

OOP-II Arv

- Specialisering av en basklass
- UML-form
- En slags återanvändning av kod och datastrukturer
- Åtkomst mellan basklass/subklass
- Multipelt arv
- Abstrakta basklasser

2000-11-08

Thomas Johansson

1

Specialisering

- Nedåt i hierarkin - specialisering
- Uppåt - generalisering
- Exempel
Fordon - Motorfordon - Bil

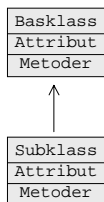
2000-11-08

Thomas Johansson

2

UML-form

- Basklassen (superklassen) överst (oftast)
- Pilen mot basklassen



2000-11-08

Thomas Johansson

3

Återanvändning

- En subclass kan *ärva* både attribut och metoder från sin basklass - då behöver den inte definiera dem
- En subclass kan *lägga till* attribut och metoder
- En subclass kan *definiera om* attribut och metoder
 - En Fordonsklass kan ha attributet *hastighet*
 - En subclass Bil ärver automatiskt hastighet, och kan lägga till t.ex. *antalPassagerare*
 - Om basklassen har metoden *tanka* så kan subclassen byta ut den mot en metod som tankar något speciellt (men den har samma signatur - omdefinition)
 - Förväxla inte med överlagring - olika signatur

2000-11-08

Thomas Johansson

4

Arv i C++

```
class Fordon
{
private:
    int bransleMangd;
public:
    void tanka(int liter);
    int getBransle();
};

class Bil:public Fordon
{
private:
    int antalPass;
};
```

2000-11-08

Thomas Johansson

5

Omdefinition

```
class Fordon
{
private:
    int bransleMangd;
public:
    void tanka(int liter);
    int getBransle();
};

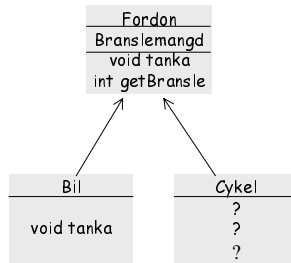
class Bil:public Fordon
{
private:
    int antalPass;
public:
    void tanka(int liter);
};
```

2000-11-08

Thomas Johansson

6

Problem



2000-11-08

Thomas Johansson

7

Mer om arv

- När ej basklassens privata data om de är private

```
double Bil::berakna_rackvidd()
{
    return bransleMangd / 0.9; // Går ej!
}
```

- Men kan anropa basklassens operationer

```
double Bil::berakna_rackvidd()
{
    return Fordon::getBransle() / 0.9; //Ok
}
```

2000-11-08

Thomas Johansson

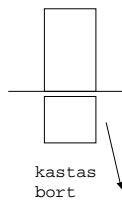
8

Typkonverteringar

- Konverteras om nödvändigt till basklass, "slicing"

```
Fordon f;
Bil b;

f = b; // ok, 'slicing'
b = f; // ej ok !
// Odefinierat utrymme
```

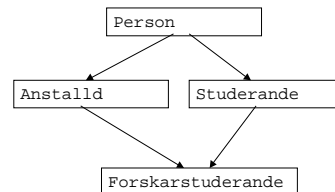


2000-11-08

Thomas Johansson

9

Multipelt arv



```
class Forskarstuderande : public
Anstalld, public Studeraende
{
};
```

2000-11-08

Thomas Johansson

10

Problem med multipelt arv

- Omdiskuterat behov
- Namnkonflikter
- Dubbla data (löses med virtuell arv)
- Undvik!

2000-11-08

Thomas Johansson

11

Abstrakt basklass

- Fångar gemensamma egenskaper
- Påtvingar ett gränssnitt
- Med äkta virtuell funktion:

```
class Figur
{
public:
    virtual double getArea() = 0;
};
```

- Eller med protected konstruktör

2000-11-08

Thomas Johansson

12