# IN1010 - Seminar 5

- Interface

Laget av: Marlen Jarholt ( marleja@ifi.uio.no ) og Johanna Haarseth ( johannph@ifi.uio.no )

# Praktisk

Oblig 2 har frist 15. februar kl 23.59! (PS: Hvis du vil ha tilbakemelding på obligen må du eksplisitt skrive det i devilry)

Husk å sjekke emnesiden regelmessig

Undervisningstilbud:

- https://www.uio.no/studier/emner/matnat/ifi/IN1010/v21/undervisningstilbud/
- Jeg har konkrete spørsmål/problemer med min kode -> Labtime!
- Jeg vil ha mer liveprogrammering -> Plenumstime!
- Jeg vil jobbe med andre (og kanskje en kjapp recap av forelesning) -> Gruppetime!
- Jeg vil ha en recap av de vanskeligste konseptene fra forelesning -> Repetisjonsgruppe!

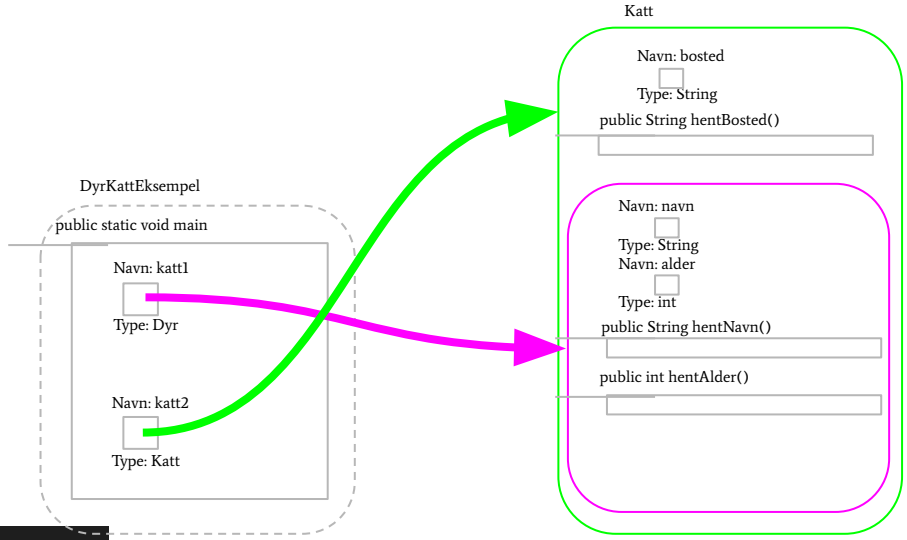
# Repetisjon forrige uke

# Casting

```java
class Katt extends Dyr{

    private String bosted;

    public Katt(String navn, int alder, String bosted){
        super(navn, alder);
        this.bosted = bosted;
    }

    public String hentBosted(){
        return bosted;
    }
}
```

```java
abstract class Dyr {

    protected String navn;
    protected int alder;

    public Dyr(String navn, int alder){
        this.navn = navn;
        this.alder = alder;
    }

    public String hentNavn(){
        return navn;
    }

    public int hentAlder(){
        return alder;
    }
}
```

```java
class DyrKattEksempel{

    public static void main(String[] args) {

        Dyr katt1 = new Katt("Pus", 1, "Oslo");

        System.out.println(katt1.hentNavn())
        System.out.println(katt1.hentAlder())
        //System.out.println(katt1.hentBosted())

        Katt katt2 = (Katt) katt1;
        System.out.println(katt2.hentBosted());
        System.out.println(katt2.hentNavn());

    }
}
```




Når vi nå ser på katte objektet men katt2 sine briller får vi også alle egenskapene til en Katt. (Det som er markert i grønt)

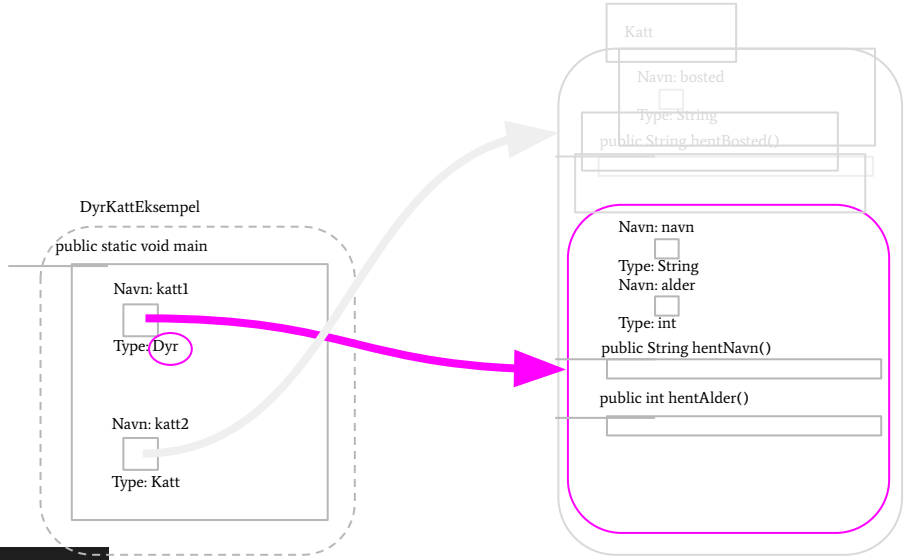
PS: katt1 kan fortsatt bare se dyreegenskapene (Det som er markert i rosa)

# Casting

```java
class Katt extends Dyr{

    private String bosted;

    public Katt(String navn, int alder, String bosted){
        super(navn, alder);
        this.bosted = bosted;
    }

    public String hentBosted(){
        return bosted;
    }
}
```

```java
abstract class Dyr {

    protected String navn;
    protected int alder;

    public Dyr(String navn, int alder){
        this.navn = navn;
        this.alder = alder;
    }

    public String hentNavn(){
        return navn;
    }

    public int hentAlder(){
        return alder;
    }
}
```

```java
class DyrKattEksempel{

    public static void main(String[] args) {

        Dyr katt1 = new Katt("Pus", 1, "Oslo");

        System.out.println(katt1.hentNavn())
        System.out.println(katt1.hentAlder())
        //System.out.println(katt1.hentBosted())

        Katt katt2 = (Katt) katt1;
        System.out.println(katt2.hentBosted());
        System.out.println(katt2.hentNavn());

    }
}
```




Når vi nå ser på katte objektet men katt2 sine briller får vi også alle egenskapene til en Katt. (Det som er markert i grønt)

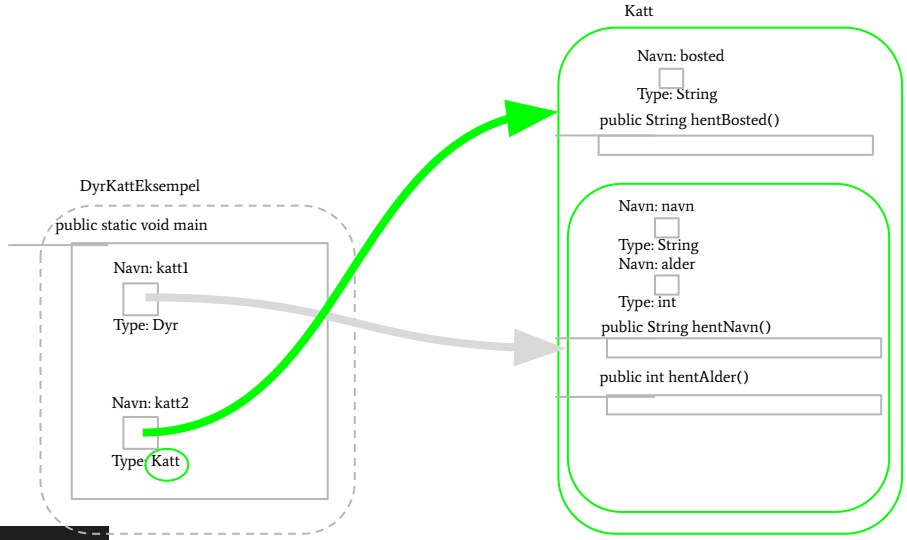
PS: katt1 kan fortsatt bare se dyreegenskapene (Det som er markert i rosa)

# Casting

```java
class Katt extends Dyr{

    private String bosted;

    public Katt(String navn, int alder, String bosted){
        super(navn, alder);
        this.bosted = bosted;
    }

    public String hentBosted(){
        return bosted;
    }
}
```

```java
abstract class Dyr {

    protected String navn;
    protected int alder;

    public Dyr(String navn, int alder){
        this.navn = navn;
        this.alder = alder;
    }

    public String hentNavn(){
        return navn;
    }

    public int hentAlder(){
        return alder;
    }
}
```

```java
class DyrKattEksempel{

    public static void main(String[] args) {

        Dyr katt1 = new Katt("Pus", 1, "Oslo");

        System.out.println(katt1.hentNavn())
        System.out.println(katt1.hentAlder())
        //System.out.println(katt1.hentBosted())

        Katt katt2 = (Katt) katt1;
        System.out.println(katt2.hentBosted());
        System.out.println(katt2.hentNavn());

    }
}
```




Når vi nå ser på katte objektet men katt2 sine briller får vi også alle egenskapene til en Katt. (Det som er markert i grønt)

PS: katt1 kan fortsatt bare se dyreegenskapene (Det som er markert i rosa)

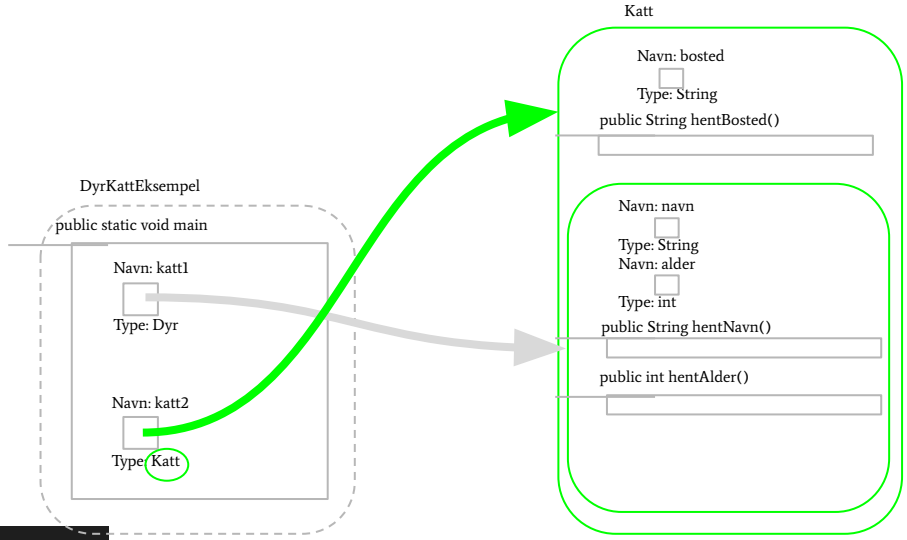
# Casting

Typen til variabelen betyr noe for hva vi kan aksessere!

```java
class Katt extends Dyr{

    private String bosted;

    public Katt(String navn, int alder, String bosted){
        super(navn, alder);
        this.bosted = bosted;
    }

    public String hentBosted(){
        return bosted;
    }
}
```

```java
abstract class Dyr {

    protected String navn;
    protected int alder;

    public Dyr(String navn, int alder){
        this.navn = navn;
        this.alder = alder;
    }

    public String hentNavn(){
        return navn;
    }

    public int hentAlder(){
        return alder;
    }
}
```

```java
class DyrKattEksempel{

    public static void main(String[] args) {

        Dyr katt1 = new Katt("Pus", 1, "Oslo");

        System.out.println(katt1.hentNavn())
        System.out.println(katt1.hentAlder())
        //System.out.println(katt1.hentBosted())

        Katt katt2 = (Katt) katt1;
        System.out.println(katt2.hentBosted());
        System.out.println(katt2.hentNavn());

    }
}
```



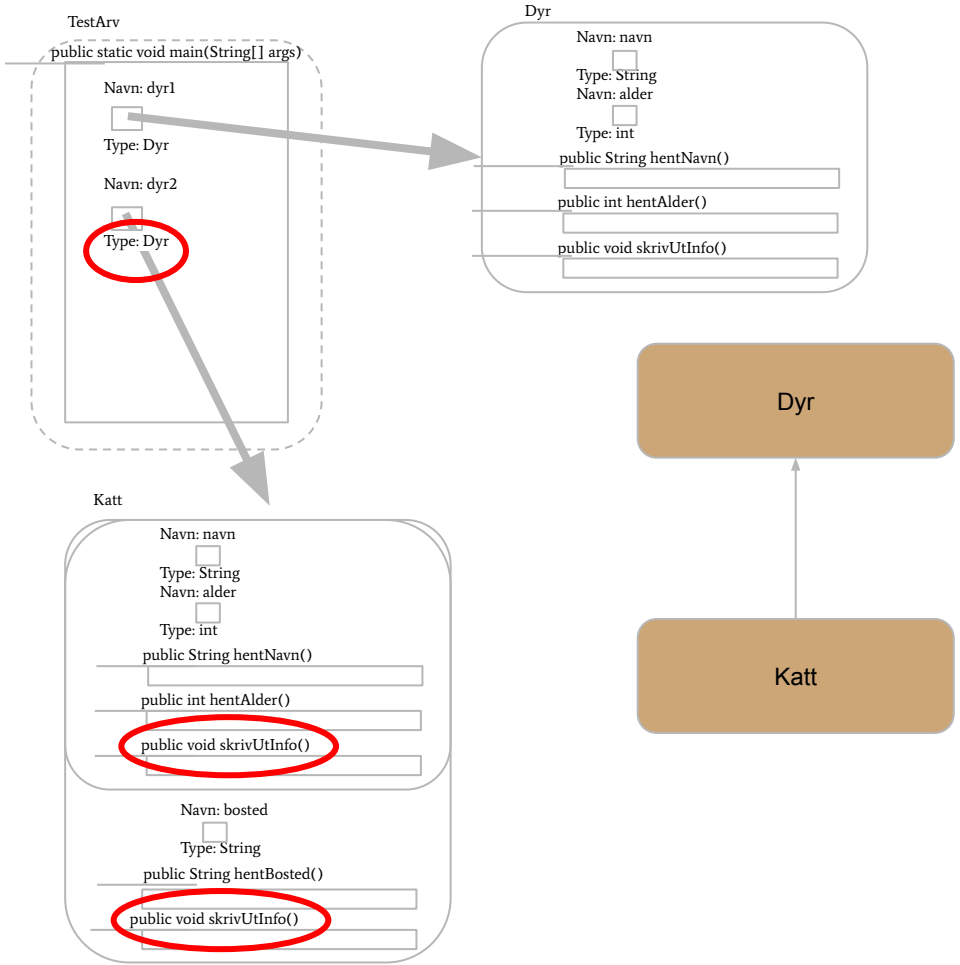

Når vi nå ser på katte objektet men katt2 sine briller får vi også alle egenskapene til en Katt. (Det som er markert i grønt)

PS: katt1 kan fortsatt bare se dyreegenskapene (Det som er markert i rosa)

# Polymorfi

```java
class Dyr{
  protected int alder;
  protected String navn;

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

  public Katt(String navn, int alder, String bostedsadresse){
    super(navn, alder);
    this.bostedsadresse = bostedsadresse;
  }
  public String hentBosted(){
    return bostedsadresse;
  }
  @Override
  public void skrivUtInfo(){
    super.skrivUtInfo();
    System.out.println(bostedsadresse);
  }
}
```

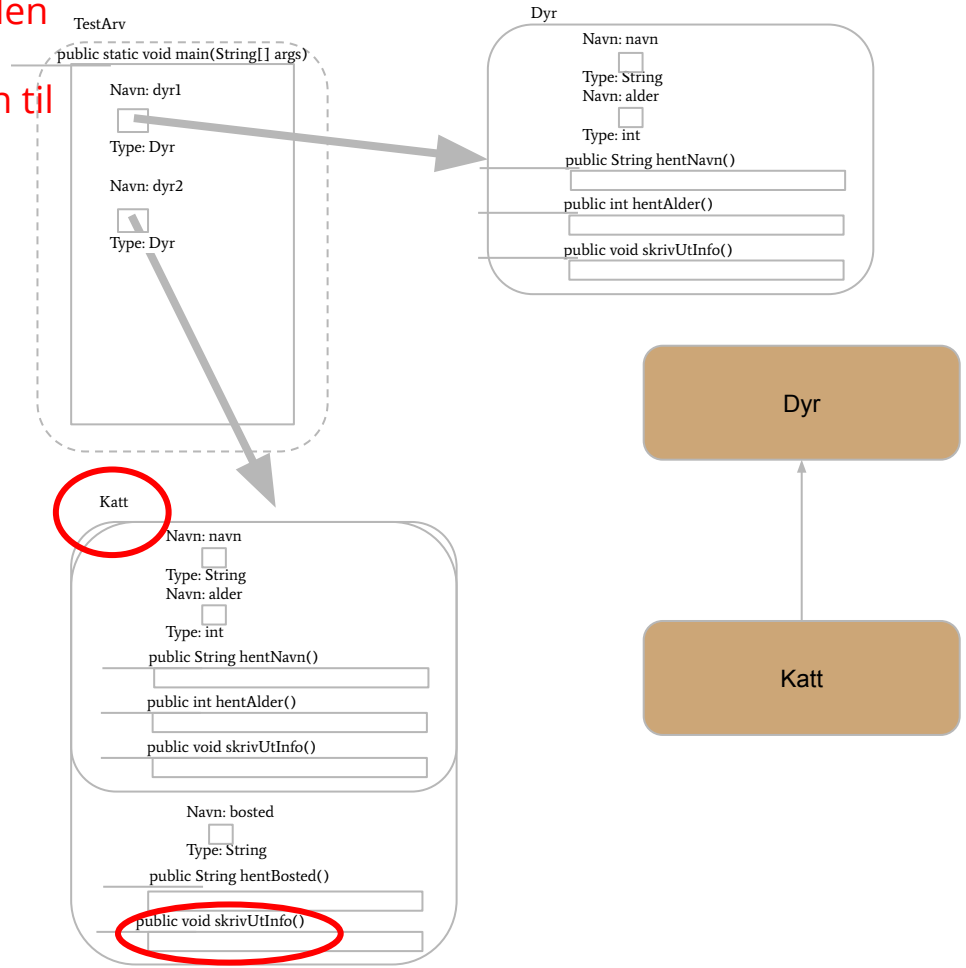


# Polymorfi

Typen til variabelen betyr ingenting, vi må se på typen til objektet

```java
class Dyr{
  protected int alder;
  protected String navn;

  public Dyr(String navn, int alder){
    this.navn = navn;
    this.alder = alder;
  }
  public  int hentAlder(){
    return alder;
  }
  public  String hentNavn(){
    return navn;
  }
  public void skrivUtInfo(){
    System.out.println("Navn: " + navn);
    System.out.println("Alder: " + alder);
  }
}
```

```java
class Katt extends Dyr{

  private String bostedsadresse;

  public Katt(String navn, int alder, String bostedsadresse){
    super(navn, alder);
    this.bostedsadresse = bostedsadresse;
  }
  public String hentBosted(){
    return bostedsadresse;
  }
  @Override
  public void skrivUtInfo(){
    super.skrivUtInfo();
    System.out.println(bostedsadresse);
  }
}
```



# Abstract - Hva er det og hvordan lage det?

- **Hva er det?** En abstract klasse er en klasse som man ikke kan opprette en instans av, men som har subklasse(r) man kan opprette instanser av. Her er det f.eks. ikke naturlig å opprette en instans av Dyr fordi det finnes ikke noe som er dyr, kun subklasser av det.
- **Hvordan lage det?**
  - Man skriver abstract helt foran i klasse definisjonen
  - Inne i en abstract klasse kan man ha en abstract metode, en abstract er en metode som klassen ikke implementer selv, men som alle ikke abstracte subklassene må implementere. Dyr lager jo ingen lyd, men alle subklassene (Katt og Fjellrev lagere en lyd)

```java
abstract class Dyr {

    protected String navn;
    protected int alder;

    public Dyr(String navn, int alder){
        this.navn = navn;
        this.alder = alder;
    }

    public String hentNavn(){
        return navn;
    }

    public int hentAlder(){
        return alder;
    }

    public abstract void lagLyd();
}
```

```java
class AbstractEksempel {

    Run | Debug
    public static void main(String[] args) {

        //Det er lov til aa lage instanser av subklassen til en abstract klasse
        Katt katt = new Katt("Pus", 1, "Oslo");
        Fjellrev fjellrev = new Fjellrev("Tom", 13);

        //Ikke lov til aa lage en instans av en abstract klasse
        Dyr dyr = new Dyr("Klara", 12);
    }

}
```

OK

X

# Abstract - Ikke abstract subklasser

En ikke abstract subklasse av en abstract klasse MÅ implementere de abstracte metodene til superklassen sin.

```java
class Katt extends Dyr{

    private String bosted;

    public Katt(String navn, int alder, String bosted){
        super(navn, alder);
        this.bosted = bosted;
    }

    public String hentBosted(){
        return bosted;
    }

    //siden lagLyd er abstract i superklassen maa vi implementer den i
    //sub klassen
    public void lagLyd(){
        System.out.println("Mjauuuuu");
    }
}
```

```java
class Fjellrev extends Dyr{

    public Fjellrev(String navn, int alder){
        super(navn, alder);
    }

    //siden lagLyd er abstract i superklassen maa vi implementer den i
    //sub klassen
    public void lagLyd(){
        System.out.println("What does the fox say?");
    }
}
```

# Repetisjon denne uken

# Interface

Vi har en klasse Dyr. I den virkelige verden finnes det en del dyr som har sånn chip så man kan sjekke hvem som eier dem og sånn.

To eksempler er katter og hunder(tror jeg??)

# Interface

```java
abstract class Dyr{
    protected int alder;
    protected String navn;

    public Dyr(String navn, int alder){
        this.navn = navn;
        this.alder = alder;
    }
    public  int hentAlder(){
        return alder;
    }
    public  String hentNavn(){
        return navn;
    }
    @Override
    public String toString(){
        return "Navn: " + navn + "\n" + "Alder: " + alder;
    }
}
```

```java
class Hund extends Dyr{

    public Hund(String navn, int alder){
        super(navn, alder);
    }
    @Override
    public String toString(){
        return "Jeg er en hund!" + super.toString();
    }
}
```

```java
class Katt extends Dyr{

    public Katt(String navn, int alder){
        super(navn, alder);
    }
    @Override
    public String toString(){
        return "Jeg er en katt!" + super.toString();
    }

}
```

# Interface

Vi har en klasse Dyr. I den virkelige verden finnes det en del dyr som har sånn chip så man kan sjekke hvem som eier dem og sånn.

To eksempler er katter og hunder(tror jeg??)

Vi skal fikse sånn at Katt og Hund blir Dyr med chip!

```
1  interface HarChip{
2      Person hentEier();
3      String hentId();
4  }
```

# Interface

Vi har en klasse Dyr. I den virkelige verden finnes det en del dyr som har sånn chip så man kan sjekke hvem som eier dem og sånn.

To eksempler er katter og hunder(tror jeg??)

Vi skal fikse sånn at Katt og Hund blir Dyr med chip!

Og så skal vi lage et dyreregister, der man kan søke opp dyr som har chip!
Da er det nice om vi kan definere hvilke dyr som har chip.

# Interface

```java
interface HarChip{
    Person hentEier();
    String hentId();
}
```

```java
class Katt extends Dyr implements HarChip{
    private Person eier;
    private String id;

    public Katt(String navn, int alder, Person eier, String id){
        super(navn, alder);
        this.eier = eier;
        this.id = id;
    }
    public Person hentEier(){
        return eier;
    }
    public String hentId(){
        return id;
    }
    @Override
    public String toString(){
        return super.toString() + "\nEier:" + eier.toString();
    }

}
```

```java
class Hund extends Dyr implements HarChip{
    private Person eier;
    private String id;

    public Hund(String navn, int alder, Person eier, String id){
        super(navn, alder);
        this.eier = eier;
        this.id = id;
    }
    public Person hentEier(){
        return eier;
    }
    public String hentId(){
        return id;
    }
    @Override
    public String toString(){
        return super.toString() + "\nEier:" + eier.toString();
    }
}
```

# Hashmap

Se filene for forelesning for å se hele programmet!

# Breakoutrooms

1. Slå på kamera og ha en presentasjonsrunde
2. Diskuter:
    a. Hva er forskjellen på interface og superklasse?
    b. Hva er fordelene på hashmap(ordbok)? Når burde man bruke det?
3. Jobb sammen med ukesoppgavene, de ligger på emnesiden -> grupper
    a. Enten ved at én deler skjerm med codecollab.io/
    b. OBS: codecollab er gratistjeneseter som UiO ikke har avtale med, sannsynligvis vil de samle data om dere. Dere kan fint løse oppgavene uten å bruke disse tjenestene!
4. Bruk “ask for help”-knappen for å få hjelp 😊
5. Vi møtes her igjen for å gå gjennom oppgavene til slutt (dere bestemmer hvilke)

# Send Marlen en direktemelding i chatten

Vil du jobbe sammen med noen andre ? (ja /nei)

Hvis du har noen ønsker på hvem du vil jobbe med, så send det i samme melding

Svar gjerne også om svaret skulle være nei

# Jobbe med oppgaver

# Ris, ros, forslag ?

https://nettskjema.no/a/180345