UMEÅ UNIVERSITET
Institutionen för datavetenskap

2000-05-28

Thomas Johansson
Kalle Prorok

Tentamen i Objektorienterad programmering för ingenjörer (TDBB09) 2000-05-31, kl 09.00 - 15.00

Lokal: Skrivsal 8, Östra Paviljongerna, Ålidbacken 23

Hjälpmedel: inga (förutom penna, suddgummi och matsäck)

Kursutvärdering: Sist bland papperen finns en kursutvärderingsblankett. Ta med den hem och fyll i den. Lämna in den till Thomas när du hämtar ut den rättade tentan.

Börja varje uppgift på ett nytt blad.
Använd bara en sida av papperet.
Skriv namn på varje blad.
Lägg uppgifterna i rätt ordning.
Observera att uppgifterna inte är numrerade efter svårighetsgrad och kom ihåg att även delvis lösta uppgifter kan ge poäng.
Kommentera noga all källkod så att vi kan se hur du har tänkt.
Skriv tydligt.

Maxpoäng är 40, fördelade på 6 frågor.

Betygsgränser:

5	32p
4	26p
3	20p

Fråga om det är något som är oklart ! Vi kommer till skrivsalen ca **10.30** och **12.00**.

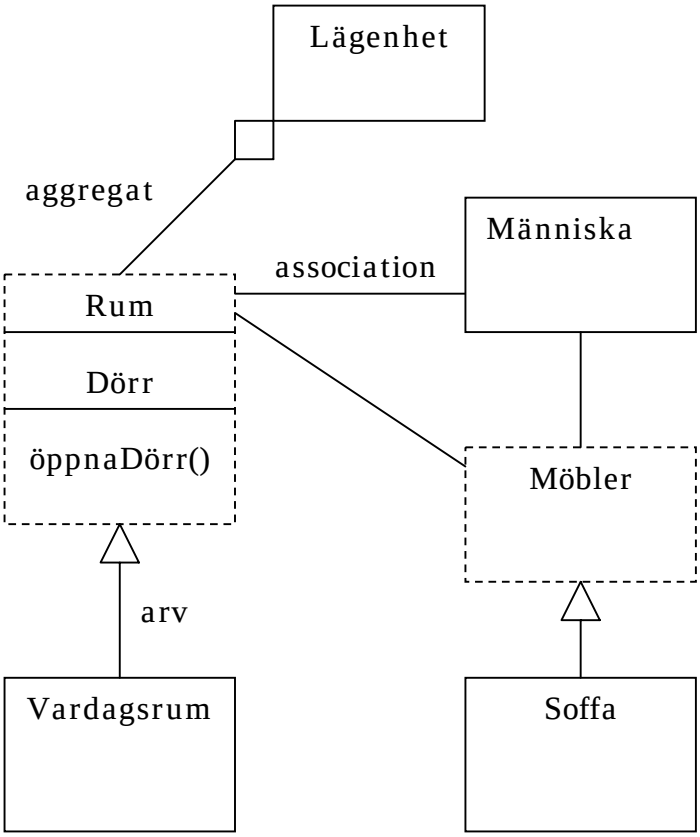
Lycka till och trevlig sommar !

Thomas Johansson
tel 12 71 16

Kalle Prorok
tel 12 99 96

UPPGIFT 1 – OO (9p)

Kalle Kunskapsförmedlare har just köpt en fin lägenhet. Hjälp honom med att **konstruera ett klassdiagram** rörande delar av lägenheten och lite allmänt om rumstyper! Minst 6 klasser, varav en abstrakt, och Arv, Association, Aggregat liksom några attribut och metoder ska ingå. Beskriv din notation! (5p)



Nämn 4 fördelar med att använda objektorientering! (4p)

- *Återanvändning – både mellan klassbibliotek och vid arv*
- *Modularisering – det är enklare för flera grupper att jobba med projektet, hör samman med:*
- *Inkapsling – en del av systemet kan byta intern representation utan att andra delar påverkas, och*
- *Produktivitet – ökas när samarbetet mellan grupper förbättras*
- *Modellering – systemet är en avbild av verkligheten, gör det enklare att förstå*
- *Terminologi – samma begrepp används i alla led i systemutvecklingen*
- *Förändringar är enklare*
- *Underhållet förenklas*

UPPGIFT 2 – Struktur (6p)

a) Förklara vad som menas med dynamisk bindning. (2p)
b) Det finns minst fyra sätt i C++ att ordna så att metoder i en klass kan komma åt attribut i en annan klass. Beskriv alla fyra sätt. (2p)
c) En av metoderna är oftast den ur OO-synpunkt mest fördelaktiga, vilken, och varför ? (1p)
d) För de övriga tre metoderna, ange varför de är mindre bra. (1p)

a) Dynamisk bindning betyder att en referens binds till ett objekt vid exekveringen, till skillnad från statisk bindning där det avgörs redan vid kompileringen vilket objektet är. En referens som är av typen Basklass kan bindas både till objekt av typen Basklass och subklasser till Basklass.
b) 1) Publika attribut, 2) protected vid arv, 3) friend, och 4) get/set-metoder
c) get/set-metoder, dessa bevarar inkapslingen
d) De övriga bryter inkapslingen i olika hög grad, publika attribut är värst och protected 'minst illa' (under föruttsättning att arvet är rimligt och inte bara kom till för att underlätta åtkomsten !).

UPPGIFT 3 – Java (6p)

a) I C++ kan objekt skapas på två sätt, antingen med *new* eller som automatiska variabler, typ *Bil b;*. I Java finns bara ett sätt, vilket ?
b) I C++:s algoritmbibliotek används *templates* flitigt. I Java finns inte templates med biblioteken har ändå ungefär samma funktionalitet. Vilken OO-relation används i stället för templates ?
c) Kan det vara någon skillnad i antal bitar i en *int* i Java jämfört med C++ ?
d) Varför behövs inte någon initieringskonstruktor i Java vid följande funktionsanrop:

```
Bil a, b;
a = funk(b);
```

e) I Java finns något som heter *AWT*, vad är det ?
f) Kan man överlagra operatorer (typ +, -) i Java ?

a) new används för att skapa nya objekt i Java; Bil b skapar en referens som kan referera till en bil (men inte gör det ännu).
b) Arv, och speciellt interfaces, kan användas på liknande sätt.
c) Ja, det kan det. Java har alltid 32 bitar för en int, i C++ kan det vara 16 – 64 (eller kanske numera 128) bitar.
d) Vid anropet skickas bara en referens, inte ett helt objekt, och därför behövs ingen speciell kopieringsrutin.
e) Abstract Windowing Toolkit, en samling klasser för fönsterprogrammering.
f) Nej, det kan man inte. En del har påpekat att + exempelvis är överlagrad, men frågan var om det gick att göra ytterligare överlagringar. + är överlagrad i de flesta programmeringsspråk (jmf 1.0 + 2.0 med 1 + 2).

UPPGIFT 4 – Klasser (8p)

Du jobbar på en av universitetets mer framstående institutioner och har tillsammans med några kollegor beslutat att anordna en musikfest. Varje festdeltagare tar med sig en CD-skiva med upp till tre låtar som bidrag, och din uppgift är att lägga samman all musik till en eller flera skivor. Eftersom du just har läst om den förträffliga objektorienteringstekniken beslutar du dig för att skissa på en klass för att lösa detta.

Skriv en klass CDSkiva (som representerar ett antal låtar, max 25 stycken) med **konstruktor**, **initieringskonstruktor**, **additionsoperator** och **tilldelningsoperator** (=).

Konstruktorn skall ta som argument den **maximala** inspelningstiden på skivan i fråga (i sekunder).

Additionsoperatorn skall lägga ihop två skivor, dvs skapa en ny skiva som innehåller alla låtar från båda skivorna. Om låtarna inte ryms (fler än 25 totalt eller för lång speltid) skall ett undantag genereras.

Utskrift skall också gå att göra, med << . Utskriften skall innehålla titlarna på skivan.

Dessutom skall det finnas en medlemsfunktion **int nyMelodi(string namn, int speltid)** som lägger in en låt på en skiva. Speltiden anges i sekunder. Funktionen skall returnera antalet sekunder som finns kvar, och –1 om låten inte rymdes. Du kan räkna med att det finns max 25 låtar på en CD.

Följande program skall fungera:

```
#include <iostream.h>
#include ”CDSkiva.h”

void main(void)
{
  CDSkiva kalle(74 * 60);           // 74 minuters speltid

  kalle.nyMelodi(”Mer Jul”, 210);
  kalle.nyMelodi(”Min Piraya”, 235);
  kalle.nyMelodi(”Anslagstavlan”, 35);

  CDSkiva musikFest(kalle);

  CDSkiva thomas(74 * 60);

  thomas.nyMelodi(”Rockodile Krock”, 250);
  thomas.nyMelodi(”Camarillo Brillo”, 550);
  thomas.nyMelodi(”Paradise by the dashboard ligth”, 830);
```

```
  musikFest = musikFest + thomas;

  cout << ”Innehållsförteckning: ” << musikFest << endl;
}
```

Lösningsförslag:
Observera att jag inte kunnat provköra detta pga problem med min Borland C++, dock går programmet genom kompilatorn !

```
#include <cstring.h>
#include <iostream.h>

class CDSkiva
{
   private:
      int maxtid;
      int hittillsTid;
      int antalTitlar;
      string titlar[25];     // avdrag om lagringsutrymme saknas
      int speltid[25];

   public:
      CDSkiva(int maxtid = 74 * 60)   // Vanlig konstruktor
      {
         this->maxtid = maxtid;
         hittillsTid = 0;
         antalTitlar = 0;
      }

      CDSkiva(const CDSkiva& skiva) // initieringskonstruktor
      {
         *this = skiva;               // utnyttjar operator= !
      }

      ~CDSkiva() {}

// testen på att tiden inte blir för stor kan göras här men det
// är faktiskt bättre att göra det vid tilldelningen
//(dålig spec i tentan)

      // avdrag om ingen temp-variabel finns med
      CDSkiva operator+(CDSkiva& b)  // a + b
      {
         if (antalTitlar + b.antalTitlar > 25)
            throw ("För många titlar");

         CDSkiva temp(*this);  // gör en kopia av vänsterledet

         for (int i = 0; i < b.antalTitlar; i++)
            if (temp.nyMelodi(b.titlar[i], b.speltid[i]) == -1)
               throw ("För lång speltid");;
```

```
      return temp;
   }

   CDSkiva& operator=(CDSkiva& b)
   {
      if (this != &b)            // avdrag om denna rad glömts
      {
         // om tiden på a inte räcker till (a = b)
         if (b.hittillsTid > maxtid)
         {
            throw ("För lång speltid");
         }
         antalTitlar = b.antalTitlar;
         hittillsTid = maxtid;
         for (int i = 0; i < antalTitlar; i++)
         {
            titlar[i] = b.titlar[i];
            speltid[i] = b.speltid[i];
         }
      }
      return *this;              // avdrag om denna rad saknas
   }

   int nyMelodi(string titel, int tid)
   {
      if (hittillsTid + tid > maxtid) return -1;
      if (antalTitlar == 25) return -1;

      antalTitlar++;
      titlar[antalTitlar] = titel;
      speltid[antalTitlar] = tid;
      hittillsTid = hittillsTid + tid;

      return maxtid - hittillsTid;
   }

   friend ostream& operator<<(ostream& os, CDSkiva cd)
   {
      os << "Dessa titlar finns på skivan:" << endl;
      for (int i = 0; i < cd.antalTitlar; i++)
      {
         os << "Titel " << cd.titlar[i];
         os << " tid [sekunder] " << cd.speltid[i] << endl;
      }

      return os;          // avdrag om denna rad glömts
   }
};
```

*Dessutom blir det avdrag om någon medlemsfunktion utelämnats. Glömda & och *-tecken har markerats där jag sett felet men*

leder inte till något poängavdrag, inte heller vanliga syntaxfel som glömda semikolon osv.

UPPGIFT 5 – S T L (6p)

Ryska Posten (RRRP) har tänkt virtualisera sig så därför ska alla frimärken säljas ut. Man får teckna sig för 1 frimärke och priset kommer att bli mellan 70 och 92 rubel. Priset sätts vid teckningstillfället beroende på efterfrågan enligt följande nykapitalistiska formel:

```
pris = 70.0 + 44.0*abs((antaltecknade-500000.0)/1000000.0)
```

och det finns en miljon frimärken till salu. Man anger till vilket högsta pris man vill köpa och kontonummer som beloppet ska dras ifrån före den 8 juni och efter detta och fram till den 15 juni beräknas priset (man har icke-thomas datorer av märket dull superslow med macrohard os) och sedan dras beloppet från konton hos de som köpt. Ryska personer har lustiga kontonummer som de känner till.

Följande program har den ryska överhögheten skrivit:

```
Posta ruski;
vector<Kundski> allihopi;
typedef vector<Kundski>::iterator iv;

void main(int argc,char*argv[])
{
  while (todayski() < junski8) // redan klart i ryska
  {                            // klassbiblioteket
    int maxbeloppski, kontski;
    cout << ”Vilki belopski?”; cin >> maxbeloppski;
    cout << ”Från vilket kontski?”; cin >> kontski;
    Kundski k(maxbeloppski,kontski);
    allihopi.push_back(k);
  }

  while (todayski() <= junski15) // klart enl ovan
    for (iv p = allihopi.begin();p < allihopi.end(); p++)
      if (!p->boughtski() && p->maxbetalski() >=
          ruski.frimarkPrisi())
        ruski.tekni(p->buyskikonto()));

  ruski.taInPengarna();  // dra från alla konton efter 15 juni
}
```

Skriv klasserna Posta och Kundski och ev fler så att det fungerar innan den 1 juni så får du en datja annars behöver du en mycket tjock tröja. Du kan anta att todayski(), junski8 resp 15 fungerar och att uttag görs via bel =

bankiRuski.geHit(belopp,konto); Antalet Kundski är en statshemlighet så du måste använda STL (Statliga Templaski Libresski). (För övrigt blir frimärkena värdelösa efter den 15:e juni.)

Kommentarer: Fick bara teckna en gång, vilket löses via nödlösning (sådana designfel är tyvärr vanliga i verklig kod). En iterator bör testas med != istf < (som i p < allihopi.endi()) för att klara listor etc också. Vanliga fel: Ej lagrat pris vid köptillfälle –1p, Inga kommentarer –0.5p. Nästan ingen har tänkt på namespaces och några har undvikt STL.

```
#include <iostream.h>
#include <vector.h>
//------------------Givet-------------------------------------
-------
class Bank
{ public:
  int geHit(int bel,int kto)
    { cout << "Uttag: " << bel << " Rubel från " << kto << ".";
return bel; }
} bankiRuski;

int todayski()      // För debugändamål
{
  cout << "Datum i Juni?"; int ret; cin >> ret; return
6*30+ret;
}
int junski8  = 6*30+8; // Något Förenklat
int junski15 = 6*30+15;
//---------------- Lösning ------------------------
class TransAkt // Håll ihop önskat tecknat belopp med kontonr
{
public:
  int b,k;
  TransAkt(int bel,int kto): b(bel),k(kto) {};
};

class Posta
{
  vector<TransAkt> konton; // alla som tecknat sig
  static int bel;
  public:
  friend void kammaHem(TransAkt& t);
  double frimarkPrisi()
     { return 70.0 + 44.0*abs((konton.size()-
500000.0)/1000000.0);}
  void taInPengarna() { bel = 0;
for_each(konton.begin(),konton.end(),kammaHem);
                   cout << "Rub" << bel  << " intaget!";}
  void tekni(int kto) { cout << kto;
konton.push_back(TransAkt(frimarkPrisi(),kto));}
};
int Posta::bel = 0;
```

```
void kammaHem(TransAkt& t) // Ej nödvändig. Bara för att visa
vad som händer
{
  Posta::bel += bankiRuski.geHit(t.b,t.k);
}
class Kundski
{
  int bet;  // Vad man är villig att betala
  int konto;
  bool buyed; // får bara teckna en gång
public:
Kundski(int b,int k):bet(b),konto(k),buyed(false) { }
int maxbetalski() { return bet;}
bool boughtski() { return buyed; }
int buyskikonto() { buyed = true; return konto;} // Obs
sidoeffekt(Nödlösn)!
};
```

UPPGIFT 6 – Templates (5p)

Implementera en template som lagrar/hämtar dataposter via en nyckel på valfritt sätt. Maximala antalet poster anges vid användning av templaten. Nycklarna antas vara unika.

```
template<class Key, class Data, int max=10> class Datalager
{
  public:
    Datalager();
    ~Datalager();
    void lagra(const Key& k,const Data& d) throw(char*);
                                          // om   fullt

    Data *finn(const Key& k); // returnerar 0 om nyckeln
                              // ej finns

  private:
    … // implementeras av dig samt medlemsfunktionerna ovan
};
```

Exempel på användning:

```
Datalager<long,int,60> tentares;
tentares.lagra(690106, 32);
tentares.lagra(740219, 30);
long pnr; cout << "Födelsedatum? "; cin >> pnr;
int *d = tentares.finn(pnr);
if (d)
  cout << *d << ” poäng.”;
else
```

```
  cout << "Hittar ej";
```

På den här uppgiften bör syntaxen vara hyfsat rätt.

```
template<class Key, class Data, int max=10> class Datalager
{
  public:
    Datalager():ix(0){};
    void lagra(const Key& k,const Data& d) throw (char*);
    Data *finn(const Key& k); // Def ej direkt i klassdekl (-
1p)

  private:
    Key  skeys[max]; // Vector hade varit en (bättre)lösn utan
max-storlek
    Data sdatas[max];// eller att briljera med new/delete
    int ix;
};
template<class Key, class Data, int max=10>
void Datalager<Key,Data,max> ::lagra(const Key& k,const Data&
d) throw (char*)
{
  if (finn(k)) throw "Finns redan";  // Kolla (ev) för
säkerhets skull
  if(ix<max)
  {
    skeys[ix]=k;
    sdatas[ix++]=d;
  } else throw "Overflow in lagra";
}

template<class Key, class Data, int max=10> Data*
Datalager<Key,Data,max> :: finn(const Key& k)
{
  for (int i=0;i<ix;i++)
    if (k==skeys[i]) // vanligt att skriva =  -0.5p
      return &sdatas[i]; // Vanligt att glömma & -0.5p
  return 0;
}
```

< SLUT PÅ UPPGIFTER >