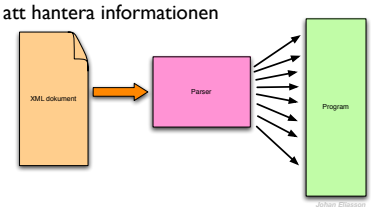
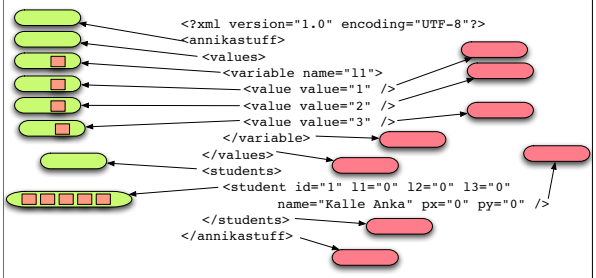
XML - SAX

Alternativ metod för att läsa XML filer

- Inte parsa hela dokumentet på en gång
- Inte bygga upp ett träd i minnet
- Istället ta element för element
- Upp till programmet att hantera informationen






```
public void parseAndPrint( String fileName )
{
   DefaultHandler eventHandler = new MyEventHandler();

   try{
      SAXParserFactory factory = SAXParserFactory.newInstance();
      factory.setNamespaceAware( true );

      SAXParser parser = factory.newSAXParser();

      parser.parse( new InputSource( fileName ), eventHandler );
   } catch ( ParserConfigurationException pce ) {
      pce.printStackTrace();
   } catch ( SAXException se ) {
      se.printStackTrace();
   } catch ( IOException ioe ) {
      ioe.printStackTrace();
   }
}
```


VARNING

Inte någon snygg lösning

Ide till lösning

- Subklassa DefaultHandler
- Implementera
  - startElement
  - endElement

```
public void startElement( String namespace, String localName, String qName, Attributes attr )
{
    switch( currentlyParsing ){
        case 0:
            ...
        case ANNIKA_STUFF:
            if( localName == "values" ){
                currentlyParsing = VALUES;
            } else if( localName == "students"){
                currentlyParsing = STUDENTS;
            } else {
                System.out.println( "Syntax error: Expected 'values' or 'students', got '" + localName + "
            }
            break;

        case VALUES:
            if( localName == "variable" ){
                currentlyParsing = VARIABLE;
                currentVariableName = attr.getValue( "name" );
                variableValues = new ArrayList<String>();
            } else {
                printError( "variable", localName );
            }
            break;

        case STUDENTS:
            if( localName == "student" ){
                currentlyParsing = STUDENT;
                // Anta att alla attribut är korrekt deklarerade
                System.out.println( attr.getValue( "name" ) + ", l1=" + attr.getValue( "l1" ) + ", l2=" +
                    attr.getValue( "l2" ) + ", l3= " + attr.getValue( "l3" ) + ", px= " + attr.getValue( "
                    ", py=" + attr.getValue( "py" ) );
            } else {
                printError( "student", localName );
            }
            break;

    }
}
```

Johan Eliasson

```
public void endElement(String uri, String localName, String qName)
{
  switch( currentlyParsing ){
    case ANNIKA_STUFF:
      currentlyParsing = 0;
      break;
    case VALUES:
      currentlyParsing = ANNIKA_STUFF;
      break;
    case STUDENTS:
      currentlyParsing = ANNIKA_STUFF;
      break;
    case VARIABLE:
      currentlyParsing = VALUES;
      System.out.print( currentVariableName + " - " );
      for( String v : variableValues ){
        System.out.print( v + " " );
      }
      System.out.println("");
      break;
    case VALUE:
      currentlyParsing = VARIABLE;
      break;
    case STUDENT:
      currentlyParsing = STUDENTS;
      break;
  }
}
```

Johan Eliasson

```
> java Demo hejsan
l1 - 1 2 3
l2 - 4 5
l3 - 6 7 8
px - 11 12
py - 13 14
Kalle Anka, l1=0, l2=0, l3= 0, px= 0, py=0
Kajsa Anka, l1=0, l2=0, l3= 0, px= 0, py=0
Knatte Anka, l1=0, l2=0, l3= 0, px= 0, py=0
Fnatte Anka, l1=0, l2=0, l3= 0, px= 0, py=0
Tjatte Anka, l1=0, l2=0, l3= 0, px= 0, py=0
```

Johan Eliasson

Andra objektorienterade språk

Jan Erik Moström

Objektorienterade språk

- Historik
  - Simula 67
  - Smalltalk 80
- Procedurorienterad programmering
  - Subprogram
  - Programbibliotek
- Dataorienterad programmering
  - Abstrakta datatyper
  - Objektbaserade språk, föregångare till…

Objektorienterade programmeringsspråk

- Smalltalk, Eiffel och (Java) är rena objektorienterade språk
- C++ och Ada stödjer objektorientering
- Det går att programmera objektorienterat i C eller assembler för den delen
  - Språket måste stödja någon form av inkapsling.
  - T.ex. på fil-nivå

C++

- Skapades i början av 80-talet av Bjarne Stroustrup (AT&T Bell Labs)
- 1985 släpptes C++ kommersiellt och fick snabbt spridning inom UNIX-värden

Största skillnaderna mot java

- Multipelt arv
- Inte *Garbage Collection* som standard
- Inte bara klasser (vi kan skriva funktioner också)
- Alla klasser inte i samma arvshierarki (ingen Object klass)
- Möjlighet till operatoröverlagring
- Klassbiblioteket (mindre i c++)
- Inte plattformsoberoende i samma grad

Målen med C++

- Bakåtkompabilitet med ANSI C i så hög grad som möjligt
- Inkludera nya programmeringsparadigmer som objektorientering utan att för den skull offra prestanda.
- C++ har:
  - Hårdare typkontroll än C
  - Mindre behov av preprocessorn, eftersom man har möjlighet att skriva funktioner som fungerar som makron, samt möjlighet att definiera konstanter utan att använda sig av #define.
  - Har möjlighet att skicka parametrar via ”call-by-reference” utan att använda sig av adressoperatorn &
  - Ett stort bibliotek av färdiga klasser och algoritmer
  - Möjlighet till multipelt arv
  - Templates


```
class Complex {
public:
  // Constructor
   Complex(double rl=0, double im=0) {
      real = rl; imag = im;
   }
// Member operators
   Complex operator+ (Complex operand2) const;
   Complex operator* (Complex operand2) const;
private:   // Data members
   double real;  // real and imaginary parts
   double imag;  //   of a complex number
// I/O operators
friend istream& operator>>(istream& is,Complex& innum);
friend ostream& operator<<(ostream& os,Complex outnum);
};
```



```
Complex Complex::operator+ (Complex operand2) const
{
   Complex result(real+operand2.real,
                  imag+operand2.imag);
   return result;
}

int main(void)
{
   Complex tal1; //0
   Complex tal2(1,2); //1+2i
   cout << tal1+tal2 << endl;
   return 0;
}
```


Allokering/deallokering av objekt

- Var allokera objekten? Tre sätt :
  - Statiskt av kompilatorn
  - Dynamiska objekt på stacken
  - Eller på heapen med new
- I Java new det enda sättet -> uniform metod
  - Inga pekare som måste derefereras a la C++
- C++ har alla sätten - och alla problemen

Vi använder new, sen då?

- Hur deallokera objekten?
- ->Explicit
  - Problem med ’dangling pointers’
  - C++ har **delete**
- -> Implicit
  - Har vi i java i form av Garbage Collection

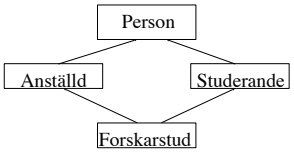
Arv i C++

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


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


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);
};
```


Multipelt arv




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


Problem med multipelt arv

- Omdiskuterat behov
- Namnkonflikter kan uppstå
- Dubbla data (löses med virtuellt arv)
- Använd med försiktighet

Olika typer av arv i C++

- Public
  - Används alltid om det inte finns något mkt bra skäl till att använda någon av de andra.
  - `public` och `protected` metoder behåller sin status
- Protected
  - `public` och `protected` metoder/attribut som ärvs blir `protected` i subklassen
- Private
  - `public` och `protected` metoder/attribut som ärvs blir `private` i subklassen
- Attribut/metoder som är deklarerade `private` blir alltid oåtkomliga i basklassen

Virtuella funktioner

- Statisk bindning är default - nyckelordet `virtual` används för att få dynamisk bindning

```
class Figur
{
   public:
     virtual double getArea();
};
class Rect : public Figur
{
   private:
     double b,h;
   public:
     double getArea() { return b*h; } // virtuell
};
```


Virtuella anrop med pekare

```
class Circ: public Figur
{
   private:
      double r;
   public:
      double getArea() { return 3.14*r*r; }
};

Figur *fp[2];
fp[0]=new Circ(5)
fp[1]= new Rect(3,4);
for(int i=0;i<2;i++)
    cout << ”Area ” << fp->getArea();
```


Virtuella anrop med referens

```
void skrivUtArea(Figur& f) {
   cout << ”Arean är ” << f.getArea() << ”.”;
}
Cirk jorden(40000000/M_PI/2);
skrivUtArea(jorden);
```

- Referensanrop behåller datat

Abstrakt basklass

- Fångar gemensamma egenskaper
- Påtvingar ett gränssnitt
- Går ej skapa objekt från
- Med äkta (pure) virtuell funktion:

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

- Eller med `protected` konstruktor

Virtuellt arv

- Används vid multipelt arv för att se till så att data inte finns i flera kopior

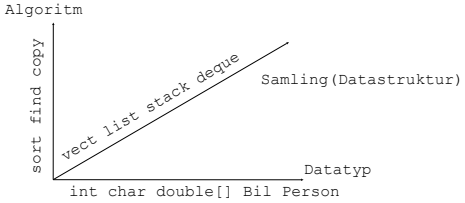
```
class Queue{
    // Member list
};
class CashierQueue : virtual public Queue{    // Member list
};
class LunchQueue : virtual public Queue{
    // Member list
};
class LunchCashierQueue : public LunchQueue, public
   CashierQueue{
    // Member list
};
```


Algoritmbiblioteket (STL)

- (f.d.) Standard Template Library
- Samlingstyper (containers)
- Algoritmer
- Funktionsobjekt
- Adaptrar

Designstrategi

- Generiska algoritmer som fungerar på godtyckliga samlingsdatatyper, vilka har iteratorer



forts designstrategi

- Effektiva algoritmer
- Använder enbart mallar i C++
  - Inga arvshierarkier eller virtuella funktioner
- Ej objekt-orienterat
- Bygger på forskning om generiska algoritmer av Stepanov/Lee/Musser -79-95
  - General Electric, AT&T Bell, HP
- Presenterades för ANSI/ISO C++ 93
  - Mycket gott mottagande

Samlingstyper

- Sekvenser
  - vector<T>,deque<T>,list<T>
- Associativa
  - set<T,C>, map<K,V,C>, multiset<T,C>, multimap<K,V,C>
- Adaptrar
  - stack<T>, queue<T>, priority_queue<T,C>
  - impl t.ex. som vector, deque, eller list

Anm. C comparator. Funktionsobjekt för sortering

Algoritmer

- Ca 70 st
- Opererar på intervall
  - Oberoende av datastruktur
- Sekvensalgoritmer
  - for_each, find, copy, transform, replace
- Sortering
  - sort, stable_sort, binary_search, max_element
- Numeriska
  - accumulate, inner_product

Ett litet exempel

```
void sort_file(char *name)
{
  ifstream file(name);
  istream_iterator<string> in(file),eof;
  vector<string> strs;

  copy(in,eof,back_inserter(strs));
  sort(strs.begin(),strs.end());
  copy (strs.begin(),strs.end(),
    ostream_iterator<string>(cout,"\n"));
}
```


C#

- Utvecklat av Microsoft
- Designmål
  - Enkelt objektorienterat språk
  - Robusthet
  - Programmerarportabilitet
- Alla typer i språket ärver en klass (System.Object)

Exempel

```
using System;

// A class for two-dimensional objects.
class TwoDShape {
  double pri_width;  // private
  double pri_height; // private

  // properties for width and height.
  public double width {
     get { return pri_width; }
     set { pri_width = value; }
  }

  public double height {
     get { return pri_height; }
     set { pri_height = value; }
  }

  public void showDim() {
    Console.WriteLine("Width and height are " +
                      width + " and " + height);
  }
}
```



```
// A derived class of TwoDShape for triangles.
class Triangle : TwoDShape {
  string style; // private

  // Constructor
  public Triangle(string s, double w, double h) {
    width = w;  // init the base class
    height = h; // init the base class
    style = s;  // init the derived class
  }

  // Return area of triangle.
  public double area() {
    return width * height / 2;
  }

  // Display a triangle's style.
  public void showStyle() {
    Console.WriteLine("Triangle is " + style);
  }
}
```



```
public class Shapes3 {
  public static void Main() {
    Triangle t1 = new Triangle("isosceles", 4.0, 4.0);
    Triangle t2 = new Triangle("right", 8.0, 12.0);

    Console.WriteLine("Info for t1: ");
    t1.showStyle();
    t1.showDim();
    Console.WriteLine("Area is " + t1.area());

    Console.WriteLine();

    Console.WriteLine("Info for t2: ");
    t2.showStyle();
    t2.showDim();
    Console.WriteLine("Area is " + t2.area());
  }
}
```