Uke 8 - repetisjon

agneshb@uio.no

I dag

Hovedtemaer hittil:

- Arv og subklasser
- Beholdere
- Rekursjon

Andre tema

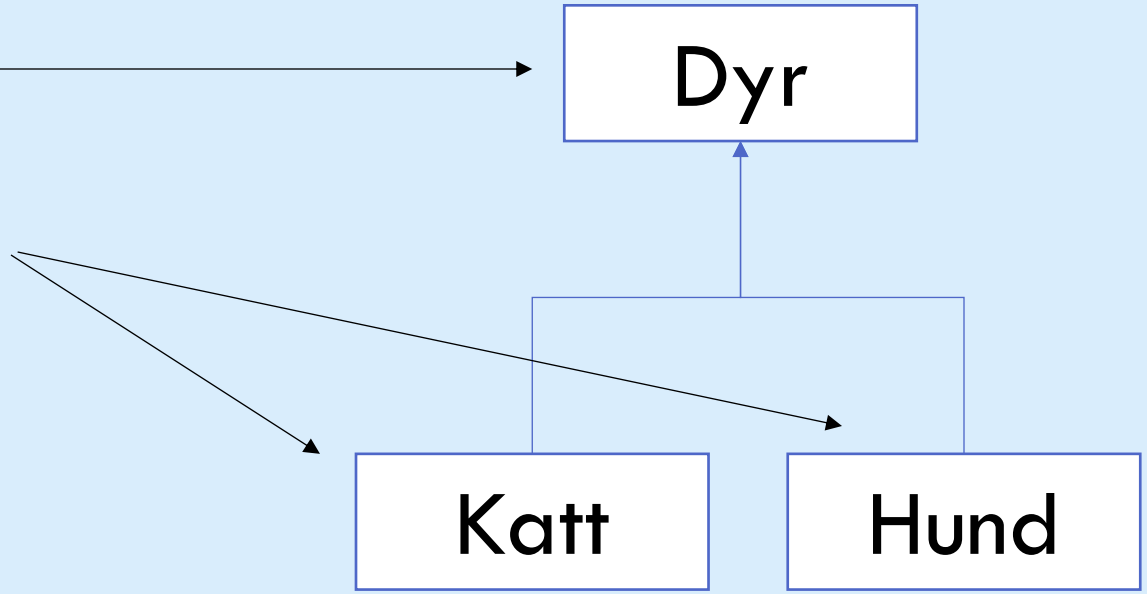
- Lese og skrive til fil, scanner
- Exceptions
- Enhetstesting
- Datastrukturtegning

- Quiz

Arv og subklasser

Arv og subklasser 1

- **Superklasse** har et sett egenskaper
- En **subklasse** er en klasse som **arver** egenskaper fra superklassen
- Extends: Subklassen er en utvidelse av superklassen.



Eks: Class Katt extends Dyr

Arv og subklasser del 1

- Subklassen(e) må ha de samme instansvariablene som superklassen
  - Instansierer de i konstruktøren med super(..)
- Subklassen(e) arver metodene til superklasser
  - Altså: kan bruke metodene uten å implementere de i subklassene

```java
public abstract class Dyr {

    String navn;

    public Dyr(String navn){
        this.navn = navn;
    }
```

```java
public class Katt extends Dyr{

    String favorittmat;

    public Katt(String navn, String favorittmat) {
        super(navn);
        this.favorittmat = favorittmat;
    }
```

Arv og subklasser del 2 - Polymorfi

- Hvis man ønsker samme funksjonalitet i subklassen som superklassen, gjør man ingenting.
- Hvis man ønsker annen funksjonalitet: @Override
- Samme metodesignatur
- Ulikt innhold/funksjonalitet

```
public abstract class Dyr {

    String navn;

    public Dyr(String navn){…

    public String spis(){
        return "jeg spiser";
    }
```

```
public class Katt extends Dyr{

    String favorittmat;

    public Katt(String navn, String favorittmat) {…

    @Override
    public String spis() {
        return super.spis() +" "+ favorittmat;
    }
```

Arv og subklasser del 2 – arv og komposisjon

- Arv:
  - Utvidelse av en klasse
  - Er-en-relasjoner
  - En katt er et dyr

- Komposisjon
  - Referanse til objekter av andre klasser.
  - Har-en/består-av-relasjoner
  - En katt har en favorittmat

- Var spørsmål som dette på prøveeksamen

Arv og subklasser del 3 - Interface

- Kan ikke arve fra flere klasser
- Kan implementere flere interface
- Felles egenskaper, uten å arve
  - et fly, et helikopter, en fugl og en sommerfugl kan fly.

Arv og subklasser del 3 - Interface

- Ingen variabler
- Kun abstrakte metoder
  - Må selv implementere metoder i klassen som implementerer interfacet.
  - Bruker @Override

```
public interface KanLope {

    public int maksFart();
```

```
public class Katt extends Dyr implements KanLope{
```

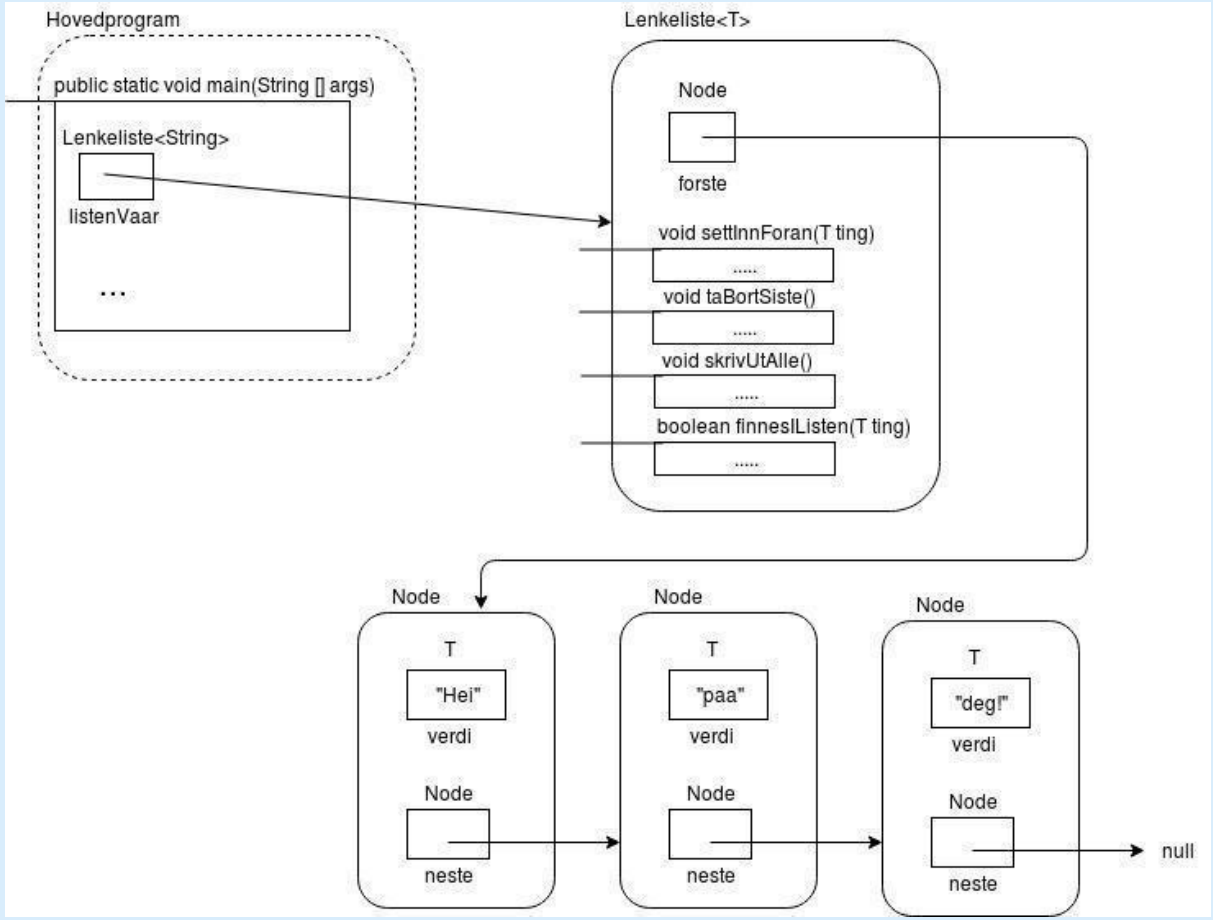
...

```
@Override
public int maksFart() {
    return 40;
}
```

Beholdere

Lenkelister

- Noder som inneholder data er koblet sammen i en lenkestruktur i en lenkeliste.
- Lenkelisten vil håndtere innsetting, sletting og andre operasjoner.
- Referanse til første (evt. siste) node i lenkelisten.
- Lenkelisten kan være generisk
- Enkeltlenket eller dobbeltlenket



Lenkelister - Ulike typer dere bør kjenne til

- **Stabel/stack (LIFO)** - elementer settes inn og tas ut foran
- **Kø (FIFO)** - elementer settes inn bakerst og tas ut foran
- **Prioritetskø** - elementer settes inn og tas ut etter ønsket prioritet

Interfacet Comparable

- For å sammenligne objekter
- compareTo-metoden må implementeres i klasser som implementerer Comparable
- Leksikografisk
- **positivt heltall** når dette objektet er større enn objektet vi sammenligner med.
- **negativt heltall** når dette objektet er mindre enn objektet vi sammenligner med.
- 0 når objektene vi sammenligner er like

```java
public class Person implements Comparable<Person>{

    int alder;

    public Person(int alder){
        this.alder = alder;
    }
    @Override
    public int compareTo(Person annenPerson) {
        return this.alder - annenPerson.alder;
    }
}
```

Interfacet Iterable

- For å iterere over elementer i lister og andre beholdere med en for each-løkke
- hashNext() – sjekker om det er et til element å iterere over.
- next() – går til neste element.
- hasNext-metoden og next-metoden kan tilpasses nøyaktig slik vi ønsker å ha dem

Iterable implementasjon

1. Klassen må implementere Iterable<T>.
2. Lager en indre iteratorklasse som implementerer Iterator<T>.
3. Implementere metoden iterator() som returnerer en instans av iteratorklassen fra steg 2.

```java
public class Koe<T> implements Iterable<T>{

    @Override
    public Iterator<T> iterator() {
        return new KoeIterator();
    }

    class KoeIterator implements Iterator<T> {

        Node denne = start;

        @Override
        public boolean hasNext() {
            return denne != null;
        }


        @Override
        public T next() {
            T data = denne.data;
            denne = denne.neste;
            return data;
        }
    }
```

Rekursjon

Rekursjon

- **Basistilfelle** (“base case”)
  - Rekursjonen stopper
  - Kallene «nøstes opp» fra siste til første kall.
- **Rekursjonssteg** (“recursive step”)
  - Endrer noe slik at man til slutt når basistilfellet.

- Merk at rekkefølgen på det rekursive kallet og printsetningen påvirker rekkefølgen

```java
public static void countDownRec(int n){
    if (n < 0) return;
    System.out.println(n);
    countDownRec(n-1);
}
```

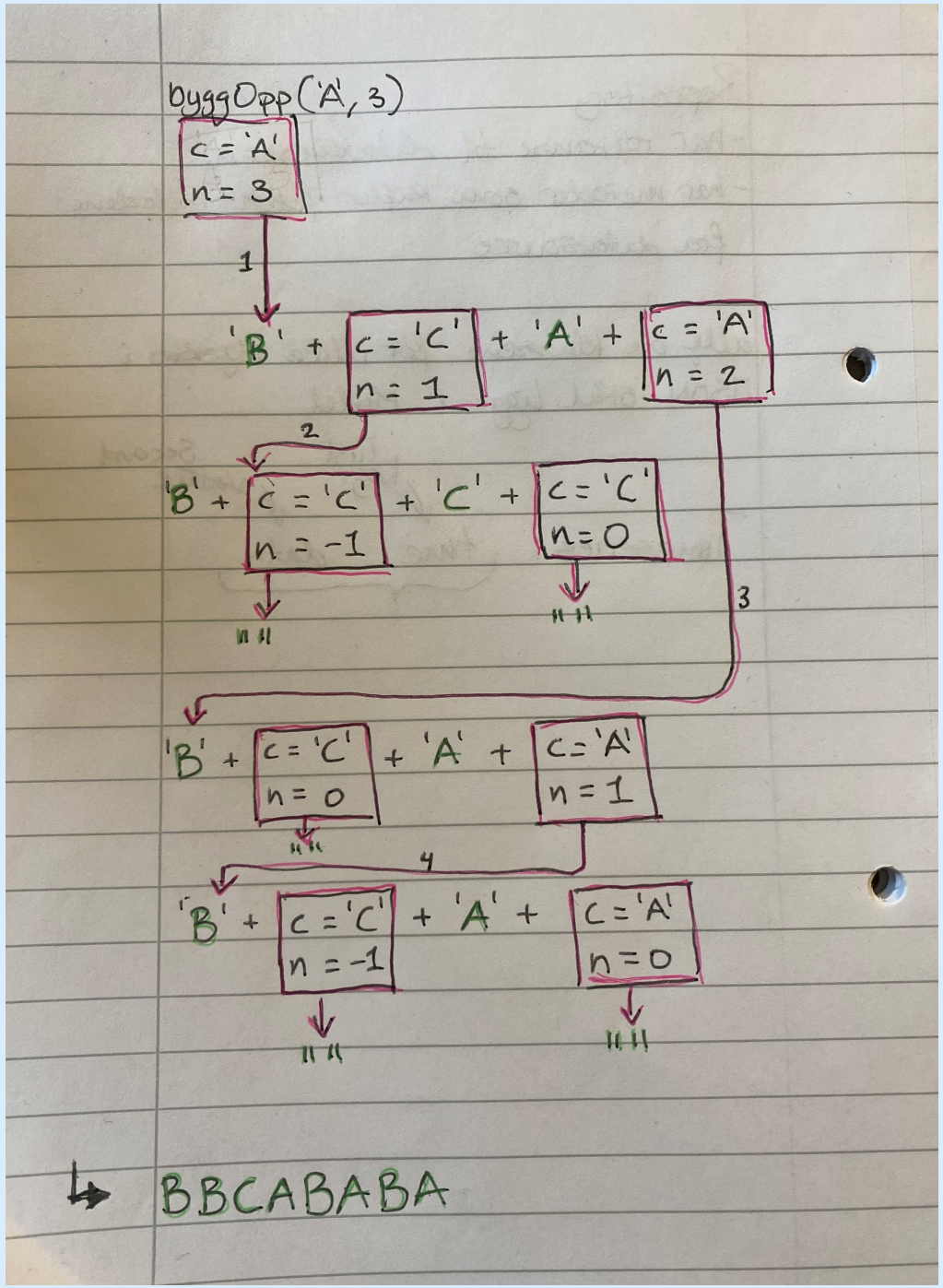
Hva printes her? (vanskelig)

```java
public class ByggOpp {

    public static String byggOpp (char c, int n) {
        if (n <= 0) {
            return "";
        } else {
            return 'B' + byggOpp(c:'C', n-2) + c + byggOpp(c, n-1);
        }
    }

    Run | Debug
    public static void main(String[] args) {
        String res = byggOpp (c:'A', n:3) ;
        System.out.println(res);
    }
}
```

Tegning fra gruppetimen



Scanner

- Scanner kan brukes for å ta imot input fra forskjellige kilder:
  - Brukerinput i terminalen
  - Lesing fra og skriving til fil.

```
Scanner sc = new Scanner(System.in);

System.out.println(x:"Hva heter du?");
String navn = sc.nextLine();
System.out.println("Hei " + navn + "!");
```

- Scanner har mange metoder, blant annet:
  - hasNextLine() – sjekker om det er en til linje.
  - nextLine() – går til neste linje

Lesing fra fil med scanner

- Oppretter en Scanner
- Bruker **hasNextLine**() for å sjekke at det er en til linje
- før man forsøker å gå til neste linje med scanner sin **nextLine**().

- Må bruke try-catch eller throws for å håndtere tilfellet der programmet ikke finner filen.

```
public static void lesFraFil(File fil){
    try (Scanner scanner = new Scanner(fil)) {

        while (scanner.hasNextLine()){
            System.out.println(scanner.nextLine());
        }

    } catch (FileNotFoundException e) {
        System.out.println(x:"fant ikke filen!");
    }
}
```

Exceptions/unntak

- Når en feil skjer utløses et unntak
- Mange innebygde unntak i Java
- Kan være nyttig å opprette egne exception-klasser for å gi mer spesifikk informasjon om en feil.
- Dette kan bidra til å gjøre feilsøking og feilhåndtering mer effektiv.

Checked og unchecked exceptions

- Unchecked:
  - RuntimeException
  - Må ikke håndteres i koden
  - NullpoinerException, IndexOutOfBoundsException
- Checked:
  - Exception
  - Må håndteres eller «kastes videre» til kallsted.
  - Try-catch eller throws
  - FileNotFoundException, IOException

Eksempel på unchecked

```java
public class ExceptionsEksempel {

    public static void settInn(String[] array, int pos){
        if (pos < 0 || pos >= array.length){
            throw new UgyldigIndexException(pos, array.length);
        }
    }
```

```java
public class UgyldigIndexException extends RuntimeException{
    public UgyldigIndexException(int pos, int maks){
        super("Indeks "+ pos +"er en ugyldig indeks! \n Laveste gyldig indeks er 0 og høyeste gyldige indeks er "+ maks);
    }
}
```

Testing

- **Enhetstesting:** test hver del eller hvert objekt for seg
  - Eks innlevering 1 – egne tester for å teste resepter og legemiddel, tester metoder/deler av metoder hver for seg
- **Integrasjonstesting:** Etter å ha testet hver del kan man teste det sammen
  - Eks innlevering 1 (del D) – brukte alle delene av programmet for å se at de fungerte sammen.



Testing forts.

- Prøv flere ulike rekkefølger på kall.
- Spesielt viktig å sjekke **edge-cases:**
  - Hva skjer hvis man prøver å hente ut noe fra en tom liste?
  - Hva skjer hvis en variabel har verdi null?
  - Hva skjer hvis man prøver å sette inn noe i et fullt array?

- Kan bruke testklasser fra innleveringer som inspirasjon.

Datastrukturtegning

Nederst på dette notatet:

https://www.uio.no/studier/emner/matnat/ifi/IN1010/v23/Notater/objekter-klasser-in1010-2023-jan-13.pdf

Jobbe med oppgaver<3

- Øv på å tolke andres kode
- Innleveringer, trix, ukesoppgaver
- Notater: https://www.uio.no/studier/emner/matnat/ifi/IN1010/v23/Notater/