

OOP-I Klassbegreppet och C++

- OOP
- UML
- Klasser och objekt i C++
- Uppdelning i filer
- Attribut och metoder
- Inkapsling - åtkomst
- Klassattribut - objektattribut

2000-11-08

Thomas Johansson Datavetenskap

1

OOP- Objekt-orienterad programmering

- Att använda ett objekt-orienterat språk för att implementera en OO-design.
- Stöd för:
 - Inkapsling
 - Arv
 - Dynamisk bindning

2000-11-08

Thomas Johansson Datavetenskap

2

UML

- Unified Modeling Language
- Three amigos
 - Grady Booch
 - James Rumbaugh
 - Ivar Jacobson
- Gav upp deras egna metoder för att uppnå standardisering

2000-11-08

Thomas Johansson Datavetenskap

3

Användbara modeller är

- riktiga (accurate)
 - beskriver systemet korrekt
- konsekventa (consistent)
 - inga konflikter
- enkla att förmedla till andra
- enkla att ändra
- förståeliga
 - så enkla som möjligt men inte enklare

2000-11-08

Thomas Johansson Datavetenskap

4

De största problemen med dagens mjukvaruutveckling

- Många projekt börjar programmera för tidigt
- Läger ner för mycket energi på kodningen
- Ledningen saknar förståelse för mjukvaruutv
- Programmerare känner sig mer säkra när de programmerar än när de bygger abstrakta modeller

2000-11-08

Thomas Johansson Datavetenskap

5

Målen med UML

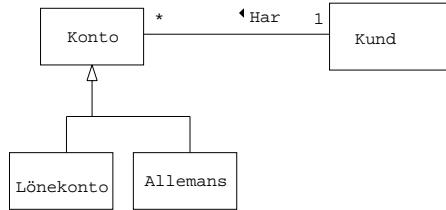
- Modellera system (inte bara mjukvaran)
- Koppla ihop koncept
- Hantera skalning
- Skapa ett språk som kan användas av både människor och datorer

2000-11-08

Thomas Johansson Datavetenskap

6

Klassdiagram

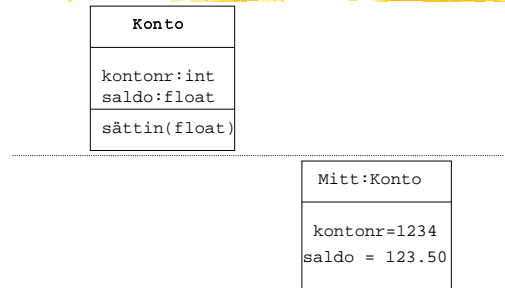


2000-11-08

Thomas Johansson Datavetenskap

7

Objektdiagram

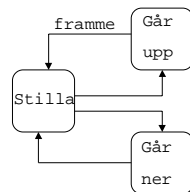


2000-11-08

Thomas Johansson Datavetenskap

8

Tillståndsdigram

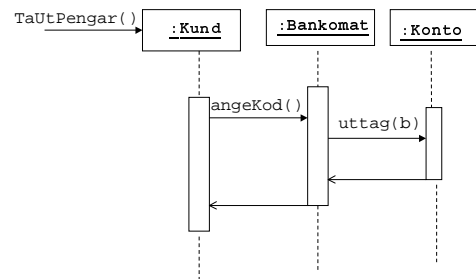


2000-11-08

Thomas Johansson Datavetenskap

9

Sekvensdiagram

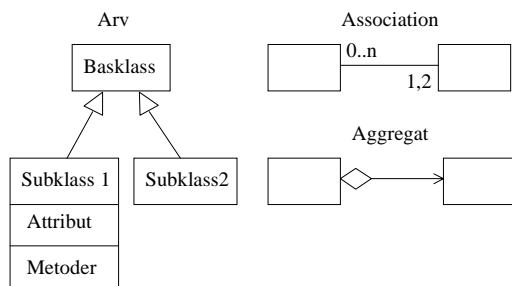


2000-11-08

Thomas Johansson Datavetenskap

10

UML i korthet



2000-11-08

Thomas Johansson Datavetenskap

11

Klasser och objekt i C++

- Klasser deklaras i en typdeklaration, precis som structs i c
- Ett eller flera objekt skapas med klassen som mall
 - Det finns två sätt att skapa objekt:
 - På heapen (med new)
 - Som lokala (automatiska) variabler

2000-11-08

Thomas Johansson Datavetenskap

12

En klass - två filer

- Ett bra sätt att dela upp sitt system
- Många filer blir det - bra att ha någon struktur på filsystemet
- Varje klass har två filer
 - Headerfil (.h) med deklarationer (interfacet)
 - Klassfilen (.cpp) med definitionen (implementationen)
- Det går att slå ihop dem, men det försvårar användningen av klassen, och minskar inkapslingen

2000-11-08

Thomas Johansson Datavetenskap

13

Exempel

- Headerfil (Interface)
Bara deklarationer av metoder och (tyvärr) attribut

```
class Konto
{
public:
    void sattIn(int kr); // metod
private:
    int saldo;          // attribut
};
```

2000-11-08

Thomas Johansson Datavetenskap

14

Forts.

- Klassfil (Implementation)
Definitioner av metoder

```
void Konto::sattIn(int kr)
{
    konto = konto + kr;
}
```

2000-11-08

Thomas Johansson Datavetenskap

15

Kombinerat interface/impl

- Kan användas för enstaka klasser, inget att rekommendera

```
class Konto
{
public:
    void sattIn(int kr) // metod
    {
        konto = konto + kr;
    }
private:
    int saldo;          // attribut
};
```

2000-11-08

Thomas Johansson Datavetenskap

16

Flera klasser

- Endast .h-filerna behöver 'synas'

<pre>class Personbil...</pre>	<pre>class Passagerare...</pre>
<pre>#include "Personbil.h" #include "Passagerare.h" Personbil::gasa()... Personbil::bromsa() ...</pre>	<pre>#include "Personbil.h" #include "Passagerare.h" Passagerare::metod1()... Passagerare::metod2()...</pre>

2000-11-08

Thomas Johansson Datavetenskap

17

Åtkomstnivåer

- Tre nivåer av inkapsling
 - Public - åtkomst för alla
 - Protected - åtkomst för subclasser
 - Private - Åtkomst endast inom klassen

```
class Bil
{
public:
    void gasa();
protected:
    int hastighet;
private:
    int modell;
};
```

2000-11-08

Thomas Johansson Datavetenskap

18

Åtkomst forts.

- Både attribut och metoder kan ha alla nivåer
- En nivå (ex private) gäller tills nästa anges
- Default är **private**
- Attribut bör vara private
 - Använd åtkomstmetoder för att inte bryta inkapslingen
- Metoder som endast används inom klassen bör vara private, eller protected om det kan tänkas att subclasser kan komma att omdefiniera dem (svårt val ibland)

2000-11-08

Thomas Johansson Datavetenskap

19

Klassattribut

- 'Vanliga' attribut hör till ett objekt, dvs flera objekt -> flera uppsättningar attribut med olika värden
- Klassattribut är gemensamma för alla objekt av en klass - 'hör till klassen'
- Nyckelordet **static** anger klassattribut/metod
- Klassmetoder kan bara modifiera klassattribut (vilket värde avses annars ?)

2000-11-08

Thomas Johansson Datavetenskap

20

Exempel

- Räkna för antalet skapade objekt

```
class Bil
{
public:
    Bil()
    {
        antal++;
    }
    int getAntal(){ return antal; }
private:
    static antal;
}
```

2000-11-08

Thomas Johansson Datavetenskap

21