Læringsmål for forelesningen

- Objektorientering
  - Delegeringsteknikken
  - Delegering vs. arv

Dagens forelesning

- Introduksjon og motivasjon
  - Hvorfor forelese om standardteknikker, såkalte patterns?
  - Hva slags problem løses med delegering?
  - Delegering vs. arv
- Delegeringseksempler
  - Person.children-liste
  - Sjefen og den som gjør jobben
  - Kokk og medhjelpere
  - «AnnenhverIterator» ved merging
  - Hierarkiske innstillinger (denne er hard)
  - …

Bortsett fra arv...
så kan dere det viktigste om Java-språket!

Men... programmering er mer enn Java, dere trenger også lære om standard kodingsteknikker, f.eks. god strukturering av koden og bruk av flere API'er

Standard kodingsteknikker

- "*a design pattern is a general repeatable solution to a commonly occurring problem in software design*"

- Gjenbrukbare kodingsteknikker som har vist seg å være praktiske/effektive løsninger

- Standardteknikker er ofte om såkalte samhandlings-mønstre – fordeling av logikk på flere objekter
  - **delegering** – et objekt får hjelp fra andre objekter
  - **observatør – observert** – et objekt oppdateres om hva et annet objekt gjør

Komposisjon og aggregering

- Komposisjon: *del-av*-relasjon (*hører-til*)
  - Spesialtilfelle av aggregering
  - B er *en del av* A
  - Når A dør, så dør samtidig B (hva med motsatt?)
  - Sterk kobling
  - Kardinalitet?

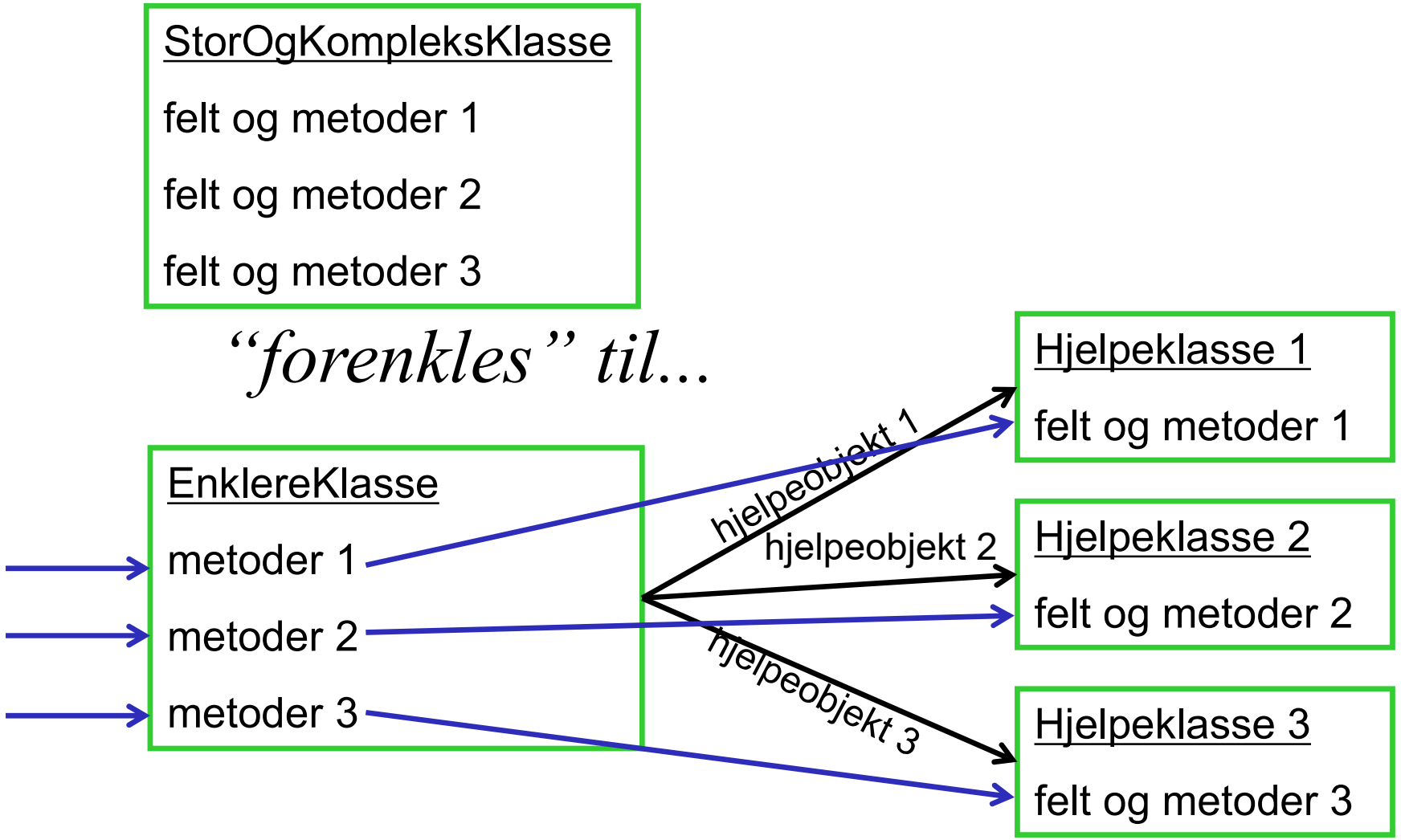
- Agreggering: *har-en*-relasjon
  - A *har en* B
  - A og B kan leve uavhengig av hverandre
  - Svak kobling
  - Kardinalitet?
- (Arv? – *er-en*-relasjon)

Delegering

- Et objekt fremstår med en ferdighet, men overlater til et annet (internt) objekt å gjøre jobben
- Relasjonen kan være via komposisjon eller aggregering
- (Matlagings)analogi
  - (kjøkken)sjefen har ansvaret, og har (kun) en koordinerende rolle overfor underordnede (kjøkken)medarbeidere
  - når kjøkkensjefen tar på seg et oppdrag (en oppskrift), så kan hun fremstå som å kunne utføre det, fordi (en av) medarbeiderne kan gjøre det for henne, dvs. få delegert oppgaven til seg
- Mer fleksibel teknikk enn arv for gjenbruk av eksisterende funksjonalitet

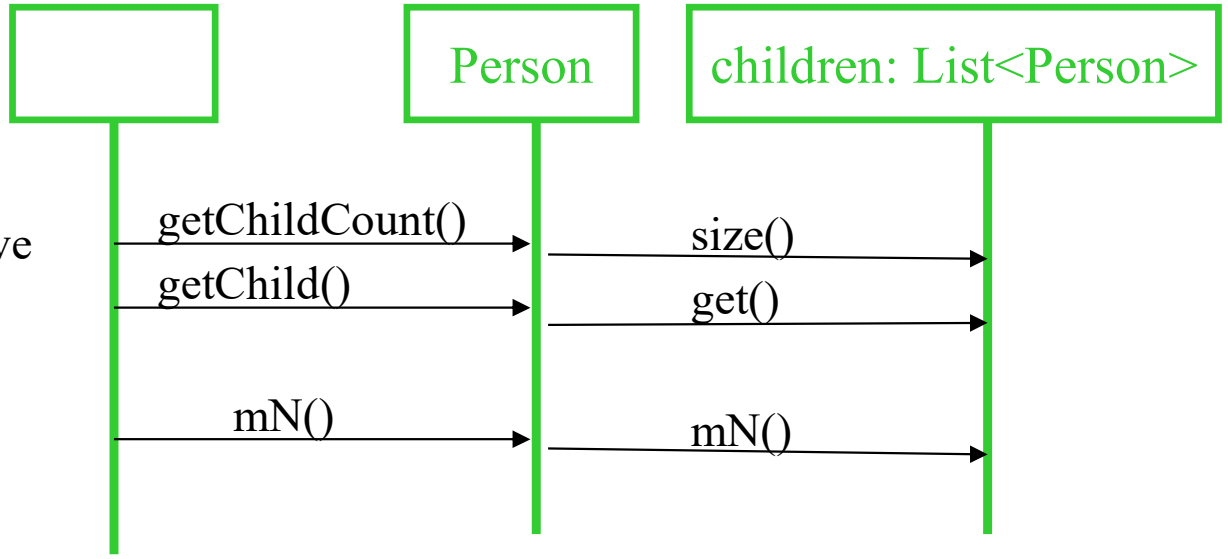
Liten digresjon...

- Delegering ligner litt på en teknikk en ofte bruker for å forenkle store klasser:



Eksempel: Person.children-liste

- Person-objekt delegerer til List<Person>-objekt
  - Delegerende: Person
  - Delegat: List<Person>
- Metoder m1.. mN:
  - getChildCount -> size
  - getChild -> get
  - addChild -> add
  - removeChild -> remove
  - ...



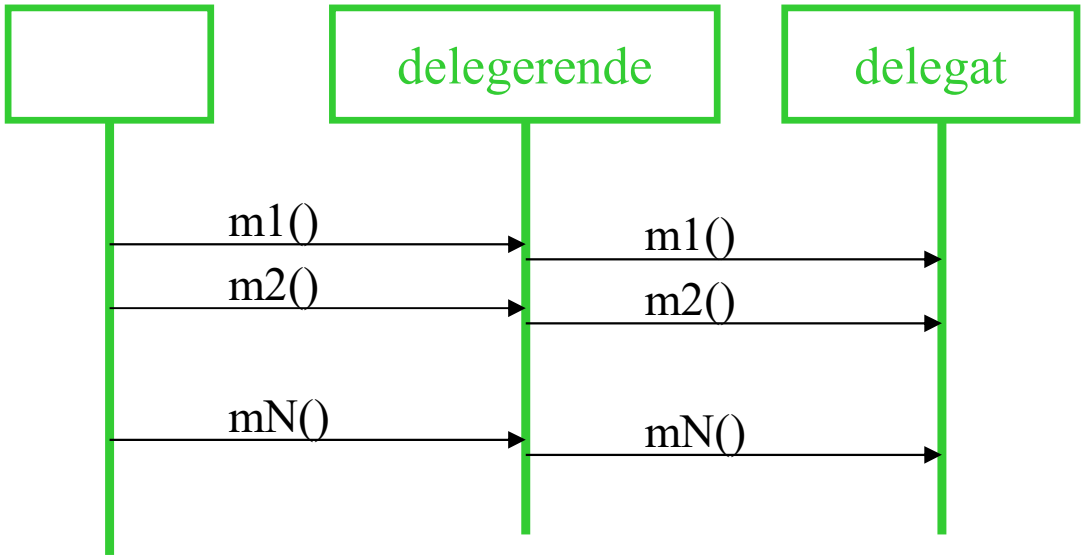
- Hva med arving? Hvordan og hvorfor ikke?

Se kode

- person

Generell delegeringsløsning

- To objekter: delegerende og delegat
  - **delegerende** eier/har en referanse til **delegat**
  - **delegerende** har et sett metoder m1..mN som (noenlunde direkte) tilsvarer metoder hos **delegat**
  - kall til metodene m1..mN på **delegerende** fører (noenlunde direkte) til tilsvarende kall på **delegat**



Eksempel: IO

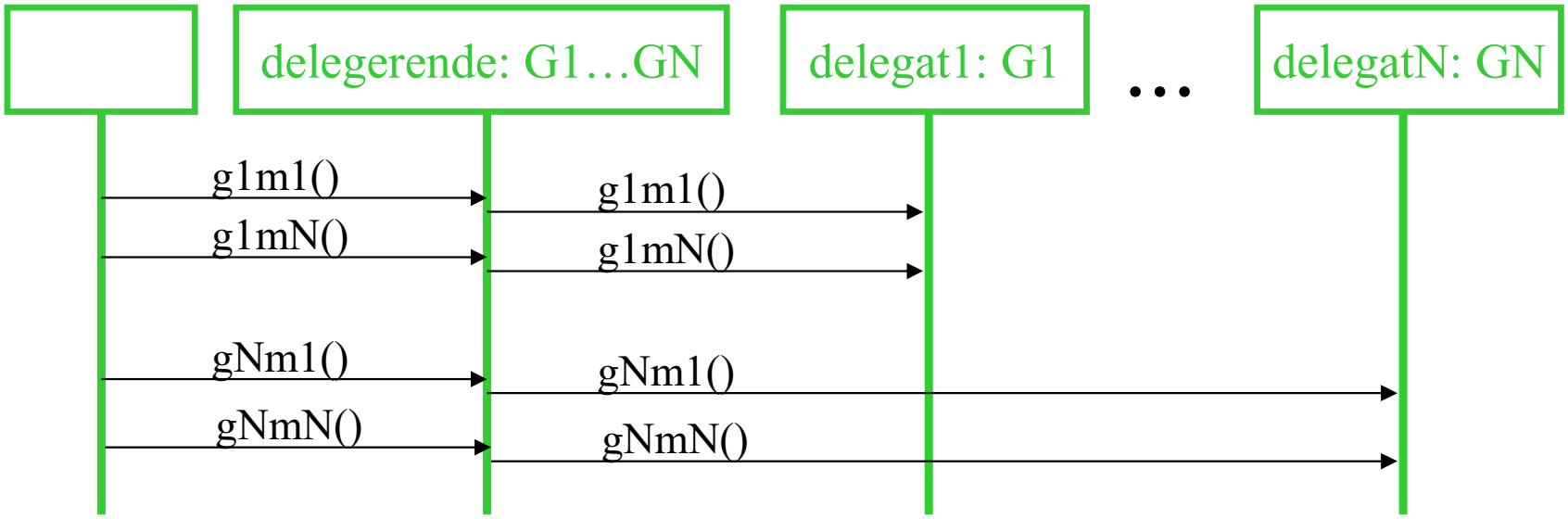
- Java sitt IO-system er bygd på delegering:
  - alle klassene bygger på en av klassene Writer, Reader, OutputStream og InputStream
- Mange av klassene delegerer til implementasjoner av disse, basert på riggingen av objektene med konstruktører
  - new InputStreamReader(inputStream)
    ny InputStreamReader – delegerer til inputStream
  - new BufferedReader(reader)
    ny BufferedReader – delegerer til reader
  - new PrintWriter(writer)
    ny PrintWriter – delegerer til writer
  - ...
- Kall til metoder som read/readln, write og print/println delegerer til det indre objektet sine metoder

Se på et par eksempler

- boss
- kokk
- Merk at en ikke må definere interface for å kunne bruke delegering. Det er bare nyttig.

Delegering og grensesnitt

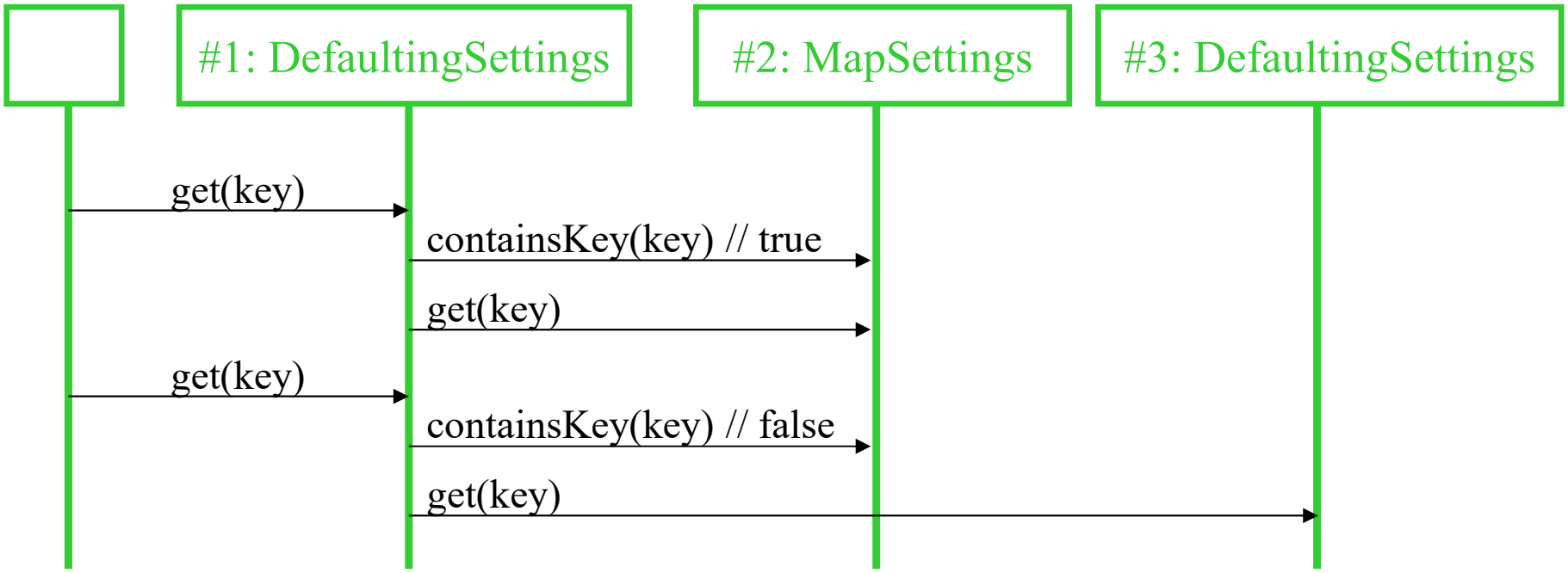
- Metodene m1.. mN er ofte deklarert i et grensesnitt som både **delegerende** og **delegat** implementerer
- **Delegerende** kan implementere flere slike grensesnitt og har da én **delegat** for hvert



- Dette er en form for *komposisjon*:
  helheten er summen av egenskapene til delene

Eksempel: Settings med defaults

- En begrenset Map-aktig klasse for hierarkiske innstillinger/kjede med defaults.



- Hva med arving? Hvordan og hvorfor ikke?

Eksempel: Settings med defaults

```
package delegation;

public interface ISettings {
        public boolean hasSetting(String key);
        public Object getSetting(String key);
        public void setSetting(String key, Object value);
}

public class MapSettings implements ISettings {

        private Map<String, Object> settings = new HashMap<String, Object>();

        @Override
        public boolean hasSetting(String key) {
                return settings.containsKey(key);
        }

        @Override
        public Object getSetting(String key) {
                return settings.get(key);
        }

        @Override
        public void setSetting(String key, Object value) {
                settings.put(key, value);
        }
}
```

Eksempel: Settings med defaults

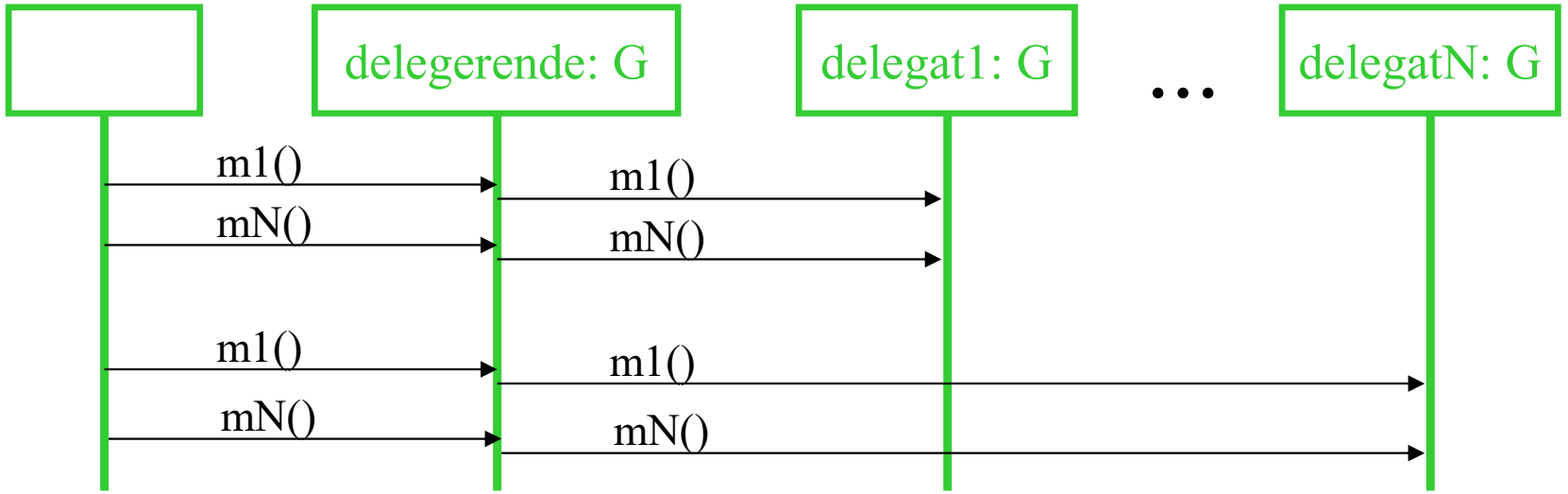
```
public class DefaultingSettings implements ISettings {

        private ISettings defaults = null;
        private ISettings local = new MapSettings();

        public void setDefaultSettings(ISettings defaults) {
                this.defaults = defaults;
        }

        public void setLocalSettings(ISettings local) {
                this.local = local;
        }

        @Override
        public boolean hasSetting(String key) {
                return local.hasSetting(key) ||
                        (defaults != null && defaults.hasSetting(key));
        }

        @Override
        public Object getSetting(String key) {
                if (local.hasSetting(key)) {
                        return local.getSetting(key);
                }
                if (defaults != null) {
                        return defaults.getSetting(key);
                }
                return null;
        }

        @Override
        public void setSetting(String key, Object value) {
                local.setSetting(key, value);
        }
}
```

Eksempel: Settings med defaults

```
public static void main(String[] args) {
        MapSettings commonMapSettings = new MapSettings();
        commonMapSettings.setSetting("font.family", "Arial");
        commonMapSettings.setSetting("font.size", 12);
        MapSettings personalMapSettings = new MapSettings();

        DefaultingSettings commonSettings = new DefaultingSettings();
        commonSettings.setLocalSettings(commonMapSettings);

        DefaultingSettings personalSettings = new DefaultingSettings();
        personalSettings.setDefaultSettings(commonSettings);
        personalSettings.setLocalSettings(personalMapSettings);

        // personalSettings 'defaults' to common settings
        System.out.println(personalSettings.getSetting("font.family"));
        System.out.println(personalSettings.getSetting("font.size"));
        // override defaults
        personalSettings.setSetting("font.family", "Times Roman");
        // and check that it works
        System.out.println(personalSettings.getSetting("font.family"));

        // change defaults
        commonSettings.setSetting("font.size", 14);
        // and check that it affects personal settings
        System.out.println(personalSettings.getSetting("font.size"));
}
```

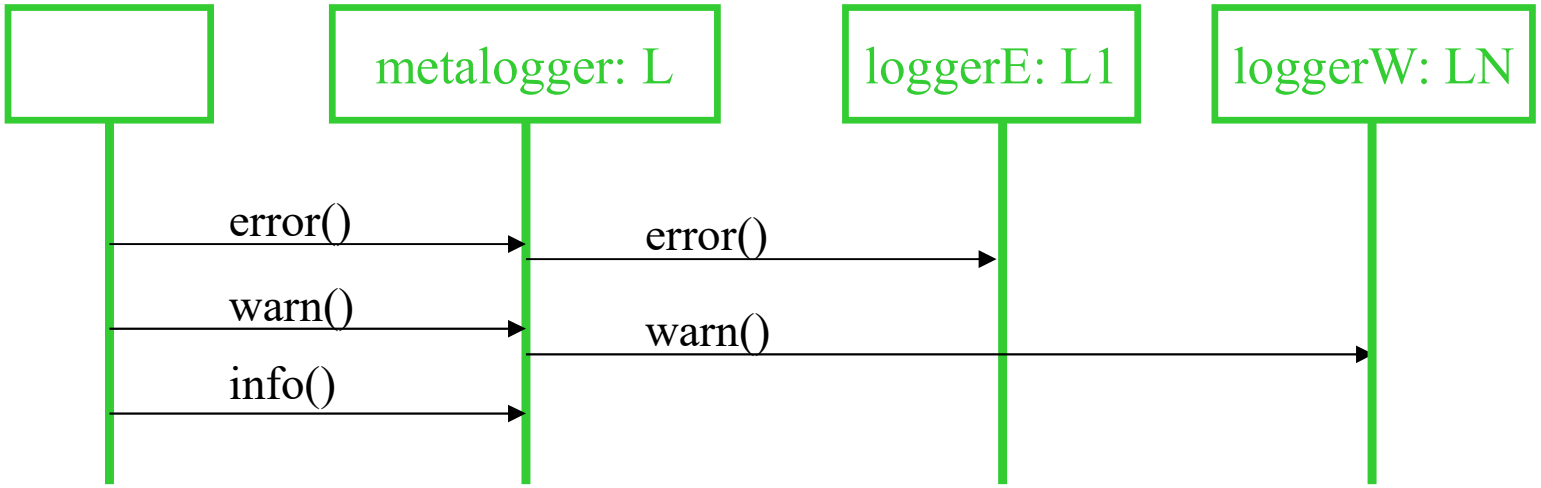
Varianter av delegering

- Flere **delegater** implementerer samme grensesnitt
- **Delegerende** velger hvilken det delegeres til i hvert tilfelle, basert på ikke-funksjonelle egenskaper
  - én implementasjon er raskere for bestemte argumenter
  - jevn fordeling av last på underliggende maskinvare
  - konfigurasjon som avhenger av formål, f.eks. drift, testing
- Delegerende kan tilpasse og filtrere metodekallene



Eksempel: logging

- Logging brukes for å samle informasjon om kjøring av program
- Mange rammeverk for logging, med noenlunde samme funksjonalitet
  - registrering av div. info om kilden: klasse, tidspunkt, ...
  - meldinger med ulik alvorlighetsgrad: **fatal**, **error**, warning og **info**
  - ulike måter å rapportere på: til fil, meldingspanel, rss, twitter, ...
  - regler for hvilke meldinger som skal havne hvor, f.eks. kaste **info**, lagre warning til fil, sende **error** til admin og **fatal** til brannvesenet
- Såkalte meta-loggere lar deg programmere med ett rammeverk, men delegere til et eller flere andre



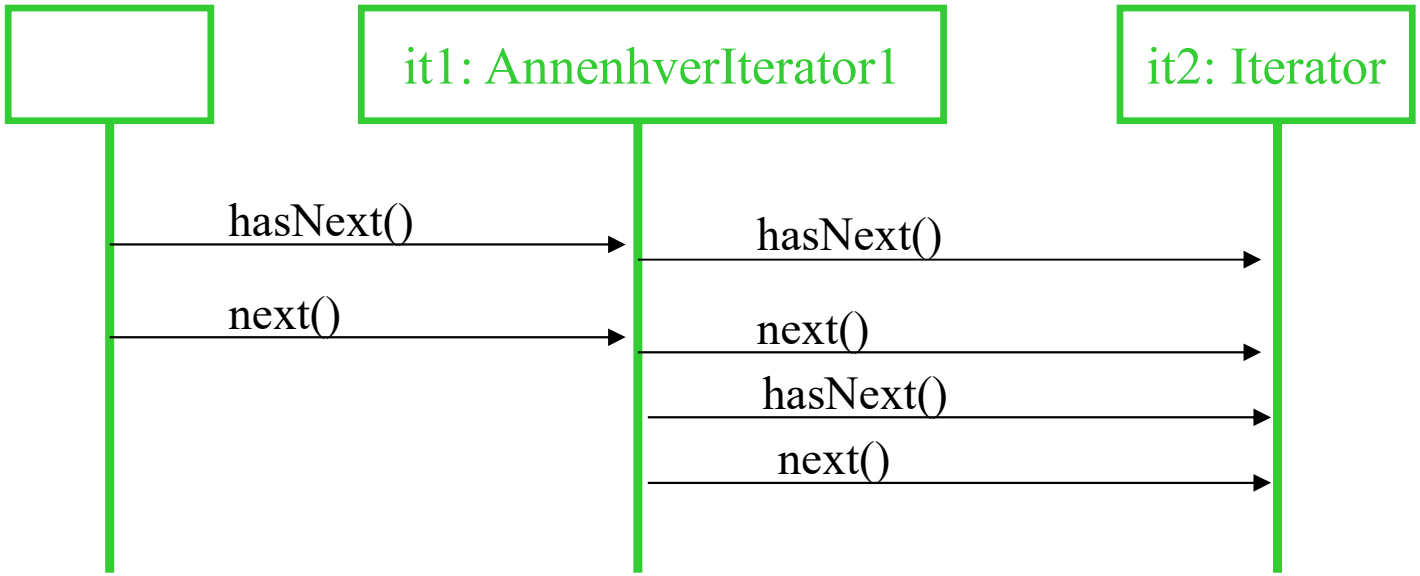
- Hva med arving? Hvordan og hvorfor ikke?

Logging-oppgave, del av øving 9

https://www.ntnu.no/wiki/display/tdt4100/Delegering+-+Logger-oppgave

Eksempel: AnnenhverIterator1

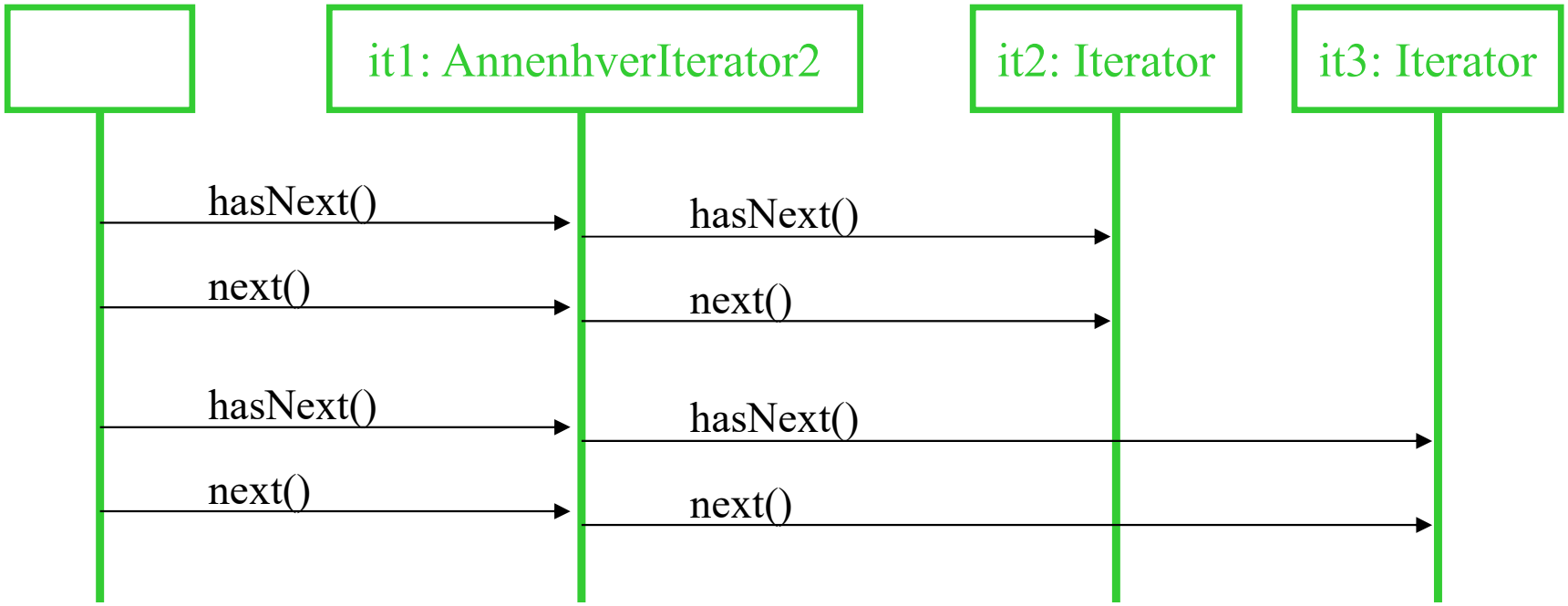
- AnnenhverIterator1 delegerer til intern Iterator, men "kaster" annenhvert element



- Hva med arving? Hvordan og hvorfor ikke?

Eksempel: AnnenhverIterator2

- AnnenhverIterator2 har to interne Iterator-objekter, og delegerer til dem annenhver gang



- Hva med arving? Hvordan og hvorfor ikke?

Læringsmål for forelesningen

- Objektorientering
  - Delegeringsteknikken
  - Delegering vs. arv