget opp av mindre deler
  - Løser oppgaver ved å samarbeide med de delene det består av
- Komposisjon
  - Student-objektet er det eneste objektet som har tilgang til navne-objektet
- Aggregering
  - Kan ha flere referanser til det samme mutable objektet og dermed kan datainnholdet endres fra mange forskjellige steder
- Sekvensdiagram
- Standardiserte metodehoder for sammenligning
  - equals()
  - compareTo()

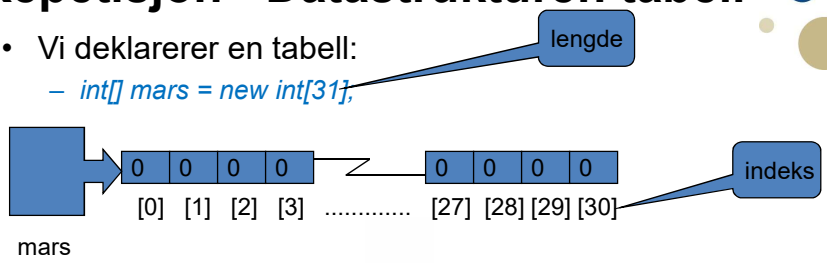
2

Repetisjon - Sekvensdiagram

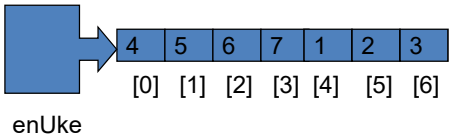


Repetisjon - Datastrukturen tabell

- Vi deklarerer en tabell:
  - *int[] mars = new int[31];*



- Kan deklarer og initiere i én og samme prosess:
  - *int[] enUke = {4, 5, 6, 7, 1, 2, 3}; // lengden blir 7*

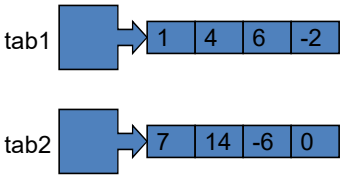


4

Repetisjon - Må kopiere element for element

Før kopiering:

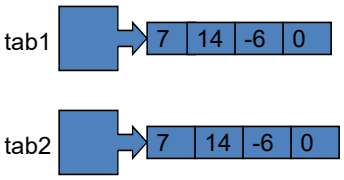
```
int[] tab1 = {1, 4, 6, -2};
int[] tab2 = {7, 14, -6, 0};
```



Kopierer element for element:

```
for (int i = 0; i < tab1.length; i++) {
  tab1[i] = tab2[i];
}
```

Etter kopiering:



5

Repetisjon - Tabell med dynamisk lengde

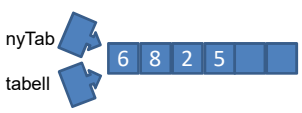
- Lag store tabeller og hold orden på hvor mye som er fylt inn
- Hvis tabellen blir full kan vi utvide den

En tabell

Oppretter ny tabell

Kopierer data

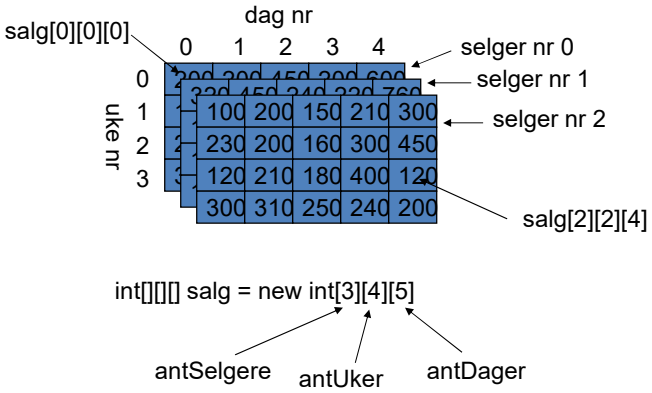
Setter tabell lik nyTab



6

Repetisjon - Flere dimensjoner

- Skal registrere salgsdata pr selger:




```
int[][][] salg = new int[3][4][5]
```

antSelgere antUker antDager

7

TDAT
Programmering Grunnkurs
Tabeller av objekter

Kunnskap for en bedre verden

Tabeller av objekter
Tabeller av objekter som medlem av klasse
Kopiering av objekter og tabeller av objekter
Å sammenligne objekter
Sortering av objekter
Biblioteksmetoder for søking / sortering

Agenda

Repetisjon fra forrige uke , agenda

Læringsutbytter kapittel 12

Lysark – tema: Tabell av objekter, String tabell m/ kodeeksempel

Oppgave 1 + 2 s 394

09.00 PAUSE

Lysark – tema: Tabell av objekter som medlem av klasse, m/ kodeeksempel

Oppgave 1 s 401

10.00 PAUSE

Kopiering av objekter og tabeller av objekter

Sortering av objekter m/ kodeeksempel

Sammenligne objekter, comparable/ comparator m/ kodeeksempler

9

Læringsutbytter, forelesning 15 – Tabeller av objekter

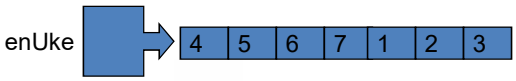
- Kunne forklare hvordan en tabell av objekter er bygd opp
- Kunne programmere aggregering ved å la en tabell av objekter være medlem i en klasse
- Kunne forklare hva som skjer når en tabell av objekter kopieres
- Kunne bruke Comparable og Comparator for å programmere sammenlikning av objekter og i neste omgang sortering av objekter
- Kunne sortere tekst riktig i henhold til norsk tegnsett ved bruk av en kollator
- Kunne bruke metoder fra Java-API'et til binærsøk og sortering

10

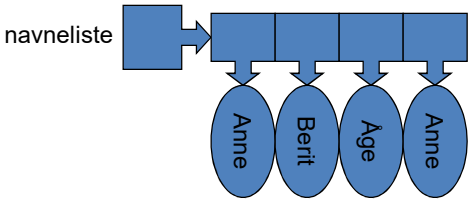
Tabell av objekter

- Vi må være ekstra oppmerksomme når vi programmerer tabeller av objekter
- Å kombinere tabeller og objekter er svært mye brukt

```
int enUke = {4,5,6,7,1,2,3};
```



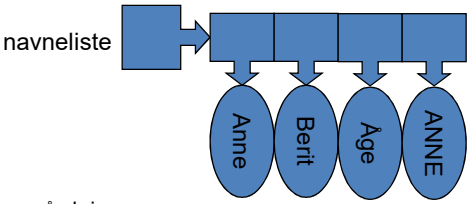

```
String[] navneliste = new String[4];
navneliste[0] = ”Anne”; navneliste[1] = ”Berit”; osv
```



11

Eksempel: En String-tabell

- Hvert enkelt element i tabellen består av en referanse til String:
  - String[] navneliste = new String[4];
- Dette er en tabell av referanser. Hver enkelt av disse referansene må vi sette til å peke til objekter av klassen String:
  - navneliste[0] = new String(”Anne”);
  - navneliste[1] = ”Berit”; // kortform går bra
- Objektet kan også være returverdi fra en metode som lager et String-objekt:
  - navneliste[2] = JOptionPane.showInputDialog("Skriv et navn: ");
  - navneliste[3] = navneliste[0].toUpperCase();



- Kan også skrive:
  - String[] navneliste = {”Anne”, ”Berit”, ”Åge”, ”ANNE”};

12

Programliste 12.1

```
/* TabellAvNavn.java */
import static javax.swing.JOptionPane.*;
class TabellAvNavn {
        public static void main(String[] args) {
                String[] navnene = new String[10];
                int antNavn = 0;
                String navn = showInputDialog("Oppgi navn: ");
                while (antNavn < navnene.length && navn != null) {
                        navnene[antNavn] = navn;
                        antNavn++;
                        navn = showInputDialog("Oppgi navn: ");
                }
                if (antNavn == navnene.length && navn != null) {
                        showMessageDialog(null, "Ikke plass til flere navn.");
                }
                String liste = "Her er navnene:\n";
                for (int i = 0; i < antNavn; i++) {
                        liste += navnene[i] + "\n";
                }
                showMessageDialog(null, liste);
        }
}
```

Gjør oppgavene 1-2 side 394.

13

Oppgaver (s 394)

1. Finn feil i følgende kodebit:

```
Vare[] varene = new Vare[3];
varene[1].settPris(320.50);
varene[2].settPris(123.70);
varene[3].settPris(120.65);
```

2. Anta at tabellen navneliste er som vist på figur 12.1 s 391. I tillegg har vi variabelen etNavn:

   `String etNavn = «Marit»;`

   Sett opp setninger som gjør følgende:

   a) Setter navneliste[1] til å referere til et nytt objekt med teksten «Anders»
   b) Setter navneliste[3] til å referere til det samme som etNavn refererer til
   c) Lagrer summen av lengdene til alle fire strengene i variabelen sumLengde. Bruk en for-setning
   d) Finner antall 'r'-er i alle strengene(hint: bruk indexOf() og en for-setning)

14

Løsning oppgave 1 s 394

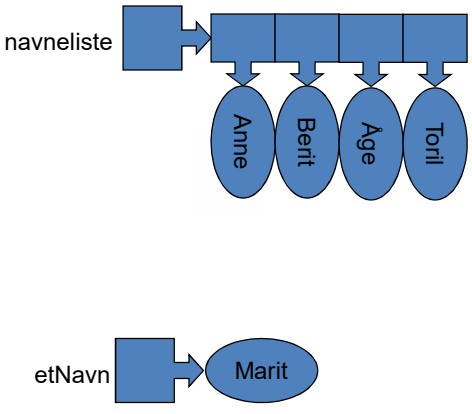
Oppgaven inneholder en utilsiktet feil. Metodenavnet settPris() er brukt. Det skal være setPris(). Etter at dette er rettet, vil kodebiten ved kjøring kaste to typer unntak:

• NullPointerException: Tabellen varene er en tabell av referanser. Disse referansene må settes til å peke til objekter, før vi kan sende meldinger til objektene.
Med andre ord: Vi må skrive for eksempel:
varene[0] = new Vare("TV-dress", 100, 575.50);
før vi kan sende melding til dette objektet:
varene[1].setPris(320.50);

• ArrayIndexOutOfBoundException: Dette unntaket kastes når vi i siste setning refererer til tabellelement med indeks 3. Elementene i en tabell med størrelse 3 nummereres 0, 1 og 2.

15

Løsning oppgave 2 s 394



16

Løsning oppgave 2 s 394

Oppgave 2a
navneliste[1] = new String("Anders"); eller navneliste[1] = "Anders";

Oppgave 2b
navneliste[3] = etNavn;

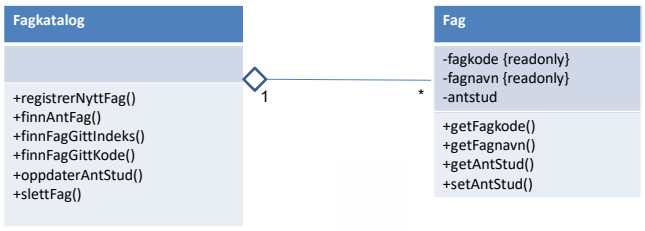
Oppgave 2c
```
int sumLengde = 0;
for (int i = 0; i < navneliste.length; i++) {
  sumLengde += navneliste[i].length();
}
```

Oppgave 2d
```
final char TEGN = 'r';
int antTegn = 0;
for (int i = 0; i < navneliste.length; i++) {
        int indeks = navneliste[i].indexOf(TEGN);
        while (indeks >= 0) {
                antTegn++;
                indeks = navneliste[i].indexOf(TEGN, indeks + 1);
        }
}
System.out.println("Antall forekomster av " + TEGN + " er: " + antTegn);
```

17

Tabell av objekter som medlem i klasse

Vi skal lage en fagkatalog som består av flere fag.



Vi skal se nærmere på klassen TabellAvFag

Filen inneholder tre klasser:

- Fag: Beskriver et fag med fagkode, navn og antall studiepoeng, tilbyr finn-metoder.
- Fagkatalog: Klassen inneholder en tabell av fagobjekter. Størrelsen på tabellen settes i konstruktøren. Objektene opprettes etter hvert som nye fag legges inn. Klienten kan sortere fagene og hente ut en oversikt ved å bruke toString(). Klienten kan også hente ut en referanse til hvert enkelt fagobjekt.
- TabellAvFag: Testklient som leser inn data om fagene fra brukeren.

Gjør oppgave 1 side 401

18

Oppgave (s 401)

1. Utvid klassen Fagkatalog med følgende to metoder:
   a) Finn totalt antall studenter i alle fagene til sammen
   b) Finn ut hvilket fag, eller hvilke, dersom det er flere, som har flest studenter.

Lag en egen klient for å prøve ut metodene.

Kode lastes ned fra: javabok.no

19

Løsning oppgave 1 s 401

```
/* Oppgave 1a */
public int finnTotAntStudenter() {
  int sum = 0;
  for (int i = 0; i < antFag; i++) {
    sum += fagene[i].getAntStud();
  }
  return sum;
}

/* Oppgave 1b */
private int finnMaksAntStud() {
   if (antFag > 0) {
      int maks = fagene[0].getAntStud();
      for (int i = 1; i < antFag; i++) {
         if (fagene[i].getAntStud() > maks) {
            maks = fagene[i].getAntStud();
         }
      }
      return maks;
   }
   return 0; // ingen fag registrert
}
```

```
public Fag[] finnStørsteFag() {
   int maks = finnMaksAntStud();
   // hjelpemetode, se nedenfor
   Fag[] fagMedMaks = new Fag[antFag];
   int antFagLikMaks = 0;

   for (int i = 0; i < antFag; i++) {
      if (fagene[i].getAntStud() == maks) {
         fagMedMaks[antFagLikMaks] =
       fagene[i];
         antFagLikMaks++;
      }
   }
   Fag[] nyTab = new Fag[antFagLikMaks];
   for (int i = 0; i < antFagLikMaks; i++) {
       nyTab[i] = fagMedMaks[i];
    }
    return nyTab;
}
```

20

Løsning oppgave 1 s 401

Testklient:

```
public static void main(String[] args) {
System.out.println("Totalt antall tester: 4");

/* Tom katalog */
Fagkatalog kat0 = new Fagkatalog();
Fag[] fag0 = kat0.finnStørsteFag();
if (fag0.length == 0) {
     System.out.println("Test 1 vellykket");
}
if (kat0.finnTotAntStudenter() == 0) {
     System.out.println("Test 2 vellykket");
}
/* Katalog med 5 fag */
Fagkatalog kat = new Fagkatalog();
kat.registrerNyttFag("LC191D", "Videregående prog»);
kat.registrerNyttFag("LV172D", "Programmering i Java");
kat.registrerNyttFag("LO347D", "Web-applikasjoner");
kat.registrerNyttFag("LO346D", "Java EE");
kat.registrerNyttFag("LC331D", "IT, miljø og samfunn");
```

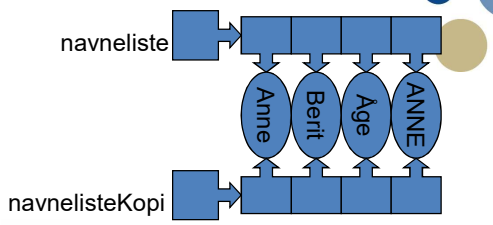
```
kat.oppdaterAntStud("LC191D", 20);
kat.oppdaterAntStud("LV172D", 30);
kat.oppdaterAntStud("LO347D", 20);
kat.oppdaterAntStud("LO346D", 30);
kat.oppdaterAntStud("LC331D", 30);

Fag[] fag = kat.finnStørsteFag();

if (fag.length == 3 &&
     fag[0].getFagkode().equals("LV172D")
&& fag[1].getFagkode().equals("LO346D")
&& fag[2].getFagkode().equals("LC331D"))
{
      System.out.println("Test 3 vellykket");
}
if (kat.finnTotAntStudenter() == 130) {
      System.out.println("Test 4 vellykket");
}
} // main
```

21

Kopiering av objekter og tabeller av objekter

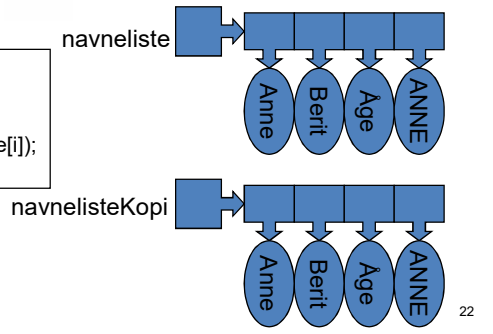
Grunn kopiering

```
String[] navnelisteKopi = new String[4];
for (int i = 0; i < navneliste.length; i++) {
  navnelisteKopi[i] = navneliste[i];
}
```



Dyp kopiering

```
String[] navnelisteKopi = new String[4];
for (int i = 0; i < navneliste.length; i++) {
  navnelisteKopi[i] = new String(navneliste[i]);
}
```



22

Forskjeller mellom tabeller av prim. datatyper og tabeller av ref.typer

- En tabell av en primitiv datatype:
  - Elementene i tabellen inneholder dataverdiene.
  - Dataverdiene kopieres dersom tabellen kopieres element for element.
  - Elementene kan sammenlignes ved å bruke sammenligningsoperatorene.
  - Elementene initieres til 0 (ev. false) dersom ikke andre verdier gis i deklareringen av tabellen.
- En tabell av en referansetype:
  - Elementene i tabellen inneholder ikke objektene, men referanser til objektene.
  - Dersom tabellen kopieres element for element, blir bare referansene kopiert, ikke objektene. Element med samme indeks i begge tabellene peker til det samme objektet.
  - Med unntak av lik (==) og ikke lik (!=) kan vi ikke bruke sammenligningsoperatorer på referanser. Operatorene lik og ikke lik sammenligner innholdet i referansene, ikke i objektene som referansene peker til.
  - Elementene initieres til null, dersom ikke andre verdier gis i deklareringen av tabellen. Dersom vi prøver å bruke et tabellelement som ikke refererer til noe objekt, kastes NullPointerException.

23

Sortering av objekter

- Vi sorterer objekter ved å bruke compareTo() – metoden og sortering ved utvelgelse
- Husk å velge hvilken egenskap vi skal sortere på
- Slik det er programmert i eksemplet vil du få advarsler fra kompilatoren. Disse kan du ignorere

Se på klassen Sortering.java => GENERELL KODE!!

Se på klassen SorteringAvFlater.java

Å sammenligne objekter - comparable

public interface **Comparable<Type>**

- Dette interfacet tilbyr sortering av objektene til alle klasser som implementerer denne. Den sorterer etter klassens naturlige orden og klassens compareTo-metode refereres til som den naturlige sammenligningsmetoden.
- Objekter som implementerer dette interfacet kan brukes som elementer i et sortert sett uten at vi trenger å spesifisere en comparator.
- Den naturlige ordenen for en klasse skal være konsistent med *equals* hvis og bare hvis e1.compareTo(e2) == 0 har den samme boolske verdien som e1.equals(e2) for alle e1 and e2 som hører til klassen.

Å sammenligne objekter - comparator

public abstract class Collator extends Object implements Comparator<Object>, Cloneable

- Collator klassen tilbyr lokaliseringssensitiv sammenligning av tekststrenger. Du bruker denne klassen til å bygge søke- og sorteringsrutiner for tekst.
- Du kan også bruke getInstance for å hente ut det passende Collator-objektet for en gitt lokalisering.

Følgende eksempel viser hvordan du kan sammenligne to tekststrenger ved å bruke Collator for default lokalisering.

```
// Sammenlign to strenger ved hjelp av Collator myCollator = Collator.getInstance();
if( myCollator.compare("abc", "ABC") < 0 )
        System.out.println("abc is less than ABC");
else
        System.out.println("abc is greater than or equal to ABC");
```

Å sammenligne objekter - comparable

- Se på klassen Flate.java
  - ✓ implements Comparable<Flate>
  - ✓ Arrays.sort(flater);
- Se på klassen FlateKompAreal.java
  - ✓ implements Comparator<Flate>
  - ✓ Arrays.sort<flater, new FlateKompAreal());

Å sammenligne objekter - comparator

Se på klassen Flate.java

String[]:
Naurlig orden: Arrays.sort(stringTab);
Alternativt: Arrays.sort(stringTab, Collection.reversorder());