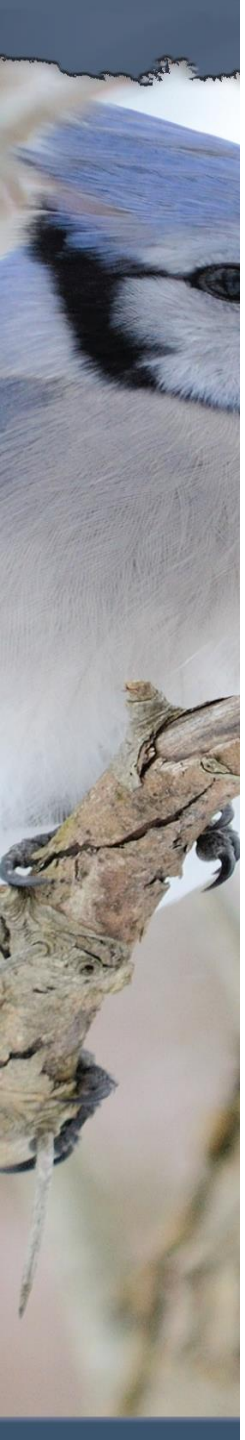
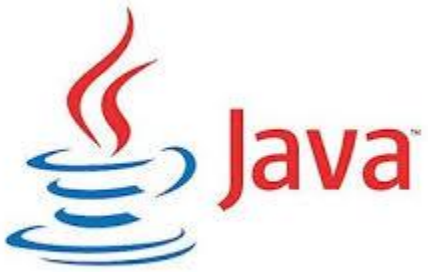
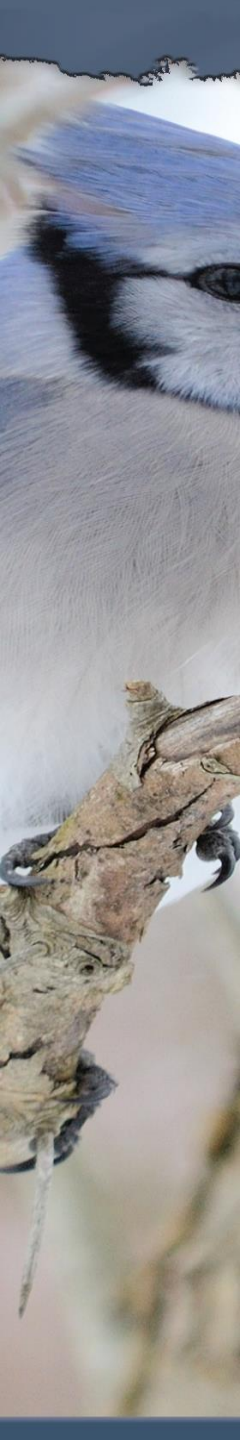


PGR101 – Objektorientert programmering 2

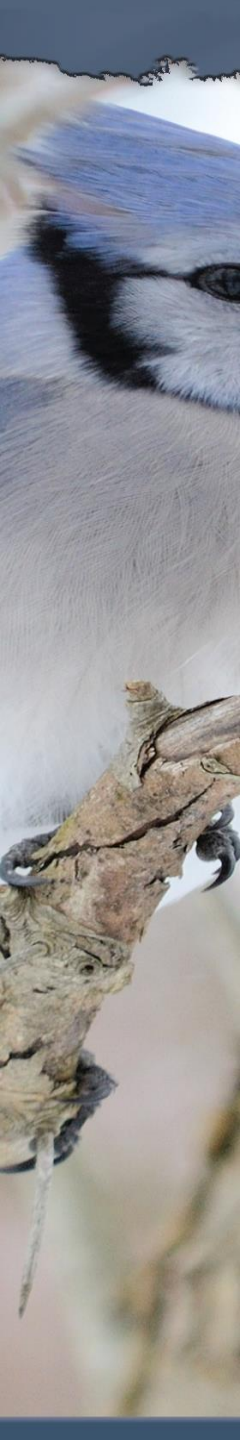


Det er bare å glede seg☺



Rammer

- Per og veiledere
- 200 timer
- BlueJ (IntelliJ)
- Emneplanen



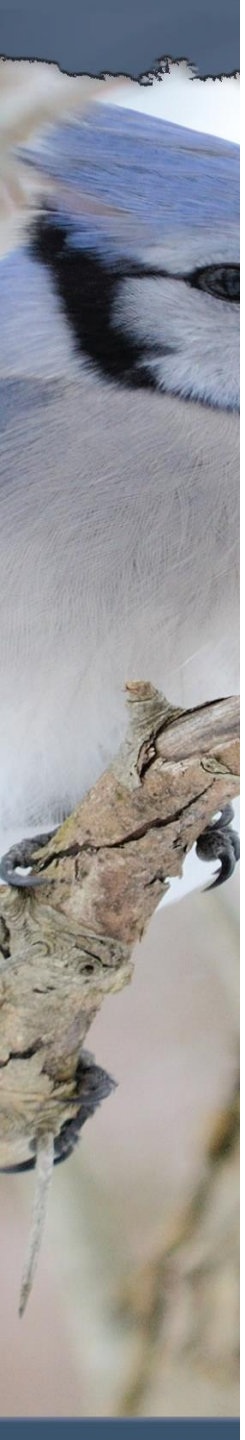
Emneplanen

- Arv mm
- Grafisk brukergrensesnitt (GUI)
  - Java FX!
- Unntakshåndtering
- Design patterns

- 3t skriftlig eksamen og to arbeidskrav

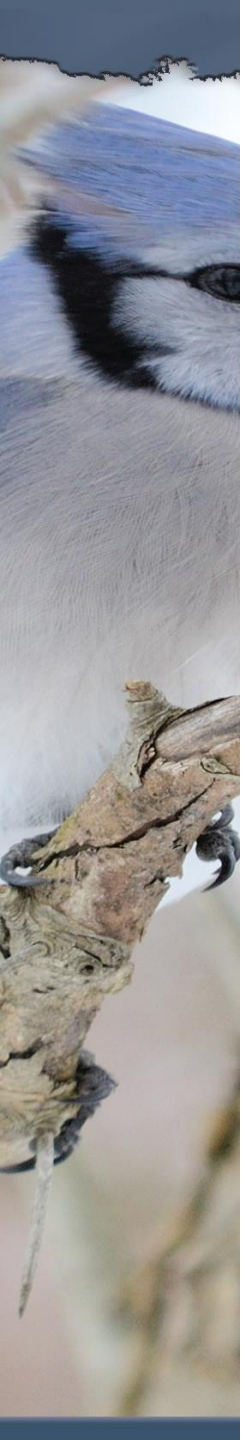
Kontekst

	Totalt		Kvinner		Menn	
Antall kandidater (oppmeldt):	745		143		602	
Antall møtt til eksamen:	600		118		482	
Antall bestått (B):	428		76		352	
Antall stryk (S):	172	29%	42	36%	130	27%
Antall avbrutt (A):	0		0		0	
Gjennomsnittskarakter:	C		D		C	
Antall med legeattest (L):	9		3		6	
Antall trekk før eksamen (T):	2		0		2	



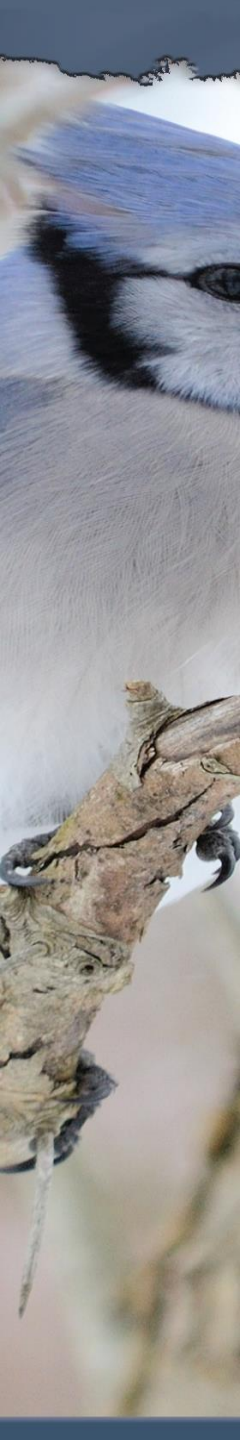
Utfordringer

- 1/3 av alle studenter stryker i «IntroProg»
- «Alle de andre kan så mye»
- Terminologi
- Katastrofalt å «falle av»



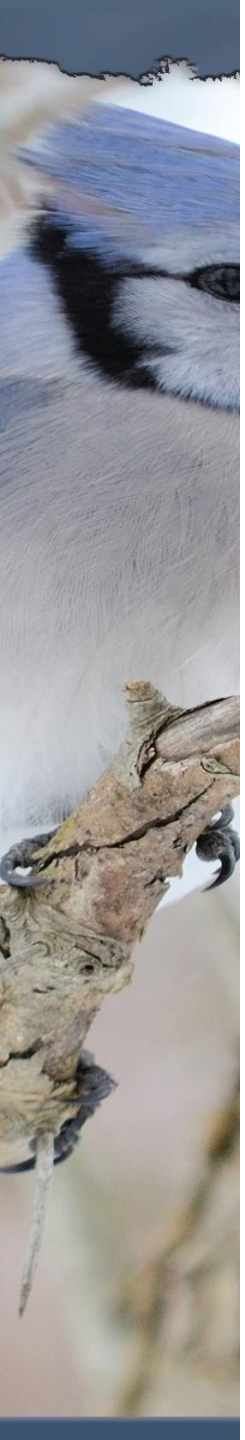
Gjennomføring

- Ikke flipped classroom, men…
- Sett av torsdag til kodedag. Alle ressurser ligger ute senest kl 0800
- Live koding
- Oppgaver i tillegg til de fra boka
- Poll hver uke: Hvilken oppgave skal Per lage videoløsningsforslag for?



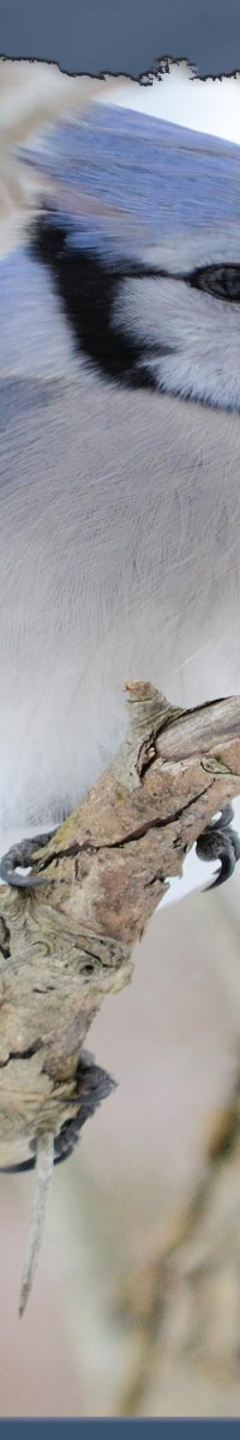
Live koding

- Bli med!
- Tenk versjonskontroll
  - Git?
  - Alltid utgangspunktet (projects fra boka) tilgjengelig. Da kan du manuelt hente opp utgangspunktet hvis du står veldig fast.

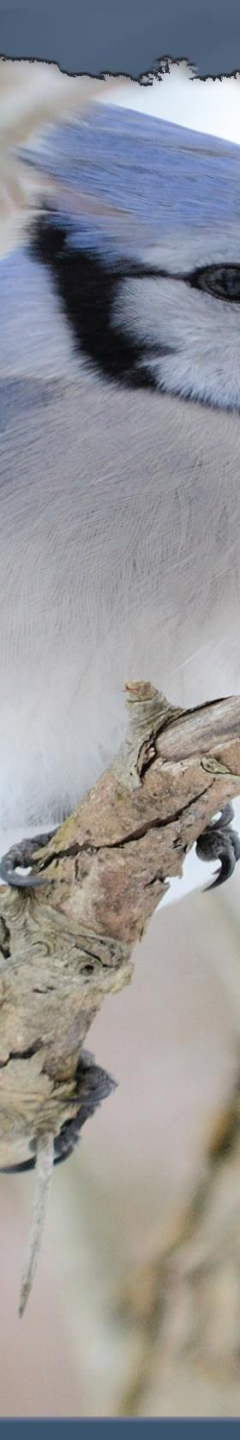


Hvorfor Java?

- Objektorientert programmering er fremdeles svært betydningsfullt i industrien.
- Java og C# er de mest brukte objektorienterte språkene i dag.

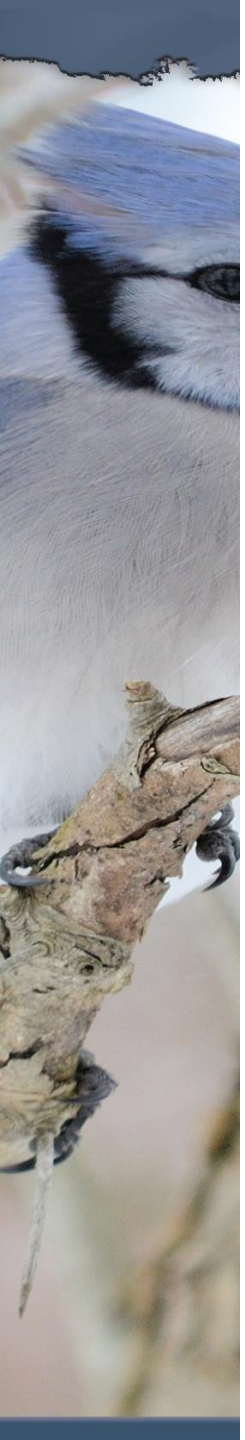


Improving structure with inheritance



Main concepts to be covered

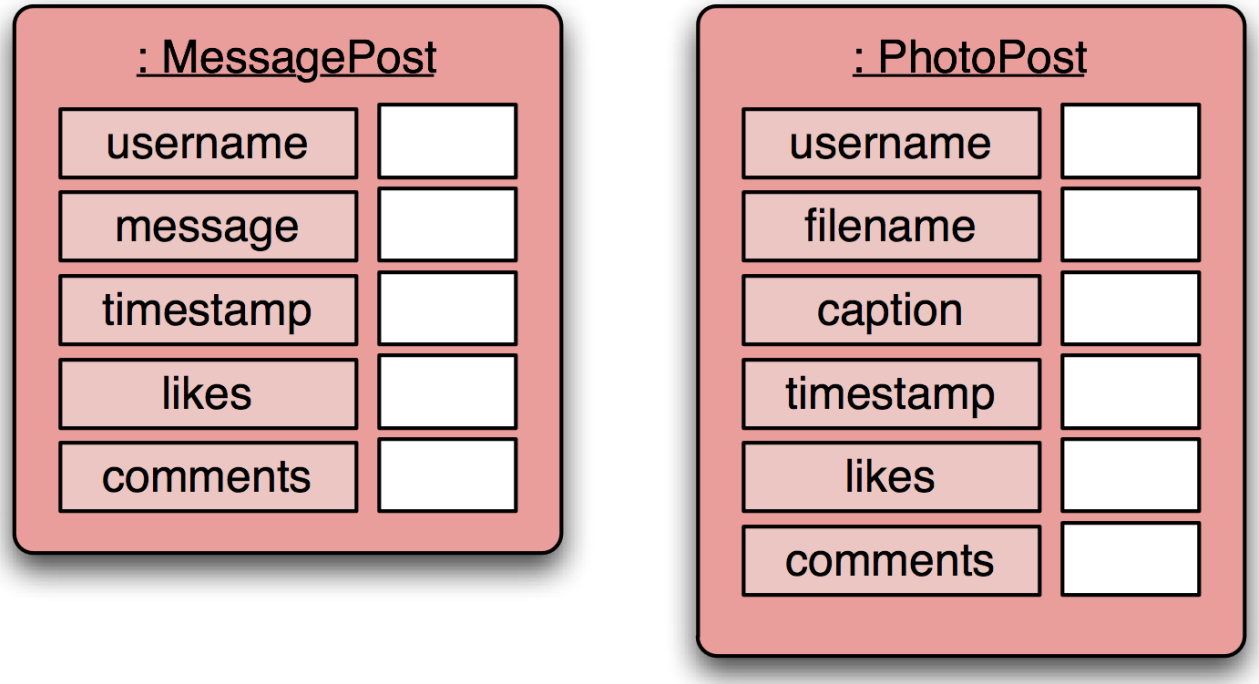
- Inheritance
- Subtyping
- Substitution
- Polymorphic variables



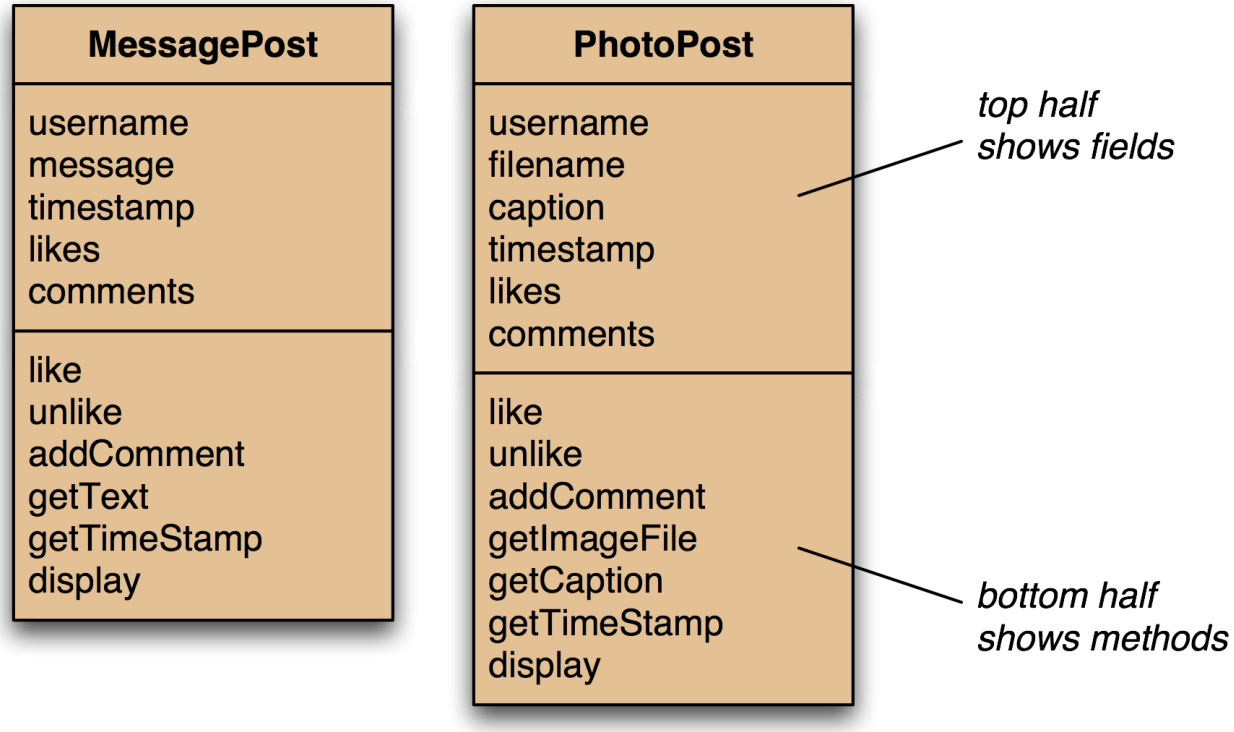
The Network example

- A small, prototype social network.
- Supports a news feed with posts.
- Stores *text posts* and *photo posts*.
  - **`MessagePost`**: multi-line text message.
  - **`PhotoPost`**: photo and caption.
- Allows operations on the posts:
  - E.g., search, display and remove.

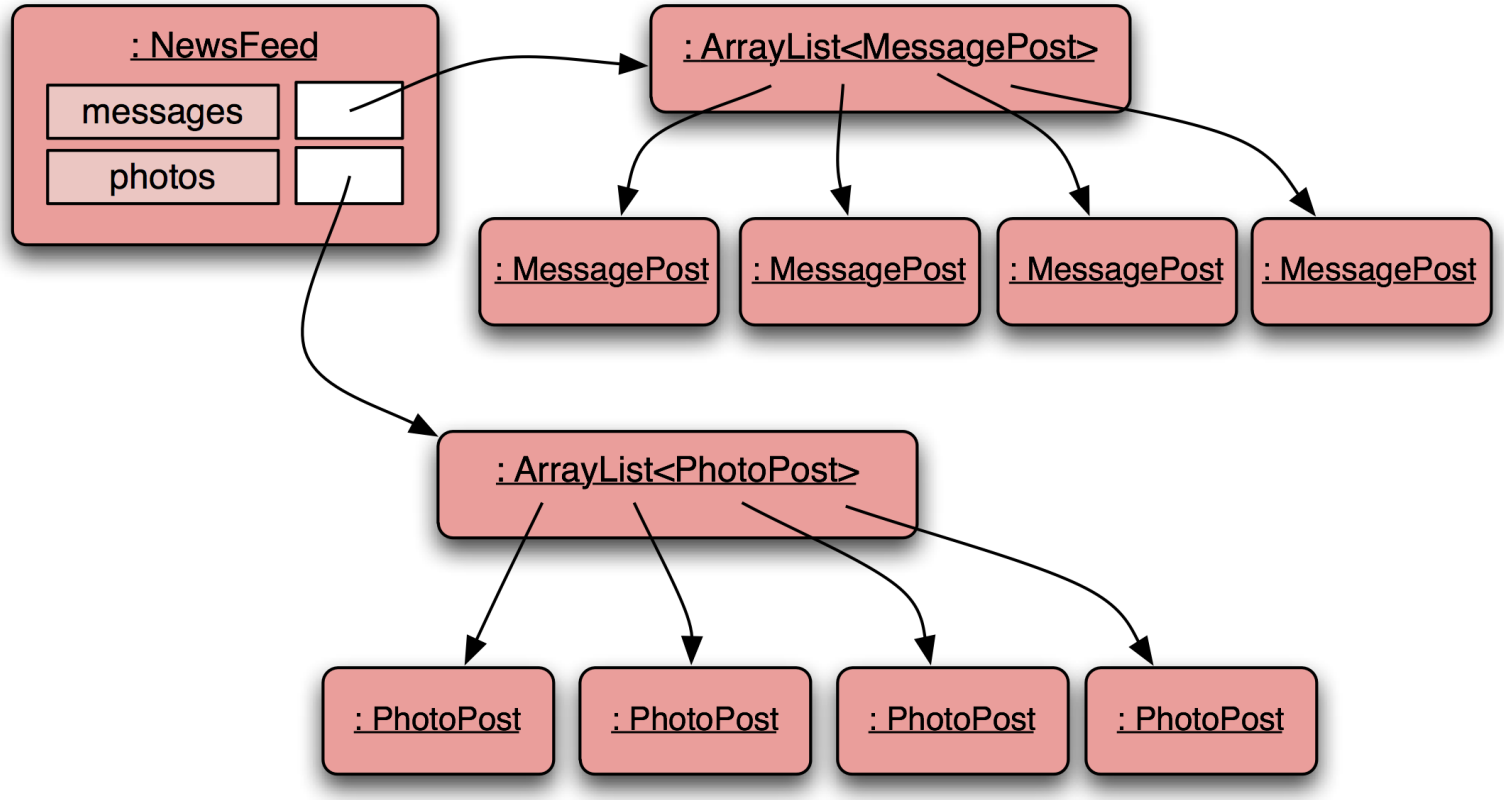
Network objects



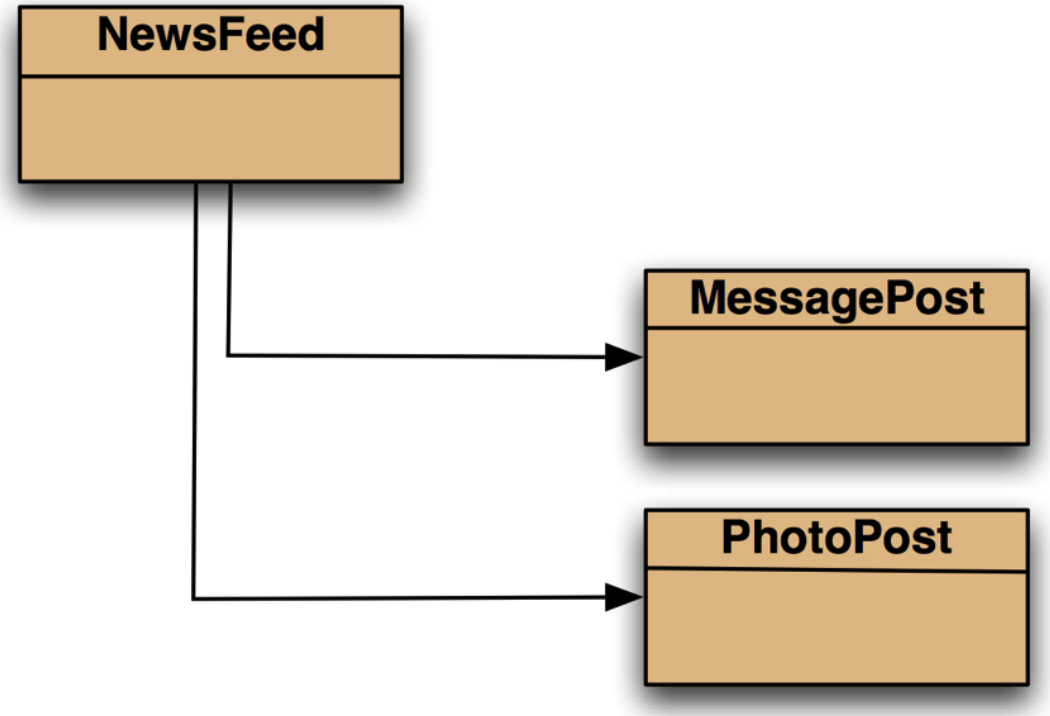
Network classes

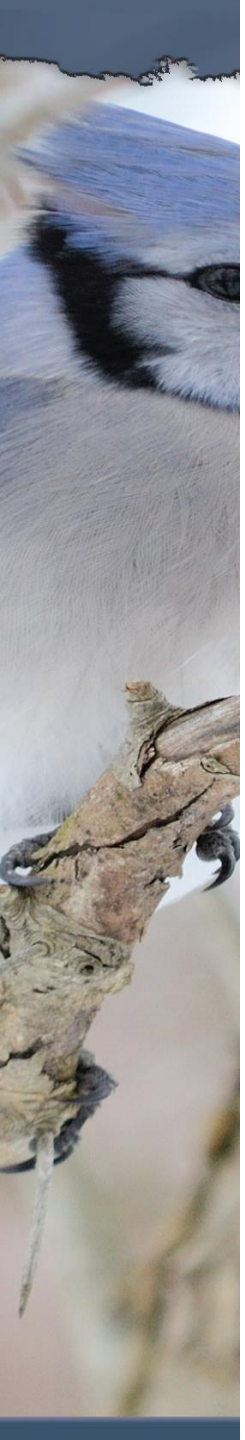


Network object model



Class diagram





Message-Post source code

Just an outline

```
public class MessagePost
{
   private String username;
   private String message;
   private long timestamp;
   private int likes;
   private ArrayList<String> comments;

   public MessagePost(String author, String text)
   {
        username = author;
        message = text;
        timestamp = System.currentTimeMillis();
        likes = 0;
        comments = new ArrayList<>();
   }

   public void addComment(String text) ...

   public void like() ...

   public void display() ...

   ...
}
```

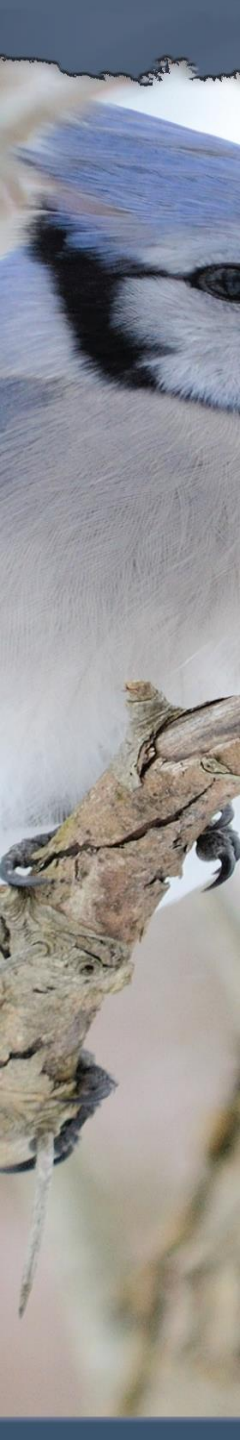
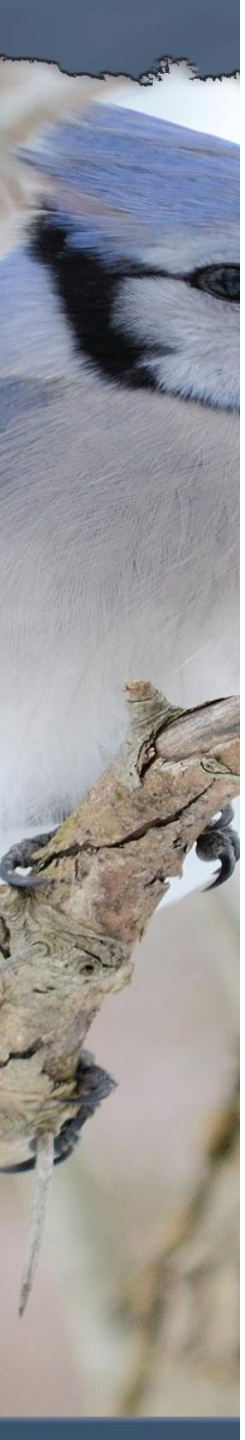


Photo-Post source code

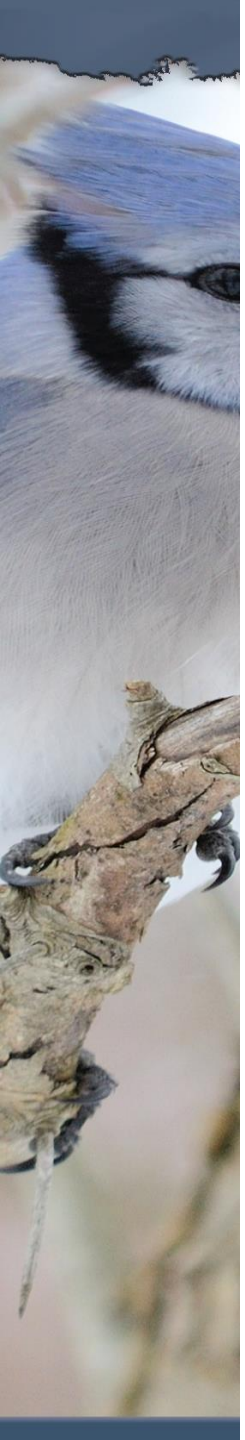
Just an outline

```
public class PhotoPost
{
   private String username;
   private String filename;
   private String caption;
   private long timestamp;
   private int likes;
   private ArrayList<String> comments;

   public PhotoPost(String author, String filename,
                    String caption)
   {
        username = author;
        this.filename = filename;
        this.caption = caption;
        timestamp = System.currentTimeMillis();
        likes = 0;
        comments = new ArrayList<>();
   }

   public void addComment(String text) ...
   public void like() …
   public void display() …
   ...
}
```



NewsFeed

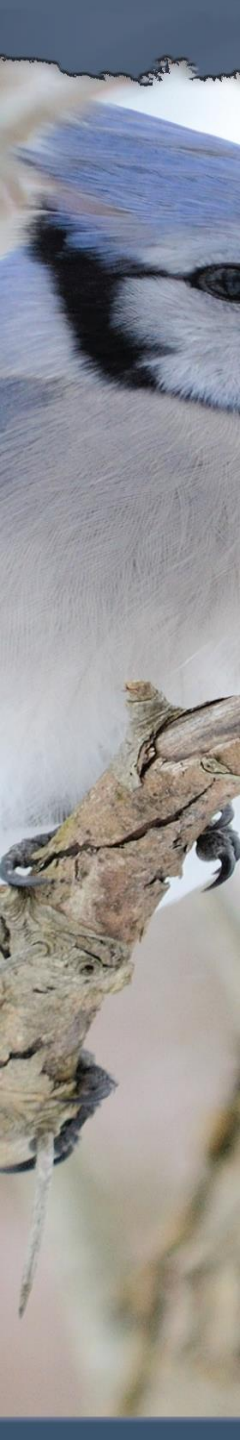
```
public class NewsFeed
{
   private ArrayList<MessagePost> messages;
   private ArrayList<PhotoPost> photos;
   ...
   public void show()
   {
       for(MessagePost message : messages) {
          message.display();
          System.out.println();  // empty line between posts
       }

       for(PhotoPost photo : photos) {
          photo.display();
          System.out.println();  // empty line between posts
       }
   }
}
```



To oppgaver (velg den du vil starte med)

- A): Legg til en metode i NewsFeed som returnerer hvor mange Posts den inneholder (totalt).
- B): Sørg for at alle nyopprettede Posts har en startmelding som lyder: “Ufine kommentarer vil bli slettet!”

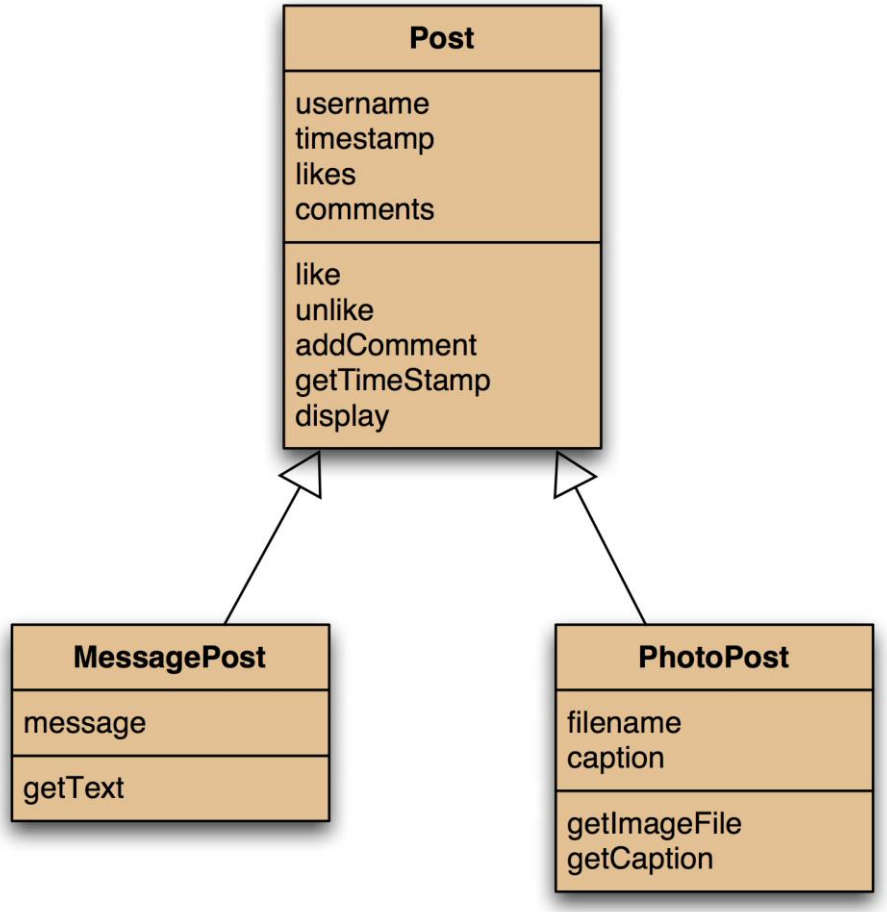
- Sjekk at løsningen din fungerer…

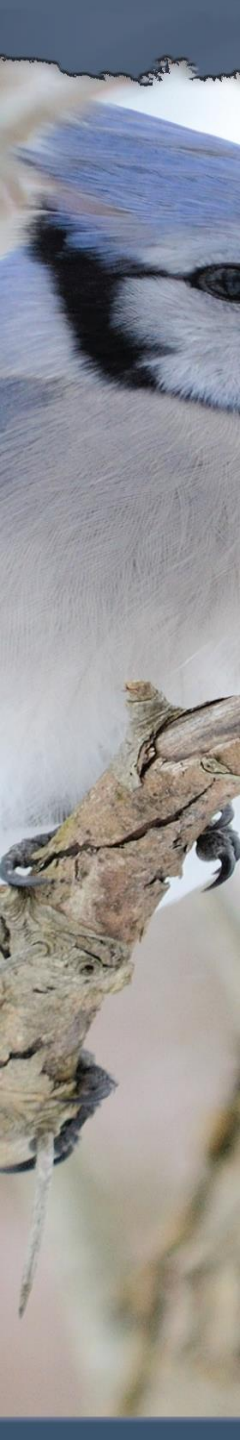


Critique of Network

- Code duplication:
  - **`MessagePost`** and **`PhotoPost`** classes very similar (large parts are identical)
  - makes maintenance difficult/more work
  - introduces danger of bugs through incorrect maintenance
- Code duplication in **`NewsFeed`** class as well.

Using inheritance

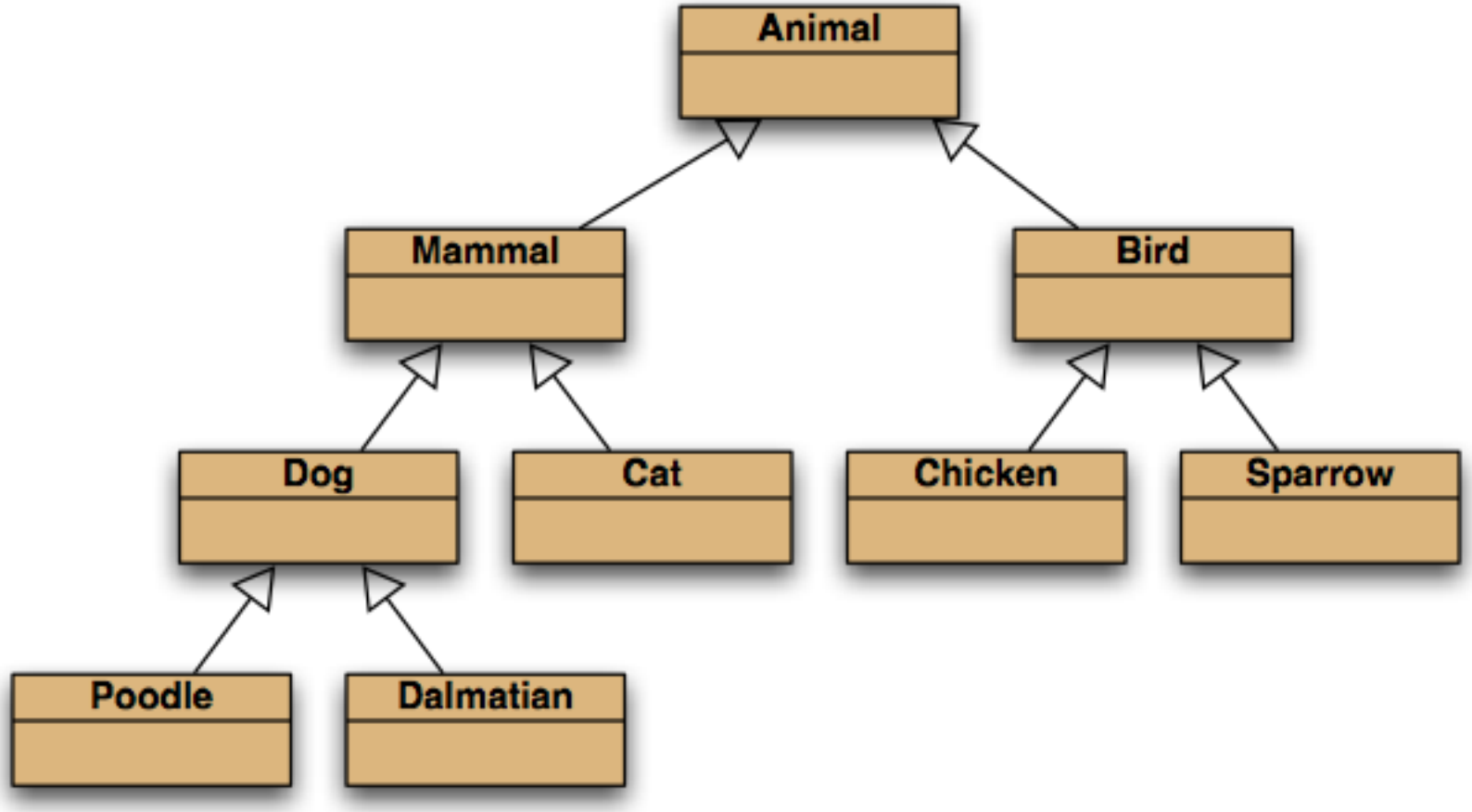


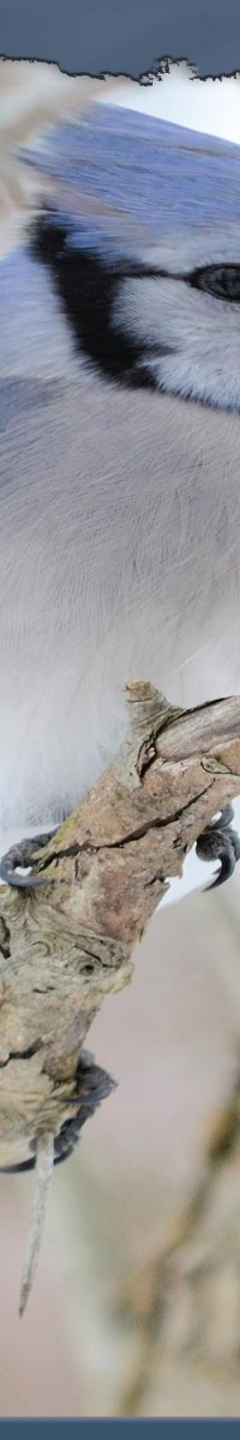


Using inheritance

- define one **superclass** : `Post`
- define **subclasses** for `MessagePost` and `PhotoPost`
- the superclass defines common attributes (via fields)
- the subclasses **inherit** the superclass characteristics
- the subclasses add other characteristics

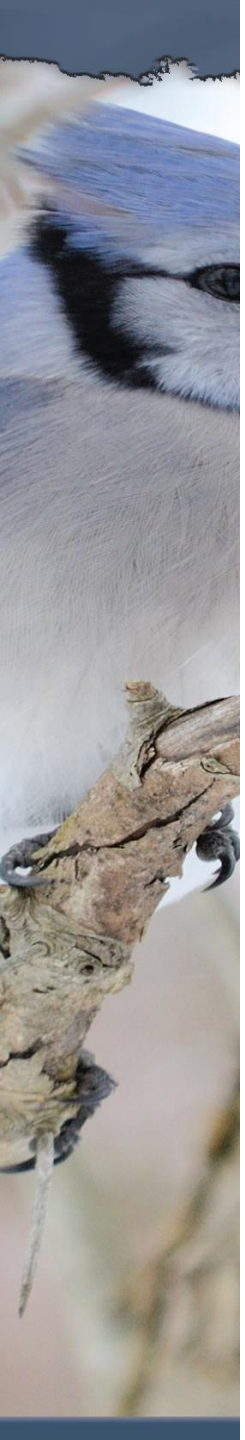
Inheritance hierarchies





is-a? has-a? relationships

- Is-a: Arv. Subklassen **er** en spesialisering av superklassen. Eks: En hund ***er*** et pattedyr.
- Has-a: Komposisjon. En klasse kan ***ha*** tilgang til en annen klasse. En hund kan ***ha*** en eier.



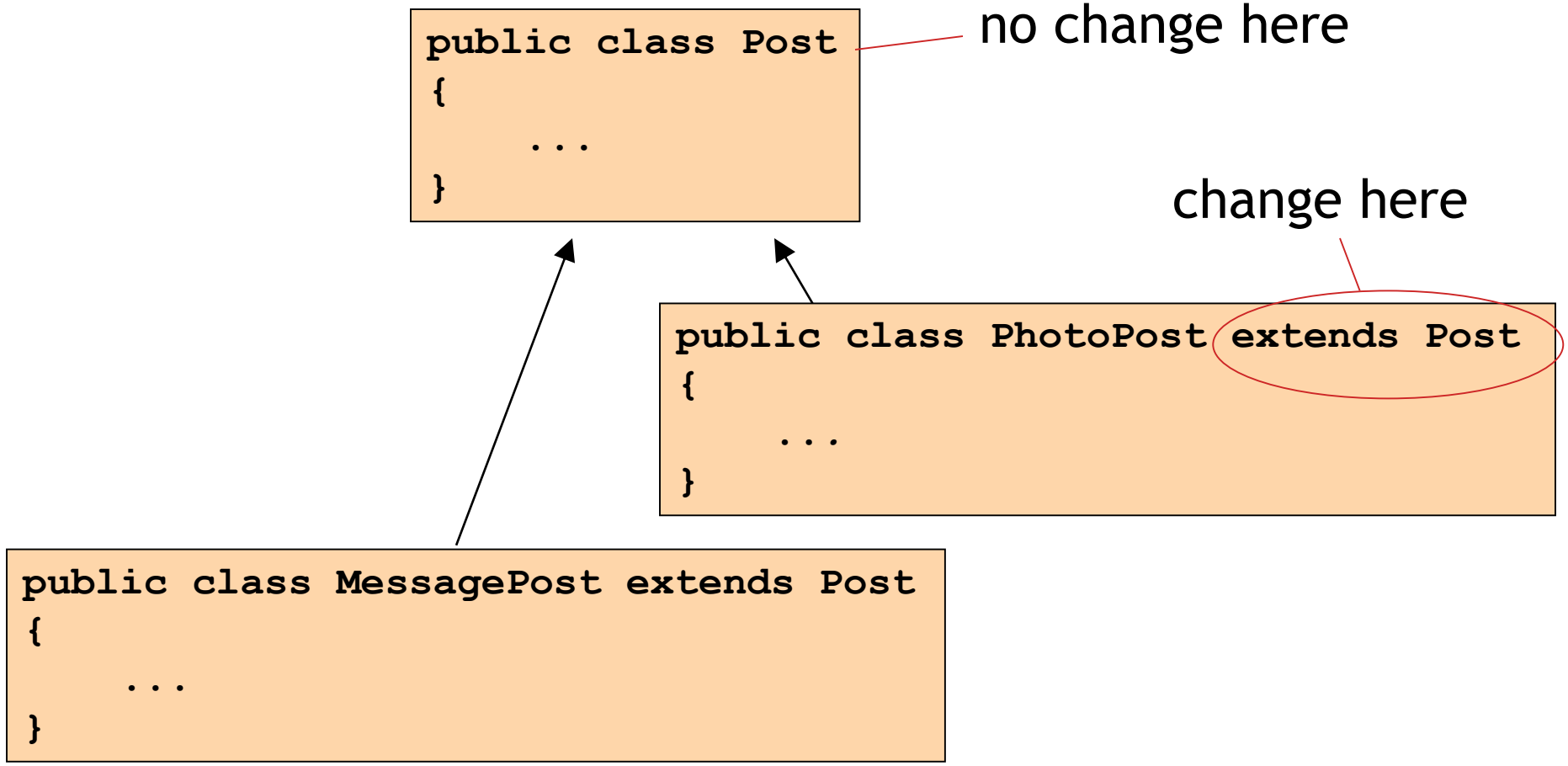
Oppgaver! is-a? eller has-a?

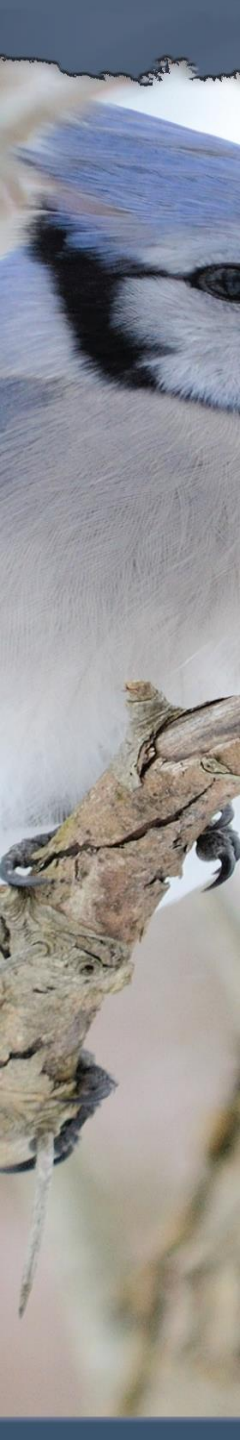
- Hvis du tror det er arv (is-a). Barn-forelder. Søsken: Bli sittende.
- Hvis du tror det er komposisjon (has-a): Reis deg opp.

Bil - Motor

Det blir flere oppgaver i Kahooten…

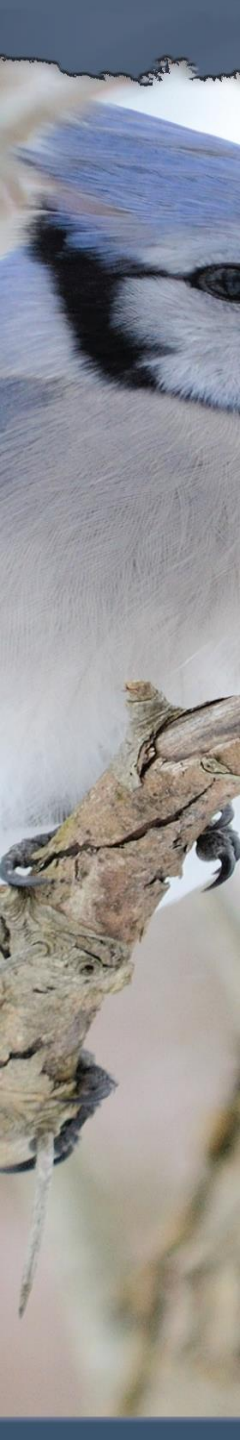
Inheritance in Java





Superclass

```
public class Post
{
    private String username;
    private long timestamp;
    private int likes;
    private ArrayList<String> comments;

    // constructor and methods omitted.
}
```



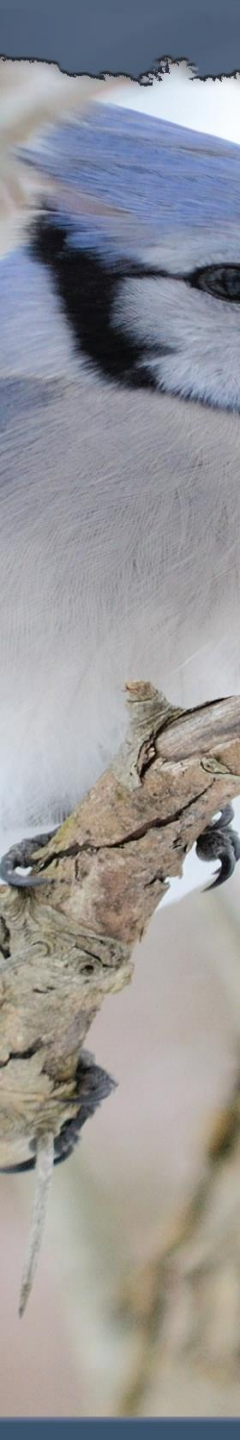
Subclasses

```
public class MessagePost extends Post
{
    private String message;

    // constructor and methods omitted.
}
```

```
public class PhotoPost extends Post
{
    private String filename;
    private String caption;

    // constructor and methods omitted.
}
```

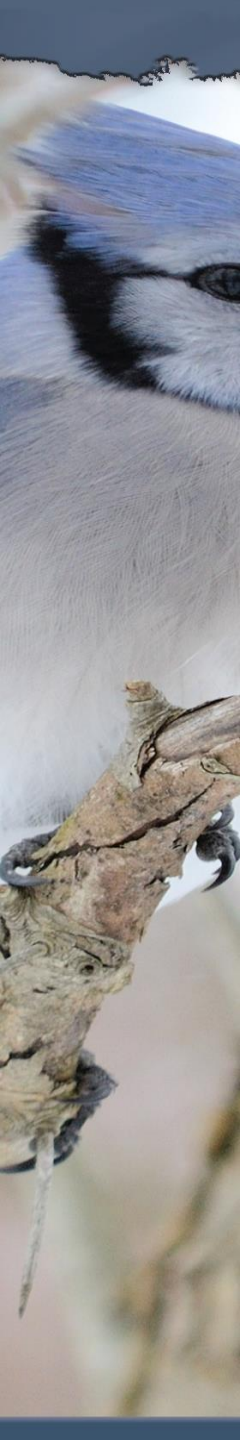
Inheritance and constructors

```
public class Post
{
    private String username;
    private long timestamp;
    private int likes;
    private ArrayList<String> comments;

    /**
     * Initialise the fields of the post.
     */
    public Post(String author)
    {
        username = author;
        timestamp = System.currentTimeMillis();
        likes = 0;
        comments = new ArrayList<>();
    }

    // methods omitted
}
```



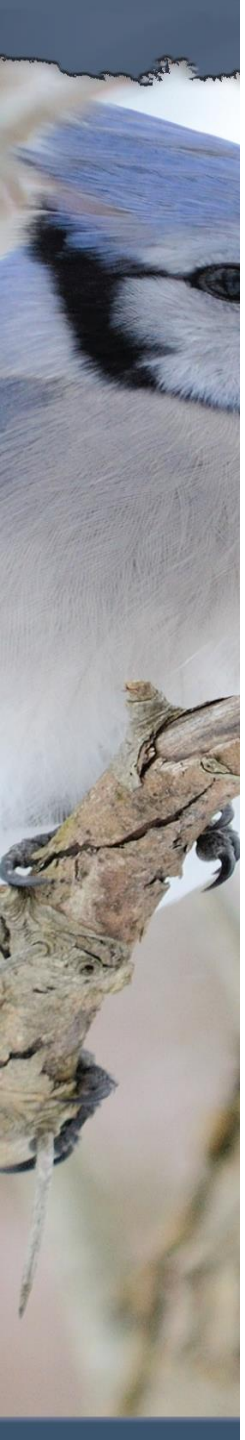
Inheritance and constructors

```
public class MessagePost extends Post
{
    private String message;

    /**
     * Constructor for objects of class MessagePost
     */
    public MessagePost(String author, String text)
    {
        super(author);
        message = text;
    }

    // methods omitted
}
```



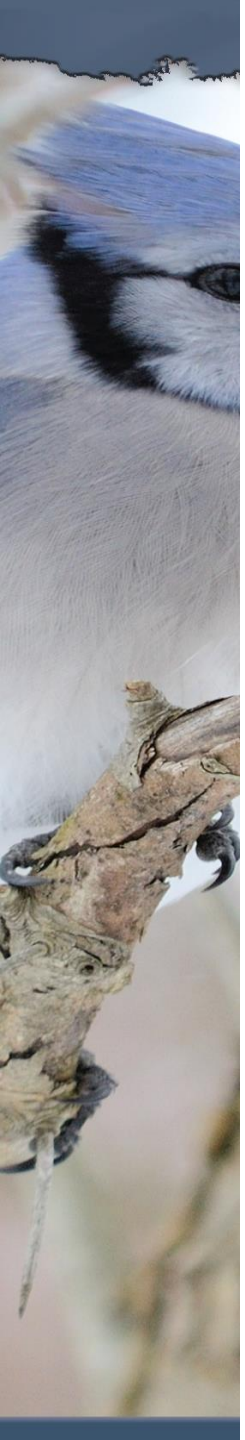
Superclass constructor call

- Subclass constructors must always contain a 'super' call.
- If none is written, the compiler inserts one (without parameters)
  - only compiles if the superclass has a constructor without parameters
- Must be the first statement in the subclass constructor.



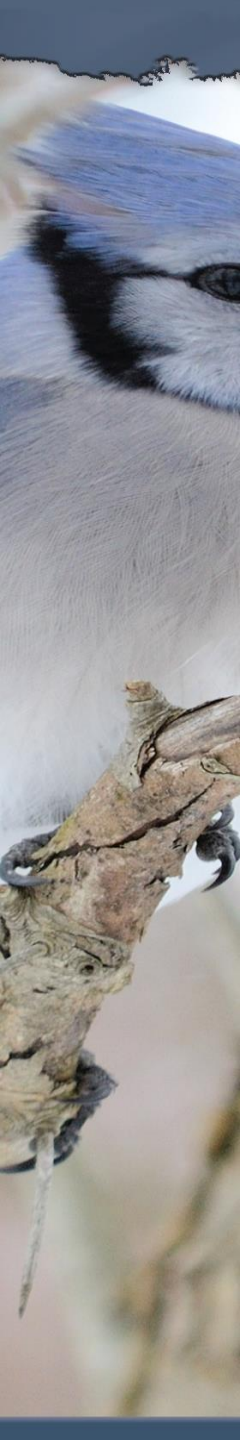
Private access

- Subclasses cannot access private fields in a superclass.
- Use getters in superclass if you want access…



Oppgave!

- Lag superklassen Post, og la klassene MessagePost og PhotoPost arve fra denne.
- Unngå unødvendig duplisering av kode.
- Koden skal funksjonelt sett fungere som tidligere.



Neste gang

- Resten av kapittel 10 (arv).
- Etter Kahooten: Øving! Sjekk TimeEdit for rom…

- Nå: Kahoot!