Pragmatisk programmering

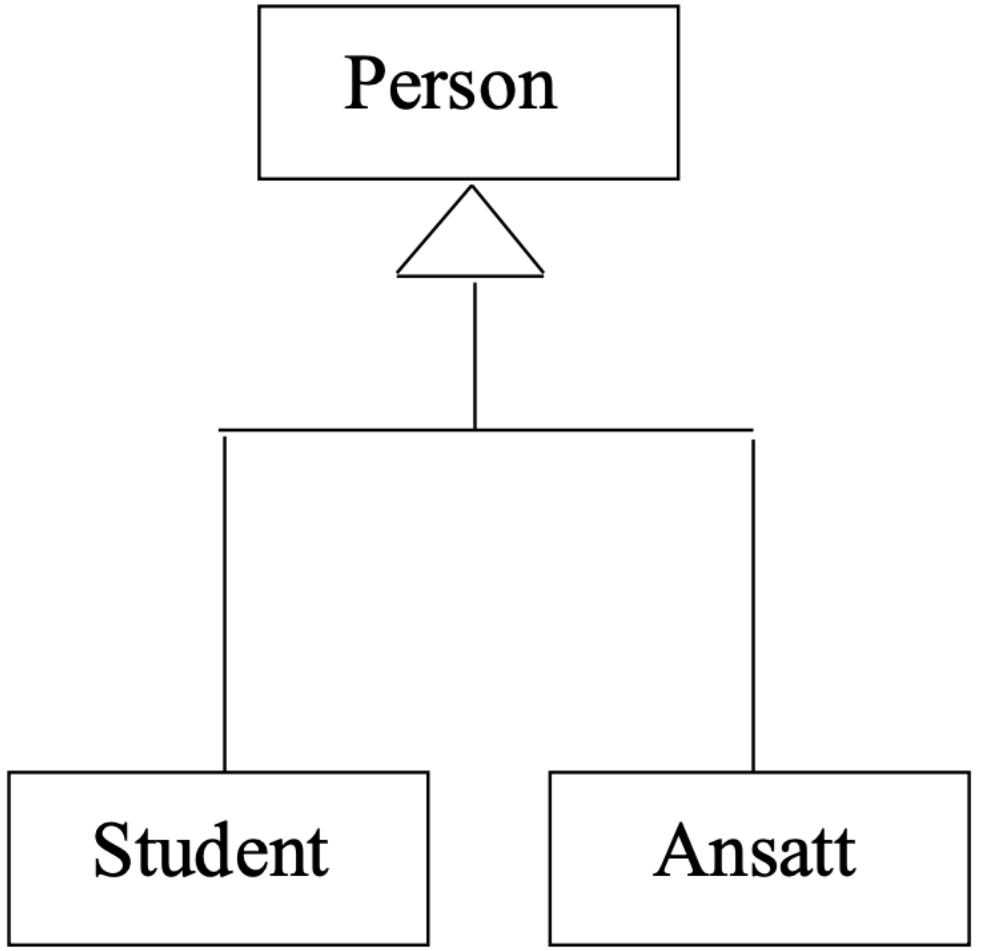
Arv og subklasser

Tema for uken

- Klasser og subklasser
  - Nytt nøkkelord: `extends`
- Referanser og typekonvertering
- Abstrakte klasser
- HashMap
  - Samme som Pythons `dict`
- Andre ønsker/behov?
  - Behov for å bruke mer tid på forrige ukes pensum?
  - (dette tar forrang!)

Merk til eksamen!

- Må kunne tegne klassehierarki
- Til midtveis: Lær dere referanser og typekonvertering (mange synes dette er krevende!)



Extends

- En subklasse er en utvidelse av en annen klasse
  - Den har alle de samme instansvariablene og metodene som superklassen
  - Men har muligens også flere variabler og metoder, som superklassen *ikke* har

- Nøkkelord:
  - `extends`
  - `super()`

```java
class Dyr {
    private String navn;

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

    public String hentNavn() {
        return navn;
    }
}

class Hund extends Dyr {
    private String eier;

    public Hund(String navn, String eier) {
        super(navn);
        this.eier = eier;
    }

    public String hentEier() {
        return eier;
    }
}
```

Referanser og typekonvertering

```
interface G { }
class A { }
class B extends A { }
class C extends B { }
class D extends B implements G { }
class E extends A implements G { }

class HovedSubInt {
  public static void main (String[] args) {
     A a = new B();
     B b = new C();
     C c = new C();
     D d = new D();
     E e = new E();
     G ii = d;
     G jj = e;
     // Her skal det stå én linje
  }
}
```

Referanser og typekonvertering

```java
interface Egglegger {
    int antallEgg ();
}

class Hare {
    final String navn;
    Hare (String n) { navn = n; }
}

class Påskehare extends Hare implements Egglegger {
    int antall;
    Påskehare (String n, int ant) { super(n);  antall = ant; }

    @Override public int antallEgg () { return antall; }
}

class Høne implements Egglegger {
    @Override public int antallEgg () { return 1; }
}

class BrukHare {
    public static void main (String[] args) {
        Hare skogshare = new Hare("Bjørk");
        Hare sesonghare = new Påskehare("Påsan", 999);
        Høne burhøne = new Høne();

        // Hvilke av disse 10 linjene er lovlige som neste linje?
        Påskehare h1 = sesonghare;
        Egglegger h2 = sesonghare;
        Egglegger h3 = burhøne;
        Egglegger h4 = (Egglegger)skogshare;
        Påskehare h5 = (Påskehare)sesonghare;
        Egglegger h6 = (Hare)sesonghare;
        Hare      h7 = sesonghare;
        Egglegger h8 = (Egglegger)sesonghare;
        Egglegger h9 = skogshare;
        Påskehare h10 = (Påskehare)skogshare;
    }
}
```

	OK	Feil
Påskehare h1 = sesonghare;	○	○
Egglegger h2 = sesonghare;	○	○
Egglegger h3 = burhøne;	○	○
Egglegger h4 = (Egglegger)skogshare;	○	○
Påskehare h5 = (Påskehare)sesonghare;	○	○
Egglegger h6 = (Hare)sesonghare;	○	○
Hare h7 = sesonghare;	○	○
Egglegger h8 = (Egglegger)sesonghare;	○	○
Egglegger h9 = skogshare;	○	○
Påskehare h10 = (Påskehare)skogshare;	○	○

Abstrakte klasser

- Kan ikke instansieres
  - Instansieres via subklasser
- Nøkkelord: `abstract`
- Kan ha både abstrakte og konkrete metoder

```java
abstract class Form {
    abstract double areal();
}

class Sirkel extends Form {
    private double radius;

    public Sirkel(double radius) {
        this.radius = radius;
    }

    public double areal() {
        return 3.14 * radius * radius;
    }
}
```

HashMap

- Som Pythons dict, men må importeres
- Tilbyr flere metoder:
  - `put(nøkkel, verdi)` – legger nytt nøkkel/verdi-par i ordboken
  - `get(nøkkel)` – returnerer verdien til nøkkelen
  - `remove(nøkkel)` – fjerner nøkkel/verdi-paret fra ordboken

```java
import java.util.HashMap;

class Main {
    public static void main(String[] args) {
        HashMap<String, Integer> antallFrukt = new HashMap<>();
        antallFrukt.put("kiwi", 15);

        System.out.println(antallFrukt.get("kiwi")); // skriver ut 15
    }
}
```