Repetition

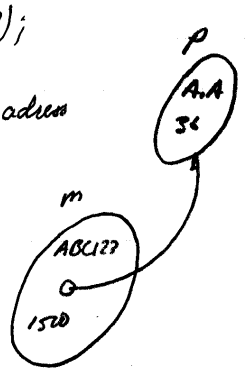
Klass, objekt, medlem, konstruktor, destruktor, relationa som består av, känner till och är en, dynamisk bindning, överlagring, friend, operatoren

(Ex) Klassen Motorfordon MF

```
// Specifikation — Mf.h
#include "Person.h"
class MF
{
    private:
        char reg[7];
        PERSON *agare; // MF känner till sin ägare!
        int skatt;
    public:
        //Konstruktor
        MF(char *reg, PERSON *agare, int skatt);
        // Destruktor
        ~MF();
        // Utskrift
        void skriv();

};
```

```
//Implementation - Mf.cpp
#include <cstring>
#include "Mf.h"
#include <iostream>

MF::MF(char *reg, PERSON *agare, int skatt)
{
   strcpy(this->reg, reg);
   this->agare = agare;
   this->skatt = skatt;
}

MF::~MF()
{
    agare = NULL;
}

 void MF::skriv()
 {
     cout << "Regnr: " << reg << endl;
     agare->skriv(); // cout << *agare om <<operator!
     cout << "Skatt: " << skatt << " kr" << endl;
 }
```

```
//Mfmain.cpp

#include "Mf.h"

void main()
{
    PERSON p("A.A", 36); //Måste definieras först!
    MF m("ABC123", &p, 1500);
                        ↑
                        //OBS! adress

    m.skriv()
}
```

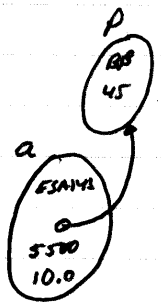


(Ex) En lastbil är ett motorfond med lastkapacitet

```
//Lb.h

#include "Mf.h"

class LB : public MF   //OBS! Arv
{
    private:
        float last;  //Lastkapacitet i ton
    public:
        LB(char *reg, PERSON *agare, int tekatt, float last)
        void skriv();
};
```

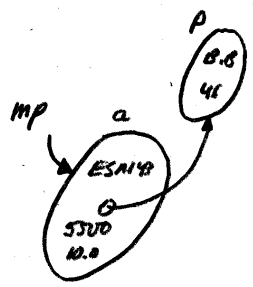
```
//Lb.cpp
#include <iostream>
#include "Lb.h"

LB::LB(char *reg, PERSON *agare, int skatt, float last)
        : MF(reg, agare, skatt)
{
   this->last = last;
}

void LB::skriv()
{
     MF::skriv(); //OBS! Anropa av basklassen skriv
     cout << "Lastkapacitet: " << last << endl;
}
```

```
//Lbmain.cpp

#include "Lb.h"

void main()
{
     PERSON p("B.B", 45);
     LB a("ESA143", &p, 5500, 10.0);

     a.skriv();
}
```



Att anrop!

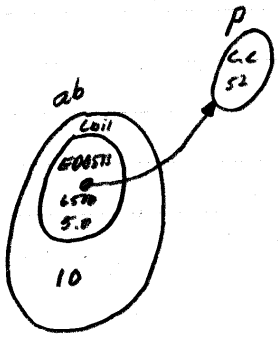
```
MF *mp;

mp = &a;
mp->skriv();
  // Om skriv markeras virtual i MF-klassen (dynamisk bindning)
  // fås korrekt utskrift av all data
```



(Ex) En lastbuss består av en lastbil och ett antal sittplatser.

```
class LBUS
{
    private:
        LB lbil; // Består av
        int platser;
    public:
        LBUS(char *reg, PERSON vagare, int tskatt
             float last, int platser);

        void skriv();
};
```

```
// Lbus.cpp

LBUS::LBUS (char *reg, PERSON *agare, int skatt, -----)
          : Lbil(reg, agare, skatt, Last)
                                          //OBS!
{
   this->platser = platser;
}

void  LBUS:: skriv()
{
      Lbil.skriv();
      cout << "Antal sittplatser : " << platser << endl;
}
!
void main()
{
    PERSON p("C.C", 52);
    LBUS ab("EO0513", &p, 6500, 5.0, 10);

    ab.skriv();
}
Alt! Dynamisk allokering
     !
    PERSON p("O.O", 32);
    LBUS *abp;

    abp= new LBUS ("QZA111", &p, 4500, 3.0, 5);
    abp->skriv();
    delete abp;
     !
```



⑥

(Ex) En tipsrad

```
//Tr.h

class TR
{
   private:
      char rad[14];
   public:
      TR(char *rad = "");
      void slumpa();
      int antal_ratt(TR ratt);
      friend ostream &operator<<(ostream &ut, TR t);
      friend istream &operator>>(istream &in, TR &t);
};

//Tr.cpp

TR::TR(char *rad)
{
   strcpy(this->rad, rad);
}

void TR::slumpa()
{
     int slump;
```

```
    randomize();
    for(int i=0; i<13; i++)
    {
        slump = random(3);
        if (slump == 0)
            rad[i] = 'x';
        else if (slump == 1);
            rad[i] = '1';
        else
            rad[i] = '2';
    }
}

int TR:: antal_ratt(TR ratt)
{
    int sum = 0;

    for(int i=0; i<13; i++)
    {
        if ( rad[i] == ratt.rad[i])
            sum++;
    }
    return sum;
}
```

7) Templates och Exceptions

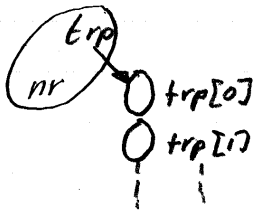
Template – mall för olika typer (klasser)

(Ex) Generell sorteringsfunktion

```
//Sort.h

template <class T>
void byt(T &a, T &b)
{
    T temp;

    temp=a;
    a = b;
    b = temp;
}

template <class T>
void unsort(T v[], int nr)
{
    for (int i=0; i<nr-1; i++)
    {
        for(int k=i+1; k<nr; k++)
        {
            if (v[k] < v[i])        //< märk funn!
                byt(v[i], v[k]);
        }
    }
}
```

(Ex) Ett tips består av ett antal tipsrader.

```
//Tips.h
  :
class TIPS
{
    private:
      TR *trp;
      int nr;
    public:
      TIPS(int nr=1);
     ~TIPS();
      TIPS(const TIPS &ts);
      const TIPS &operator=(const TIPS &ts);
      void slumpa();
      void ratta(TR ratt);
};

//Tips.cpp
  :
TIPS::TIPS(int nr)
{
     trp = new TR[nr];
     this->nr = nr;
}
```



```
void TIPS::slumpa()
{
   for (int i=0; i<nr; i++)
   {
      trp[i].slumpa();
   }
}

void TIPS::ratta(TR ratt)
{
   for (int i=0; i<nr; i++)
   {
      cout << trp[i];
      cout << "Antal rätt: " << trp[i].antal_ratt(ratt);
   }
}

void main()
{
   TIPS t(10);
   TR tr;

   t.slumpa();
   tr.slumpa();

   t.ratta(tr);
}
```