



PGR 101 Objektorientert Programmering 2
Vår 2017

Forelesning 14.1.17

(Stein Marthinsen – marste@westerdals.no)

PGR101

Gjennomføring

- 2 innleveringer (individuelle – som i høst)
- 1 skriftlig prøve (uten hjelpemidler)
100%
- Lærebok
Objects First with Java (som i høst)
Nettressurser og videoer

Vurdering

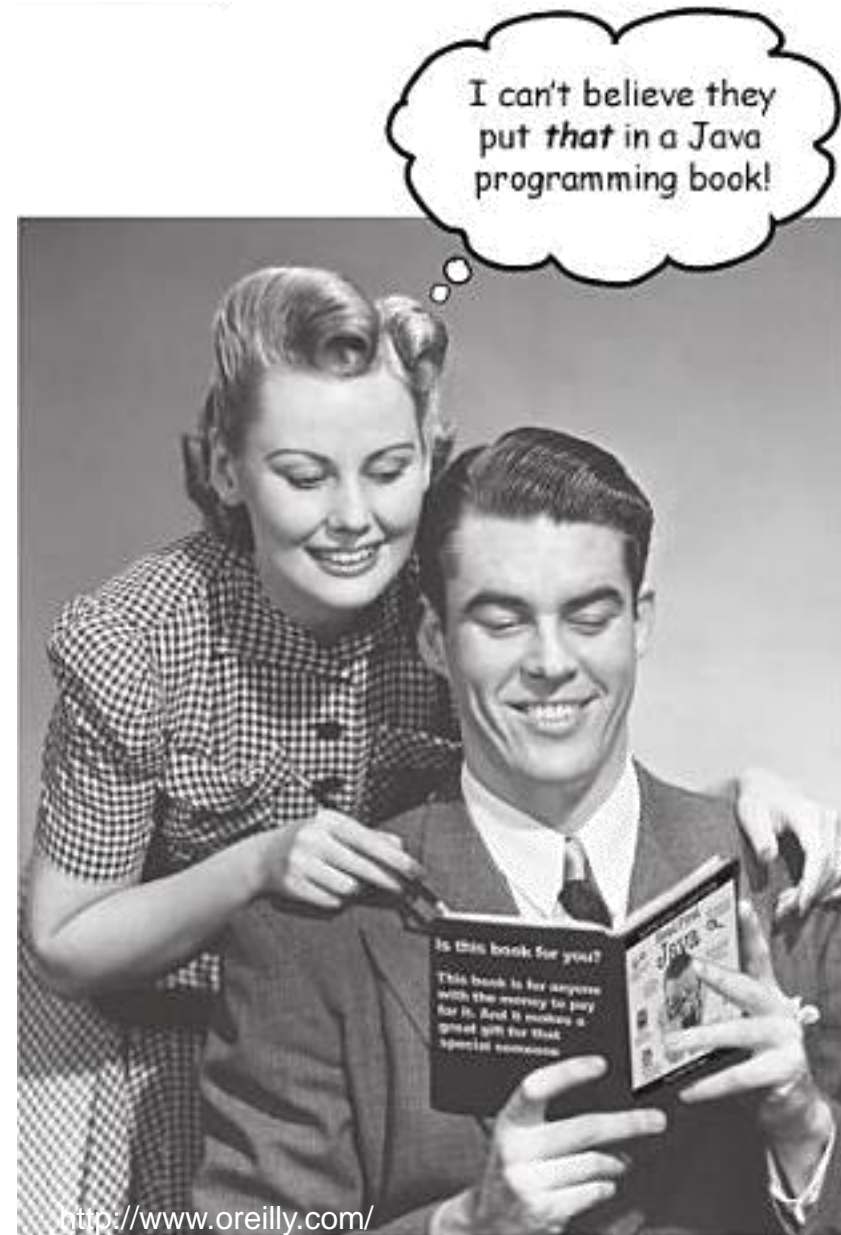
- Innleveringer og skriftlig prøve må bestås (som i høst)

TEMA
Arv
Arv
Arv
Arv
Abstrakte klasser/Interfaces
Abstrakte klasser/Interfaces
Grafiske grensesnitt
Grafiske grensesnitt
Filbehandling/Feilhåndtering
Filbehandling/Feilhåndtering
Analyse/Design
Oppsummering

(endringer kan komme!)

Dagens tema

1. Oppvarming
2. Project *westerdals*
last ned westerdals-v1
3. Forbedret struktur
ARV



Skriftlig prøve

```
public class Members {  
    private String[] members;  
    private int count;  
    private int max;  
  
    public Members (int max) {  
        count = 0;  
        this.max = max;  
        members = new String[max];  
    }  
  
    public void addMember(String name) {  
        members [count] = name;  
        count++;  
    }  
  
    public void printMembers() {  
        for (int i = 0; i < count; i++) {  
            System.out.println(members[i]);  
        }  
    }  
}
```



Skriftlig prøve

- c) Ved registrering av nye medlemmer kan det forsøkes å registrere flere medlemmer enn det er plass til.
Skriv en ny versjon av metoden `addMember` slik at dette ikke er mulig. Metoden skal returnere `false` hvis dette forsøkes gjort, `true` ellers.
- d) Skriv en metode `printMembers(char firstLetter)` som skriver ut alle medlemmene som har `firstLetter` (verdien av parameteren) som første bokstav i navnet.
- e) Skriv en klassen `Client` med metoden `clientMethod`. Metoden oppretter et objekt av klassen `Members`, registrerer noen medlemmer og kaller klassens metoder (opprinnelige og nye/endrede) på *passende* måte.
Hvis du ikke har svart på c) og/eller d), kan du allikevel anta at metodene beskrevet der er laget.

Skriftlig prøve

- c) Ved registrering av nye medlemmer kan det forsøkes å registrere flere medlemmer enn det er plass til.

Skriv en ny versjon av metoden `addMember` slik at dette ikke er mulig. Metoden skal returnere `false` hvis dette forsøkes gjort, `true` ellers.

```
public boolean addMember(String name) {  
    if (count < max) {  
        members [count] = name;  
        count++;  
        return true;  
    }  
    return false;  
}
```

Skriftlig prøve

- d) Skriv en metode `printMembers(char firstLetter)` som skriver ut alle medlemmene som har `firstLetter` (verdien av parameteren) som første bokstav i navnet.

```
public void printMembers(char firstLetter) {  
    for (int i = 0; i < count; i++) {  
        if (members[i].charAt(0) == firstLetter) {  
            System.out.println(members[i]);  
        }  
    }  
}
```


Skriftlig prøve

- e) Skriv en klasse `Client` med metoden `clientMethod`. Metoden oppretter et objekt av klassen `Members`, registrerer noen medlemmer og kaller klassens metoder (opprinnelige og nye/endrede) på *passende* måte.

Hvis du ikke har svart på c) og/eller d), kan du allikevel anta at metodene beskrevet der er laget.

```
public class Client {  
    public void clientMethod() {  
        Members members = new Members(3);  
        String [] someMembers = {"Hansi", "Gynter", "Albin", "Grufull", "Erika"};  
        for (int i = 0; i < someMembers.length; i++) {  
            if (members.addMember(someMembers[i])) {  
                System.out.println("Registrert!");  
            } else {  
                System.out.println("Ikke registrert!");  
            }  
        }  
  
        members.printMembers();  
        System.out.println();  
        members.printMembers('A');  
    }  
}
```

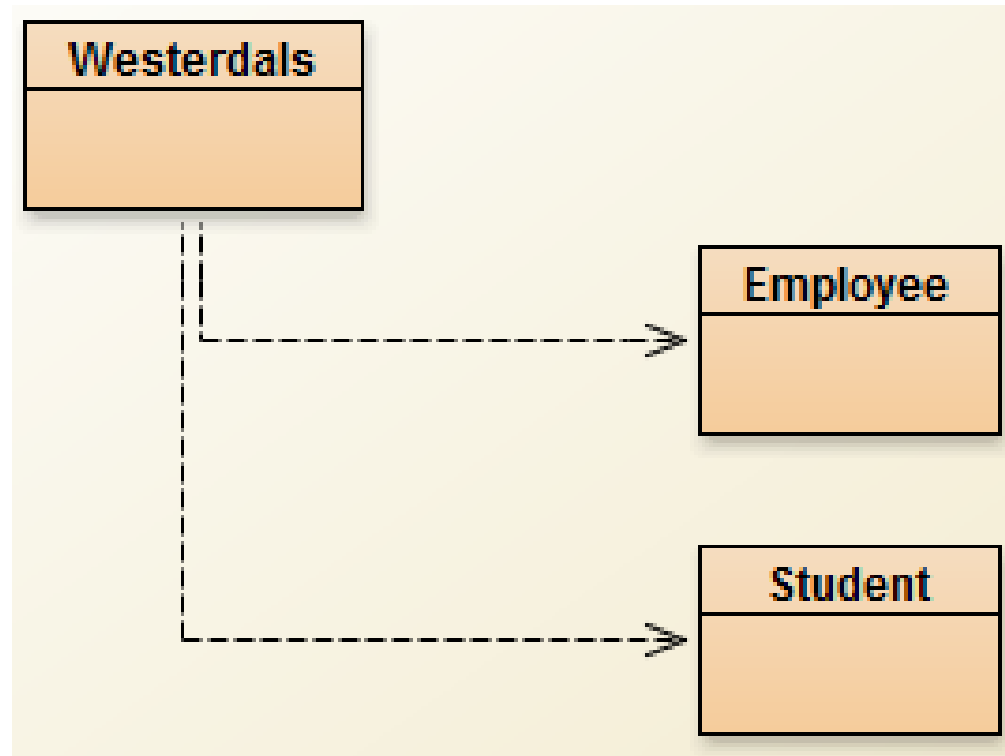
Project *westerdals-v1*

Last ned og åpne project *westerdals-v1*

Er en prototype for et arkiv over studenter/ansatte

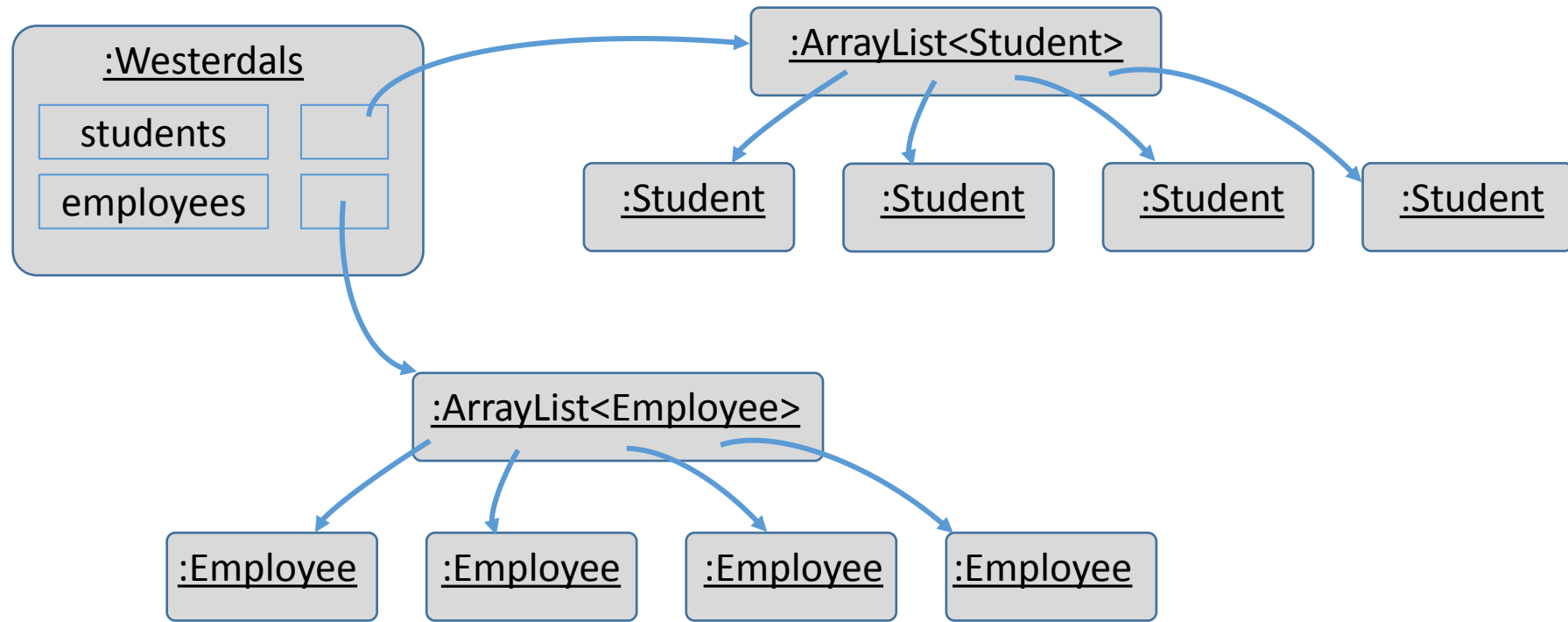
Bruker lister (`ArrayList`)

Klassediagram:



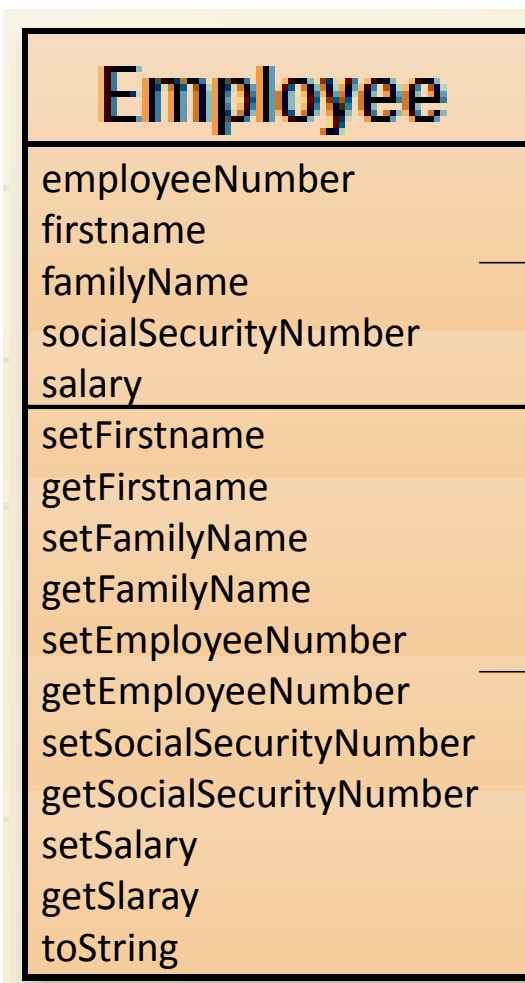
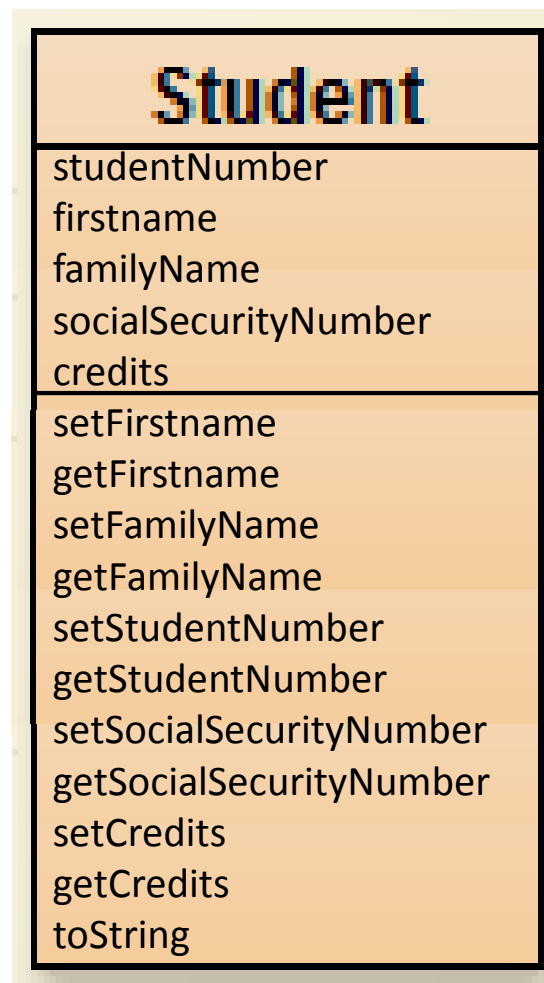
Project *westerdals-v1*

Objekter (eksempel):



Project *westerdals-v1*

Klassene:



øvre halvdel
viser *fields*
(attributter)

nedre halvdel
viser
metoder

Kildekode klassen Student (utdrag)

Klassene Student og Employee er standard (åpne editor og sjekk).

```
public class Student {  
    //Attributter  
    private String firstName;  
    private String familyName;  
    private String socialSecurityNumber;  
    private String studentNumber;  
    private int credits;  
  
    //Konstruktør  
    public Student(String studentNumber, String socialSecurityNumber,  
        String firstName, String familyName, int credits) {  
        setFirstName(firstName);  
        setFamilyName(familyName);  
        setSocialSecurityNumber(socialSecurityNumber);  
        setStudentNumber(studentNumber);  
        setCredits(credits);  
    }  
}
```

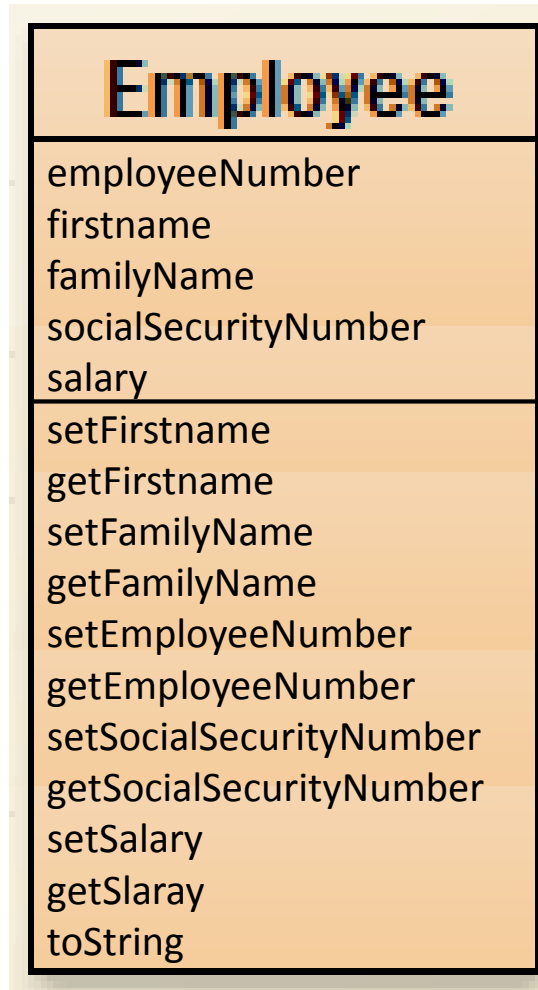
Student

studentNumber
firstname
familyName
socialSecurityNumber
credits

setFirstName
getFirstName
setFamilyName
getFamilyName
setStudentNumber
getStudentNumber
setSocialSecurityNumber
getSocialSecurityNumber
setCredits
getCredits
toString

Klassen Employee

Sjekk kildekoden



Kildekode klassen Westerdals (utdrag)

```
import java.util.ArrayList;
```

```
public class Westerdals {
```

```
    private ArrayList<Student> studentList;
```

```
    private ArrayList<Employee> employeeList;
```

```
    public Westerdals() {
```

```
        studentList = new ArrayList<Student>();
```

```
        employeeList = new ArrayList<Employee>();
```

```
    }
```

```
    public void addStudent(Student student) {
```

```
        studentList.add(student);
```

```
    }
```

Mangler

- To lister – én for hver type
- Søk etter og sletting av studenter/ansatte
- `showAll` har to løkker (student-info kommer alltid først).
- Data lagres ikke permanent

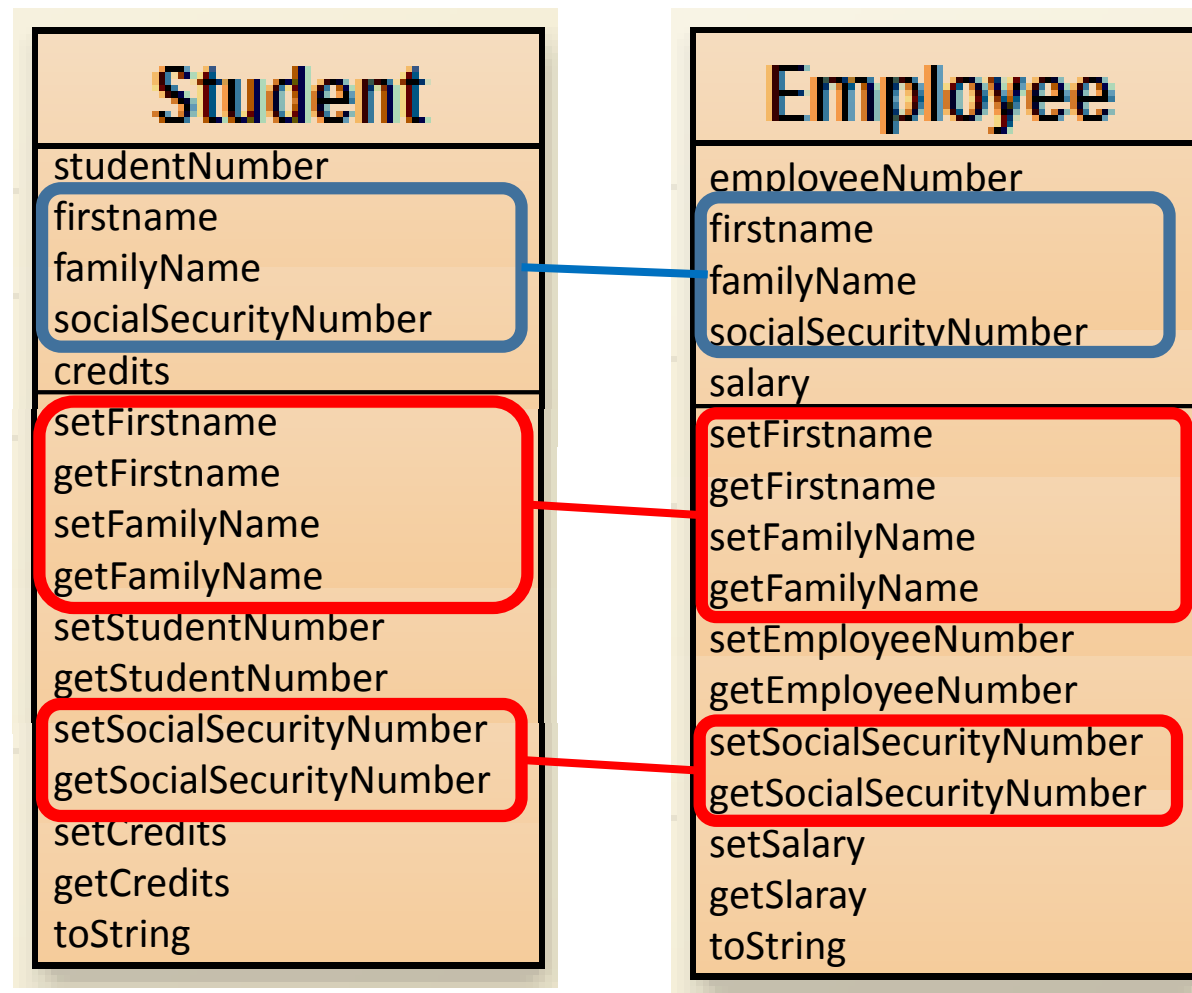
Oppgaver på øvingen

1. Med prosjektet `westerdals-v1` åpnet:
 - a) Opprett noen objekter av klassene `Student` og `Employee`.
 - b) Opprett et objekt av klassen `Westerdals` ("arkivet")
 - c) Sett objektene inn i arkiv-objektet.
 - d) Vis alt innhold i arkiv-objektet.
2. Opprett et `Student`-objekt.
 - a) Sett objektet inn i arkiv-objektet.
 - b) Vis innholdet i arkiv-objektet.
 - c) Endre `credits` i det nyopprettede `Student`-objektet.
 - d) Vis så innholdet i arkiv-objektet på nytt.
 - e) Vil `Student`-objektet nå vise at `credits` er endret?

Oppgaver på øvingen

3. Legg til metoder for å finne et student- eller ansatt-objekt i arkiv-objektet (hva er søkekriteriet?).
4. Legg til metoder for å fjerne et student- eller ansatt-objekt i arkiv-objektet.
5. Lag så en klasse **Main** med metoden **mainMethod**. Denne metoden skal opprette et "arkiv" (objekt av klassen **Westerdals**) og noen **Student-** og **Employee**-objekter og legge disse inn i "arkivet".
6. Metoden over skal kalle metoder i "arkiv"-objektet og sjekke at de fungerer som de skal.

Kritisk blikk



Det mest påfallende: gjentatt kode flere steder!

- Dårlig design
- Tynge å vedlikeholde/kan lede til feil ved vedlikehold

Kritisk blikk

Også i klassen `Westerdals`:
2 lister, 2 add-metoder

Metoden `showAll`:

```
public void showAll() {  
    for (Student s : studentList) {  
        System.out.println(s);  
    }  
  
    for (Employee e : employeeList) {  
        System.out.println(e);  
    }  
}
```

Kritisk blikk

Og hva hvis vi ønsker å utvide systemet med en ny type student/ansatt – f.eks.
`DeltidsStudent/DeltidsAnsatt`?

Mye av det samme måtte da gjøres på nytt!

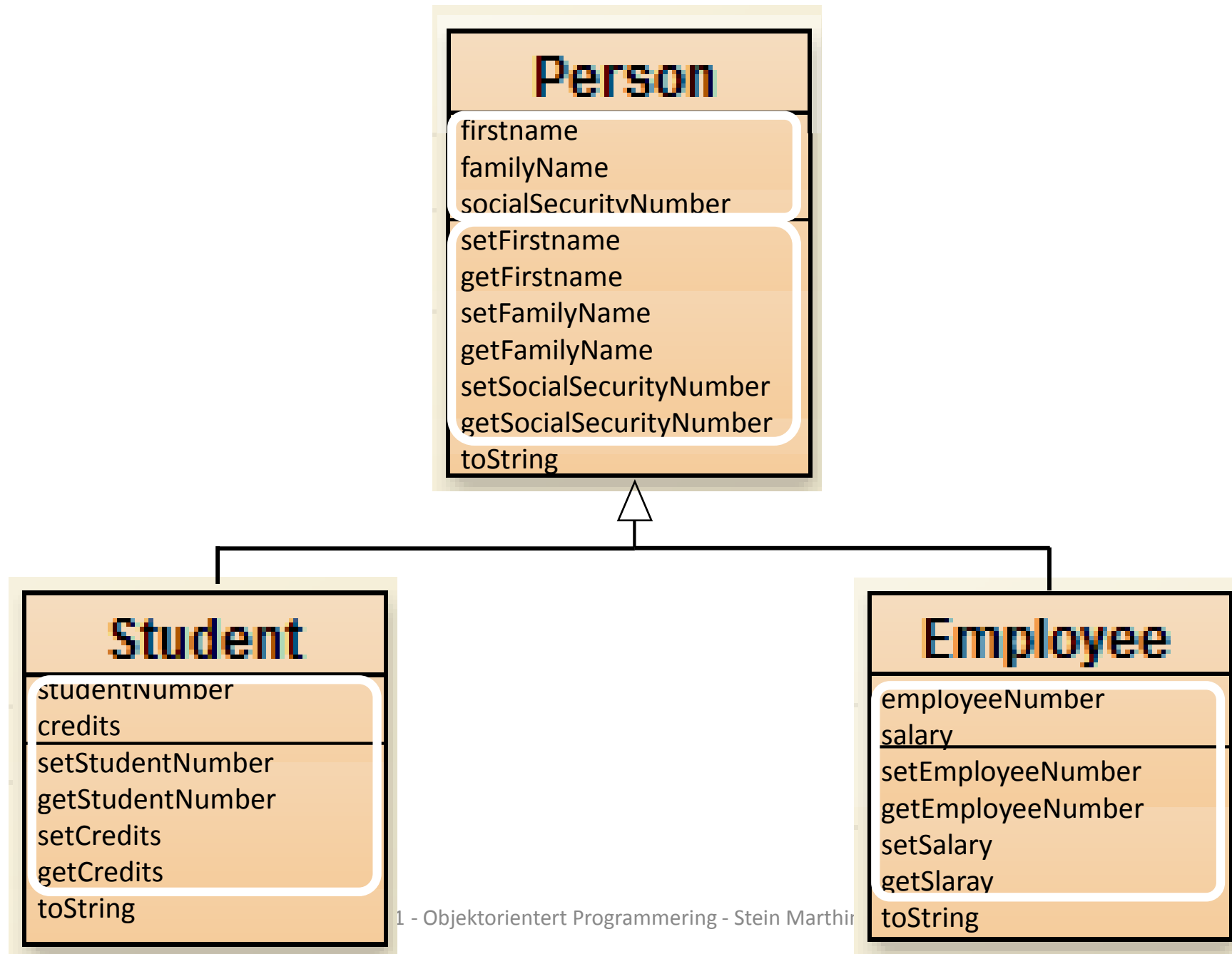
Nå er det slik:

Student
studentNumber firstname familyName socialSecurityNumber credits
setFirstname getFirstname setFamilyName getFamilyName setStudentNumber getStudentNumber setSocialSecurityNumber getSocialSecurityNumber setCredits getCredits toString

Employee
employeeNumber firstname familyName socialSecurityNumber salary
setFirstname getFirstname setFamilyName getFamilyName setEmployeeNumber getEmployeeNumber setSocialSecurityNumber getSocialSecurityNumber setSalary getSalary toString

Løsning ARV

superklasse



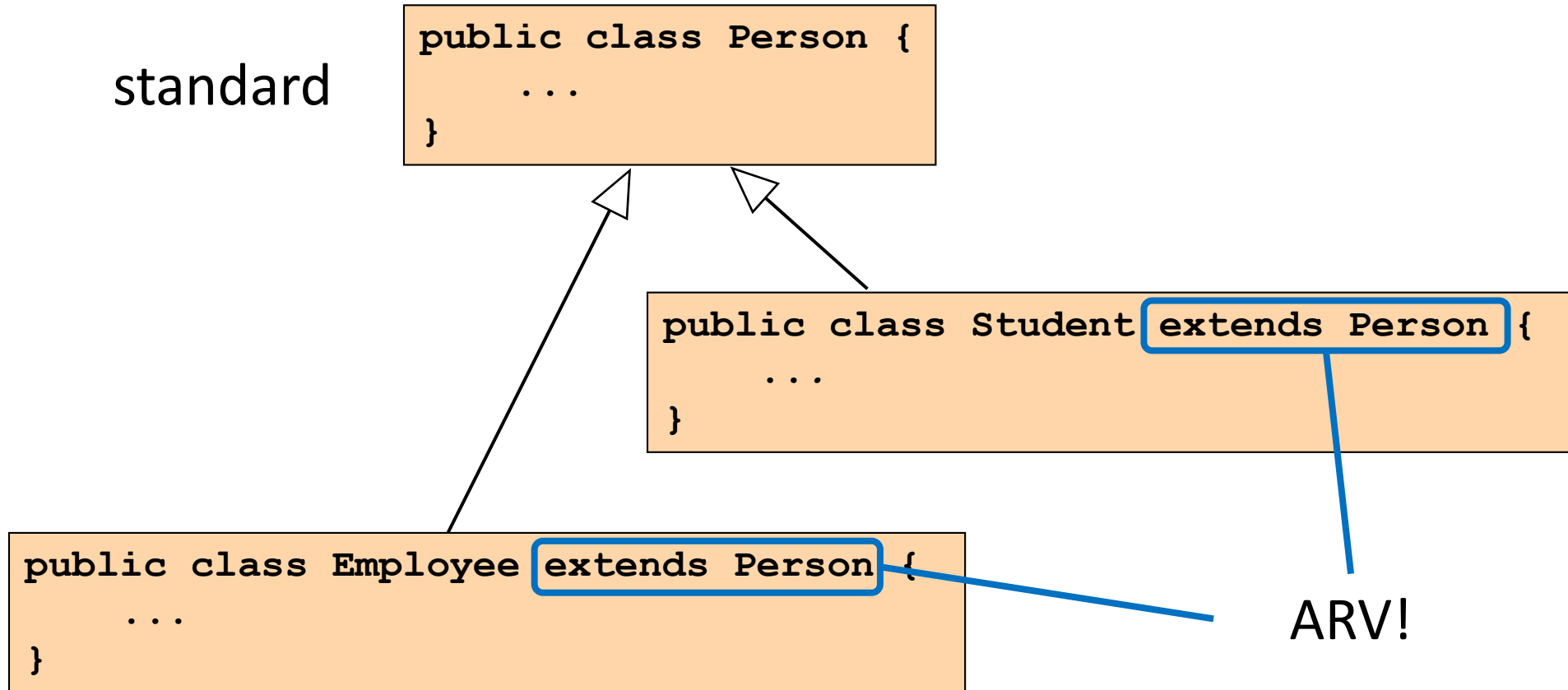
subklasser

Bruk av ARV

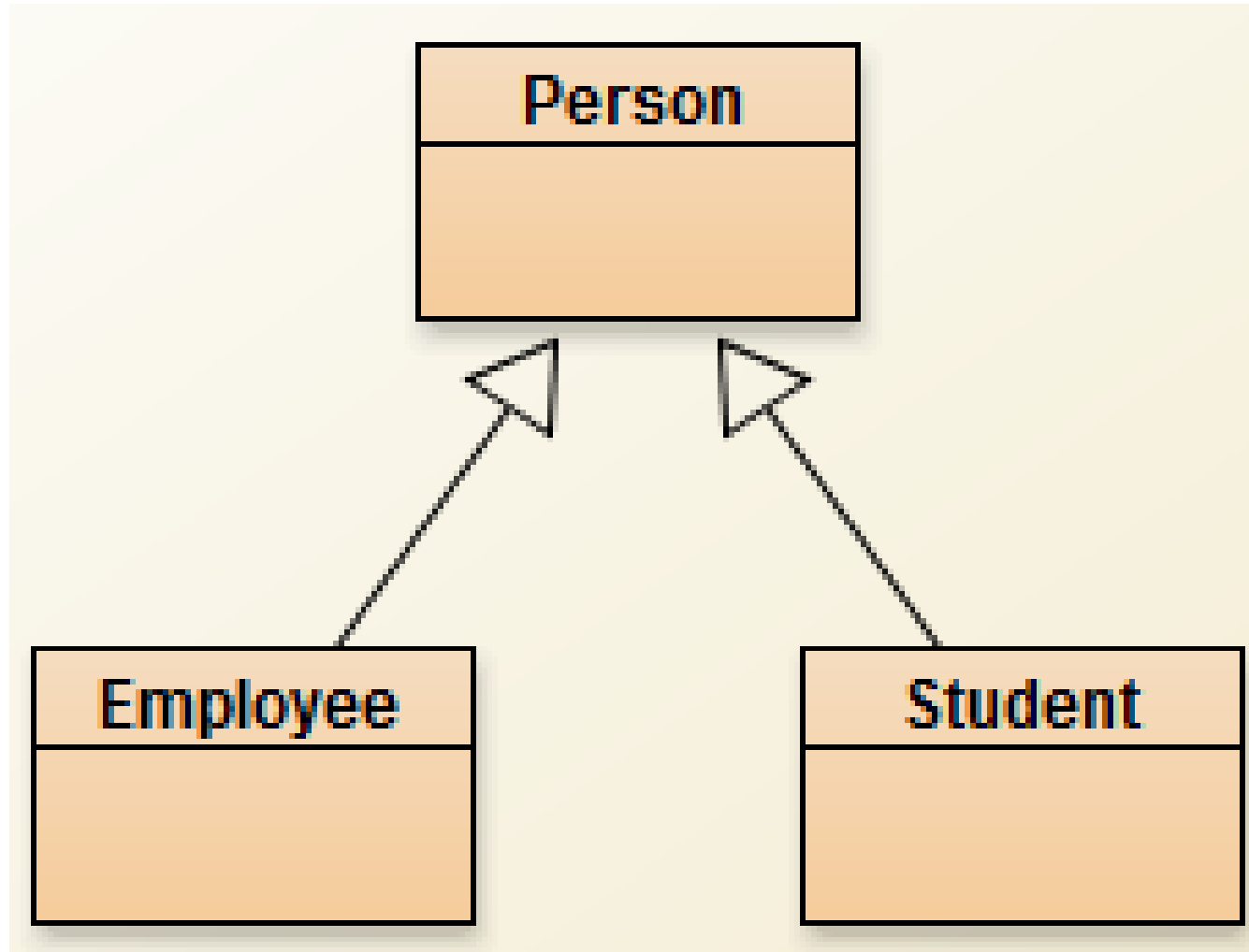
- definerer en **superklasse** : `Person`
- definerer **subklasser** for `Student` og `Employee`
- superklassen definerer *felles fields* (attributter) og *metoder*
- subklassen **arver** superklasse-*fields* og *metoder*
- subklassen legger til *spesielle fields* og *metoder*

ARV

standard



ARV!



Superklassen (er som en "vanlig" klasse)

```
public class Person {  
    private String firstName;  
    private String familyName;           // felles for subclassene  
    private String socialSecurityNumber;  
  
    // konstruktør og metoder utelatt.  
}
```

Subklassene

```
public class Student extends Person {  
    private String studentNumber; // bare for denne klassen  
    private int credits;           // bare for denne klassen  
  
    // konstruktør og metoder utelatt.  
}
```

```
public class Employee extends Person {  
    private String employeeNumber; // bare for denne klassen  
    private int salary;            // bare for denne klassen  
  
    // konstruktør og metoder utelatt.  
}
```

Når et `Student`-objekt skal opprettes, må alle opplysninger om en student oppgis:

```
Student s = new Student(studentNumber,  
                           firstName,  
                           familyName,  
                           socialSecurityNumber,  
                           credits);
```

Hva skal **`Student`**-konstruktøren gjøre med alle disse dataene?

`Student`-klassen har jo bare **`studentNumber`** og **`credits`** som fields...
(se forrige side)

Løsning: Konstruktørene samarbeider!

```
public class Student extends Person {  
    private String studentNumber;  
    private int credits;
```

alle data for et Student-objekt!

```
    public Student(String studentNumber, String socialSecurityNumber,  
                   String firstName, String familyName, int credits) {  
        super(firstName, familyName, socialSecurityNumber);  
        setStudentNumber(studentNumber);  
        setCredits(credits);  
    }
```

sender noe av dataene til
Person sin konstruktør
(**super**klassen)!

```
    // metoder utelatt  
}
```

Tilsvarende for Employee-klassen!

Kall på superklassens konstruktør

Må være første setning i subklasse-konstruktøren.

```
//Konstruktør  
public Student(String studentNumber, String socialSecurit  
    super(firstName, familyName, socialSecurityNumber);  
    setStudentNumber(studentNumber);  
    setCredits(credits);  
}
```

Konstruktøren i superklassen Person

```
public class Person {  
    private String firstName;  
    private String familyName;  
    private String socialSecurityNumber;  
  
    public Person(String firstName, String familyName, String socSecNum) {  
        setFirstName(firstName);  
        setFamilyName(familyName);  
        setSocialSecurityNumber(socSecNum);  
    }  
  
    // metoder utelatt  
}
```

tar imot data til sine fields

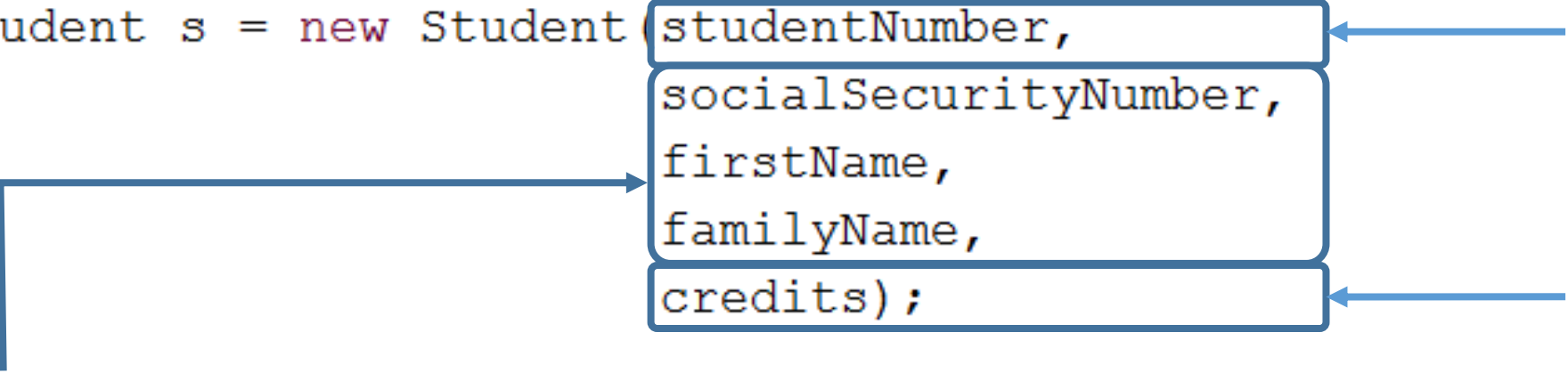
Altså:

Først kjøres `Person` sin konstruktør

Deretter kjøres (resten av) `Student` sin konstruktør

Oppretter objekt

```
String studentNumber = "123";  
String socialSecurityNumber = "5767";  
String firstName = "Stud";  
String familyName = "ent";  
int credits = 20;  
Student s = new Student(studentNumber,
```



```
socialSecurityNumber,  
firstName,  
familyName,  
credits);
```

Disse opplysningene sendes til konstruktøren i klassen `Person`, som setter de inn i passende fields der.

De to andre opplysningene settes inn i passende fields av konstruktøren i klassen `Student`.

Språkbruk – "er en"

```
public class Student extends Person
```

markerer at Student er en subklasse av klassen Person.

Vi sier da at

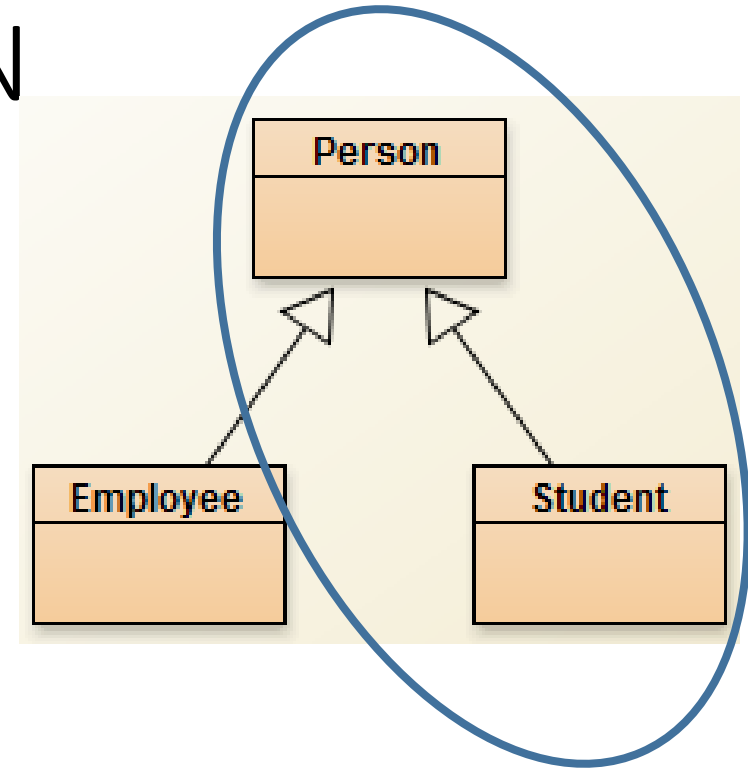
```
Student "er en" Person.
```

ARV representerer en "er en" relasjon.

Relasjoner – ARV/KOMPOSISJON

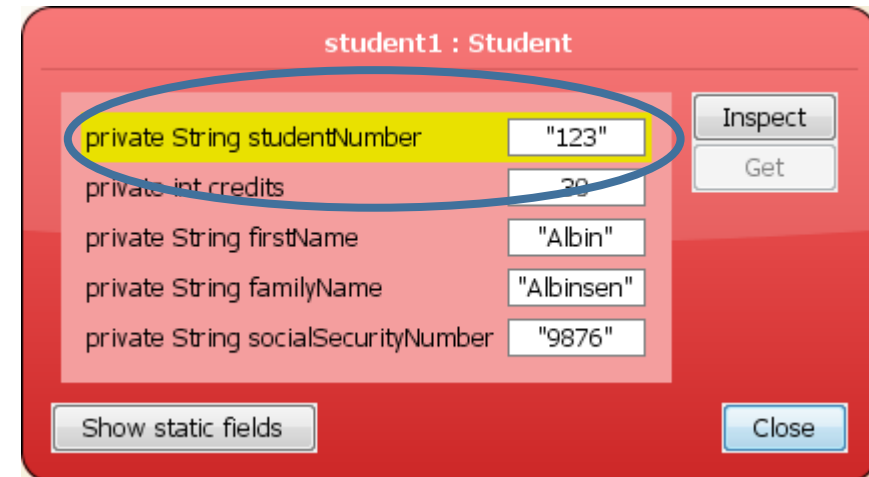
Student "er en" Person

ARV



Student "bruker en" String
(studentNumber)

KOMPOSISJON



Oppgave

Gitt følgende klasser:

Pedal

Bil

Sykkkel

Fremkomstmiddel

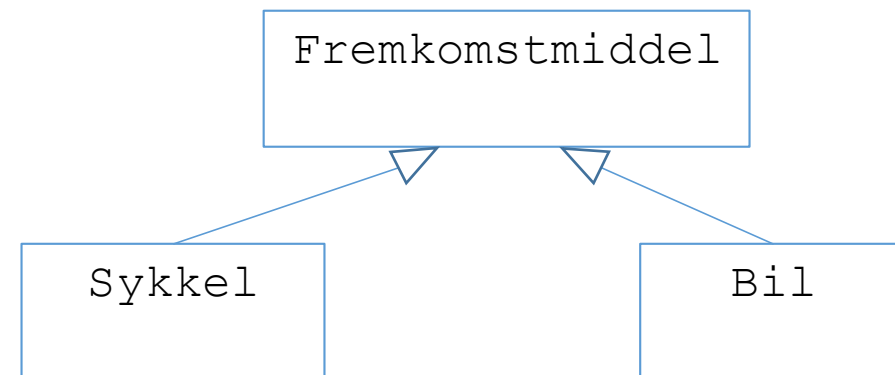
Motor

Hva representerer relasjonen "bruker en" (komposisjon) og hva representerer relasjonen "er en" (arv) i dette tilfellet?

En løsning

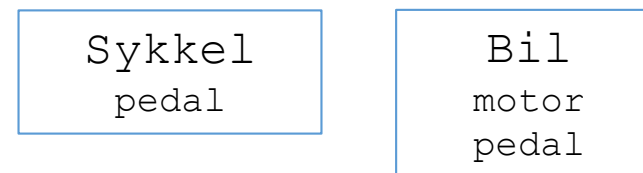
Bil **er et** Fremkomstmiddel
Sykkel **er et** Fremkomstmiddel

arv



Bil **bruker en** Motor
Bil **bruker en** Pedal
Sykkel **bruker en** Pedal

komposisjon



En løsning

```
public class Bil extends Fremkomstmiddel {  
    private Motor motor;  
    private Pedal [] pedaler;  
    ...  
}
```

En løsning

```
public class Sykkel extends Fremkomstmiddel {  
    private Pedal [] pedaler;  
    . . .  
}
```


Arv og tilgang

Fields og metoder deklarerert `private` i superklassen er **ikke direkte tilgjengelig** i subklassen

I klassen `Student` vil f.eks. en direkte henvisning til `firstName` gi feilmeldingen

firstName has private access in Person

Oppgaver på øvingen

7. Last ned og åpne prosjektet *westerdals-v2*:
8. Opprett et `Student`-objekt.
Sjekk hvilke metoder du kan kalle – egne og arvede
9. Opprett en metode `showFirstName` i klassen `Student`. Metoden skal skrive ut "Studentens fornavn er: " og så studentens fornavn.