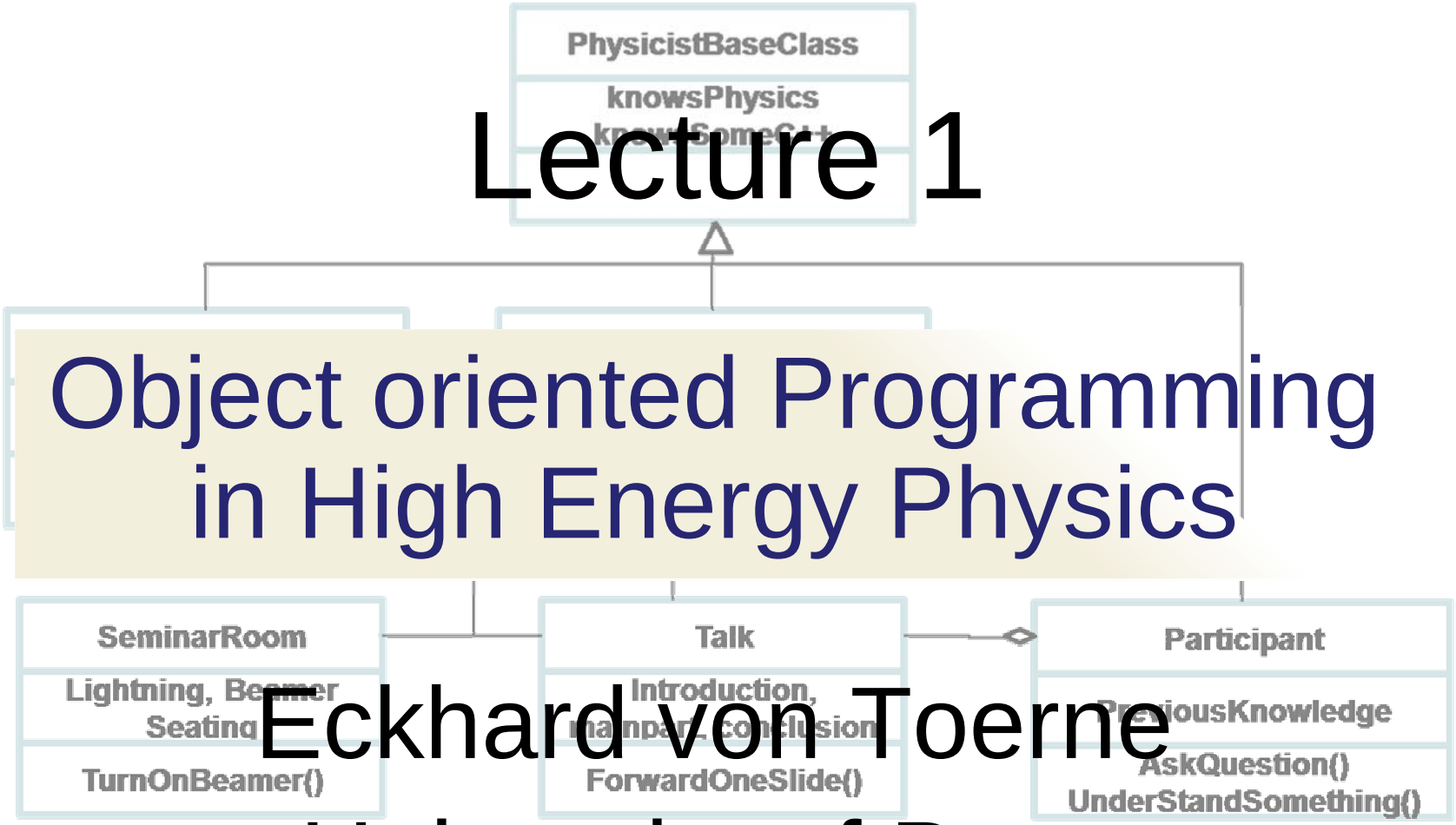
universitätbonn

Helmholtz Workshop

PHYSICS AT THE TERA SCALE Helmholtz Alliance

Advanced Methods of Software Programming



Lecture 1

Object oriented Programming in High Energy Physics

Eckhard von Toerne
University of Bonn

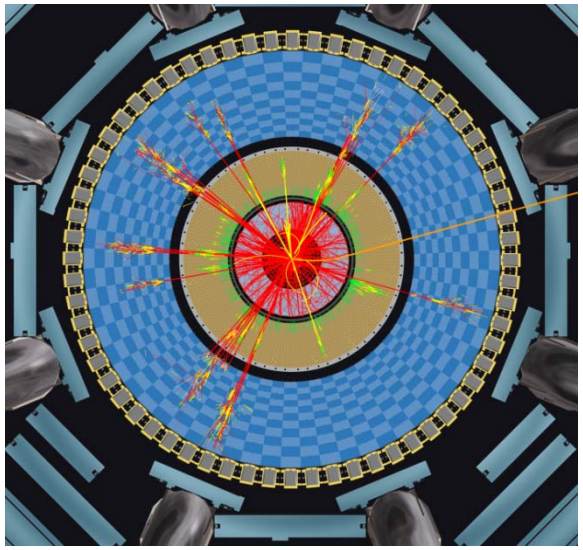
universitätbonn

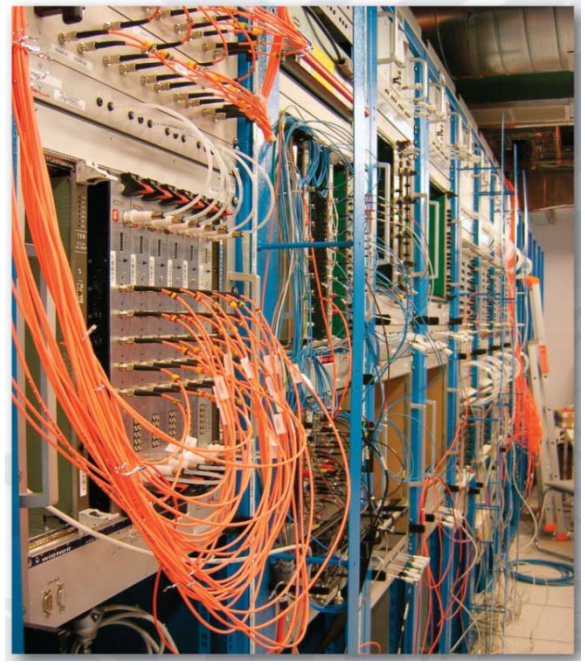
Contents

- Introduction: Ingredients of Object Oriented Programming
  - Concepts of object orientation
  - HEP experiment analysis code case study
  - What is an electron object
- What are Design patterns?
- C++ features and issues

universitätbonn

C++ in HEP
Example: ATLAS Experiment



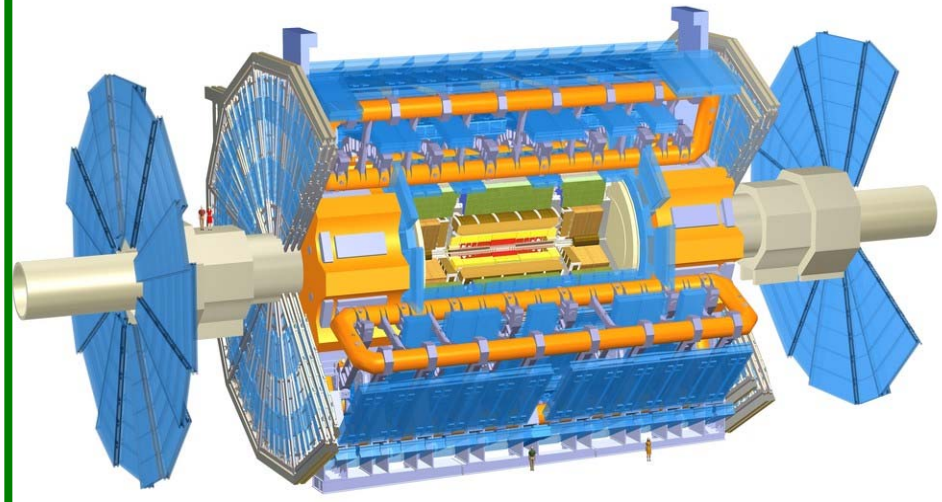


- proton-proton collision at up to 14 TeV E_{cm}.
- Size: ~22 x 22 x 44 meter^3
- Bunch crossings every 25 ns,
- Event size ~2 MB
- Several thousand particles expected to be created in every collision →0.1 Peta-byte/ sec data source awaiting direct analysis,
- Stored on tape ~200 events / sec.

  ~ 1 GB/sec

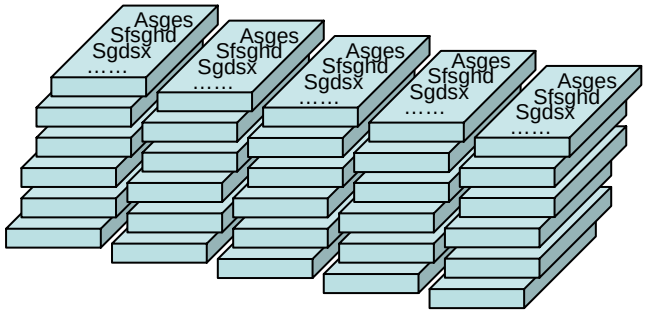
ATLAS and its software

ATLAS Detector



Weight ~7000 tonnes

ATLAS Source Code
~1 Mio lines of code



Humorous sidenote:

If written in stone…

30 lines of code per stone slab and 5kg per slab →

~200 tonnes

universitätbonn

Concepts of Object Orientation

- Most software concepts only beneficiary if problem is complex enough
- Must discuss complex example

Concepts:

- **Classes**
- **Encapsulation**
- **Inheritance**
- **Polymorphism**
- **Templates**

Why we need object orientation

Last Century Source Code

„The ancient language department is still refusing to teach Fortran77“

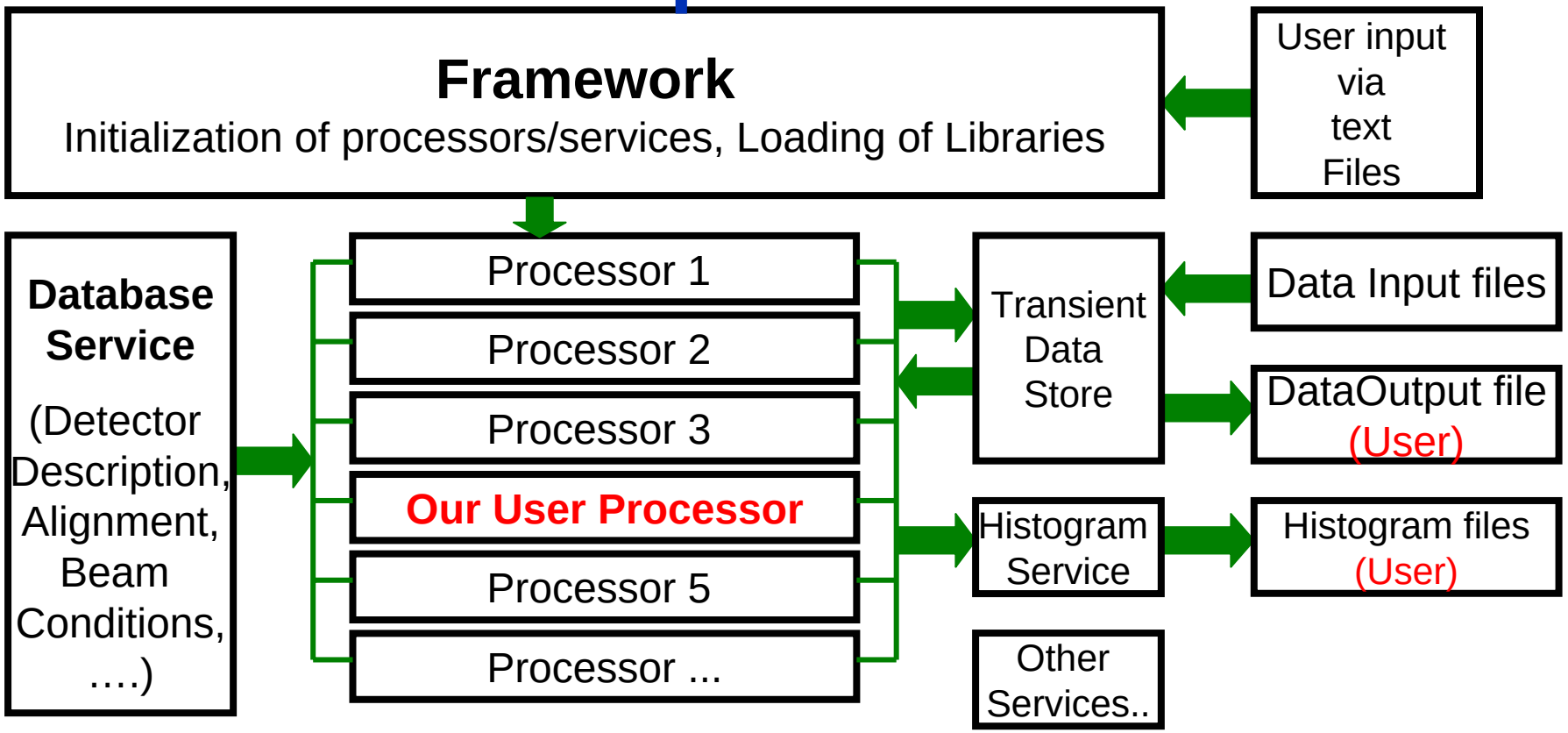
Next: Tour through reconstruction code of OPAL experiment (early 90ies)

We need object oriented code to tackle the complex problems that are the focus of science in the 21st century

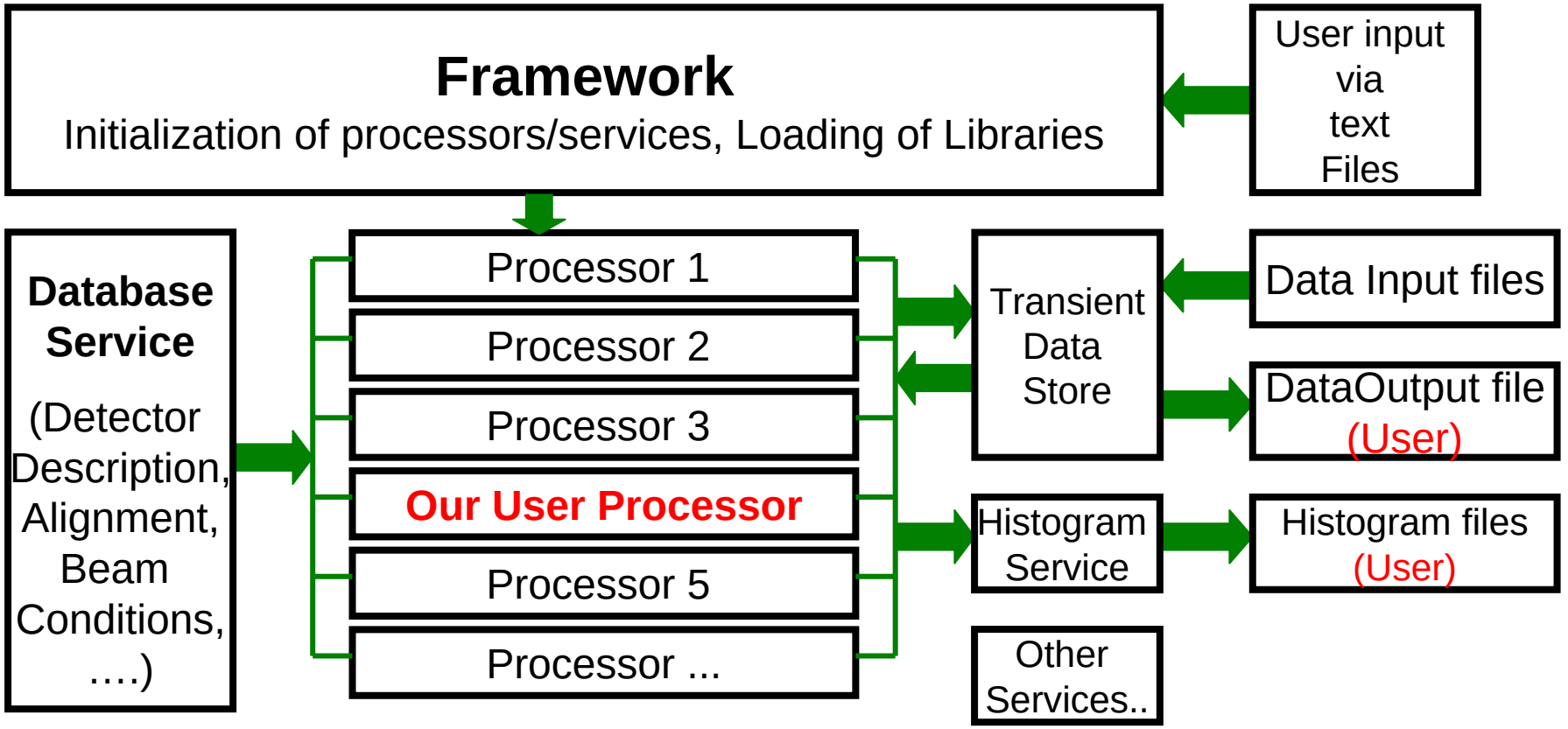
Object oriented languages (Java, Python, C++, Fortran90,...)

universitätbonn

Example Analysis Framework of HEP experiment



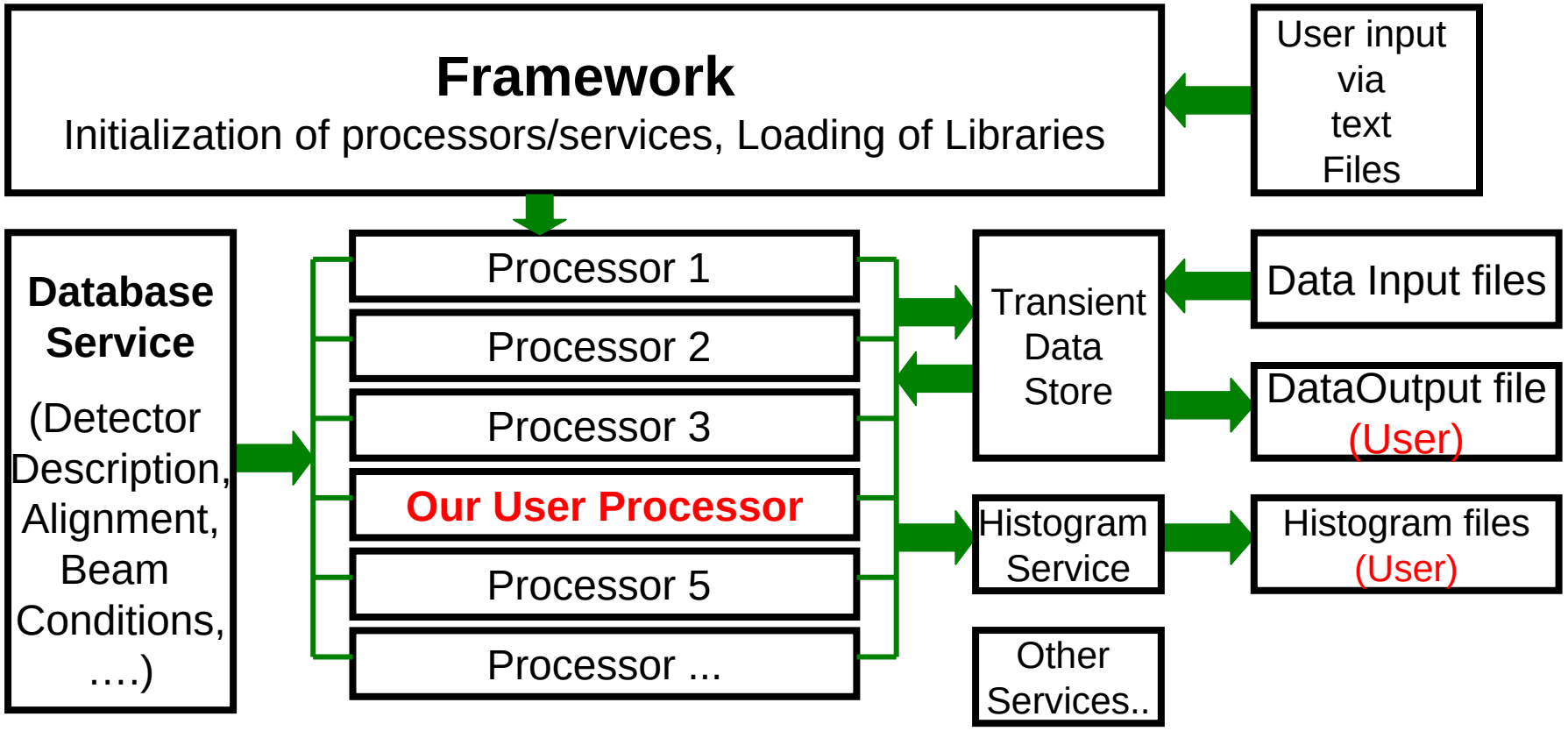
Classes



- Elements of analysis framework described in classes
- Packaging data and functions together

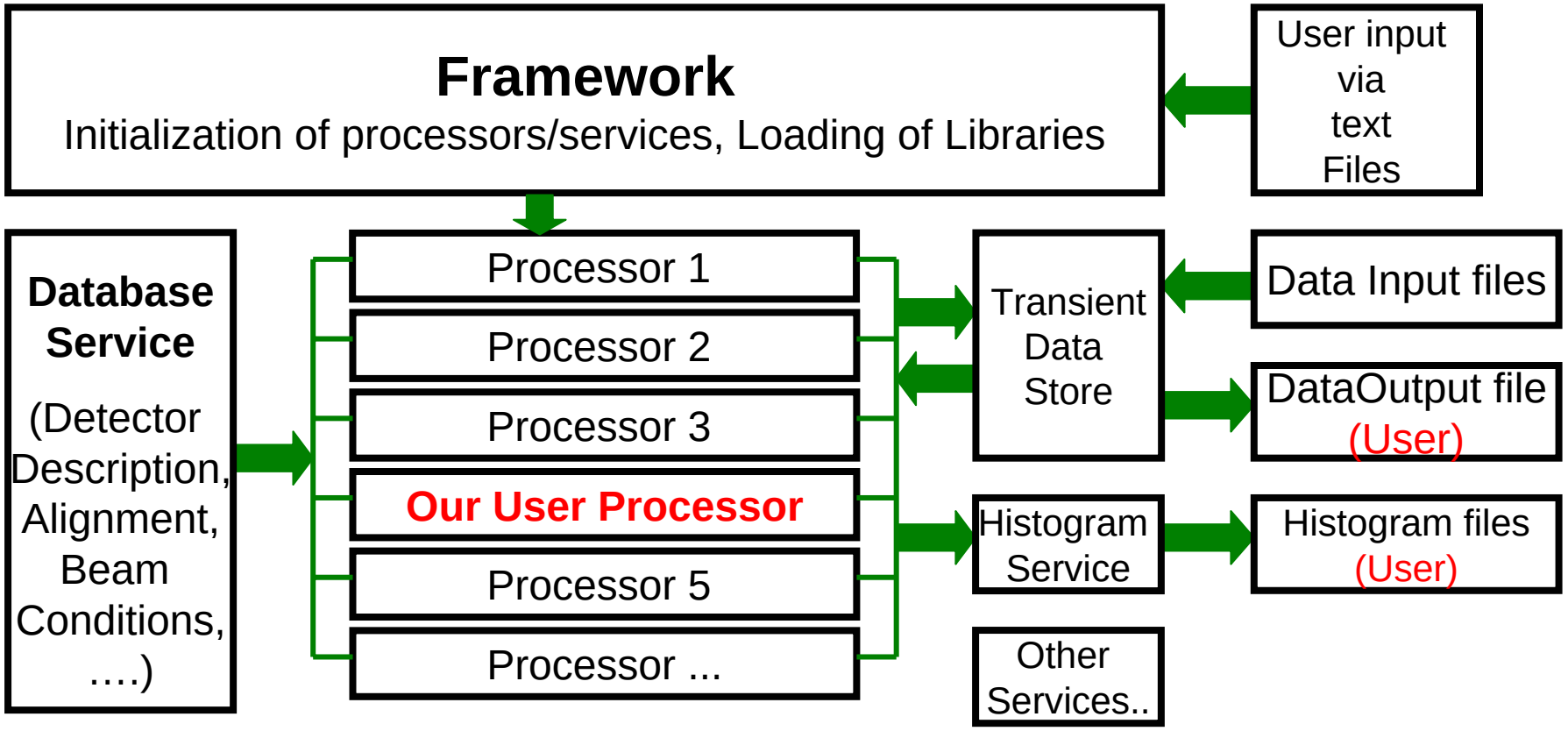
universitätbonn

Encapsulation



- Framework does not know about processors
- Storegate does not know about Datatypes
- Acces to data is restricted (private functions)

Inheritance



- Processors are derived from common base class
- Functions common to all processors are only defined once

- Data classes are derived from **`DataObject`** interface

universitätbonn

What is Inheritance

Inheritance is a method of including data and functions of one class into another class.

Syntax:

```
Class Daughter : public Mother {
…
}
```

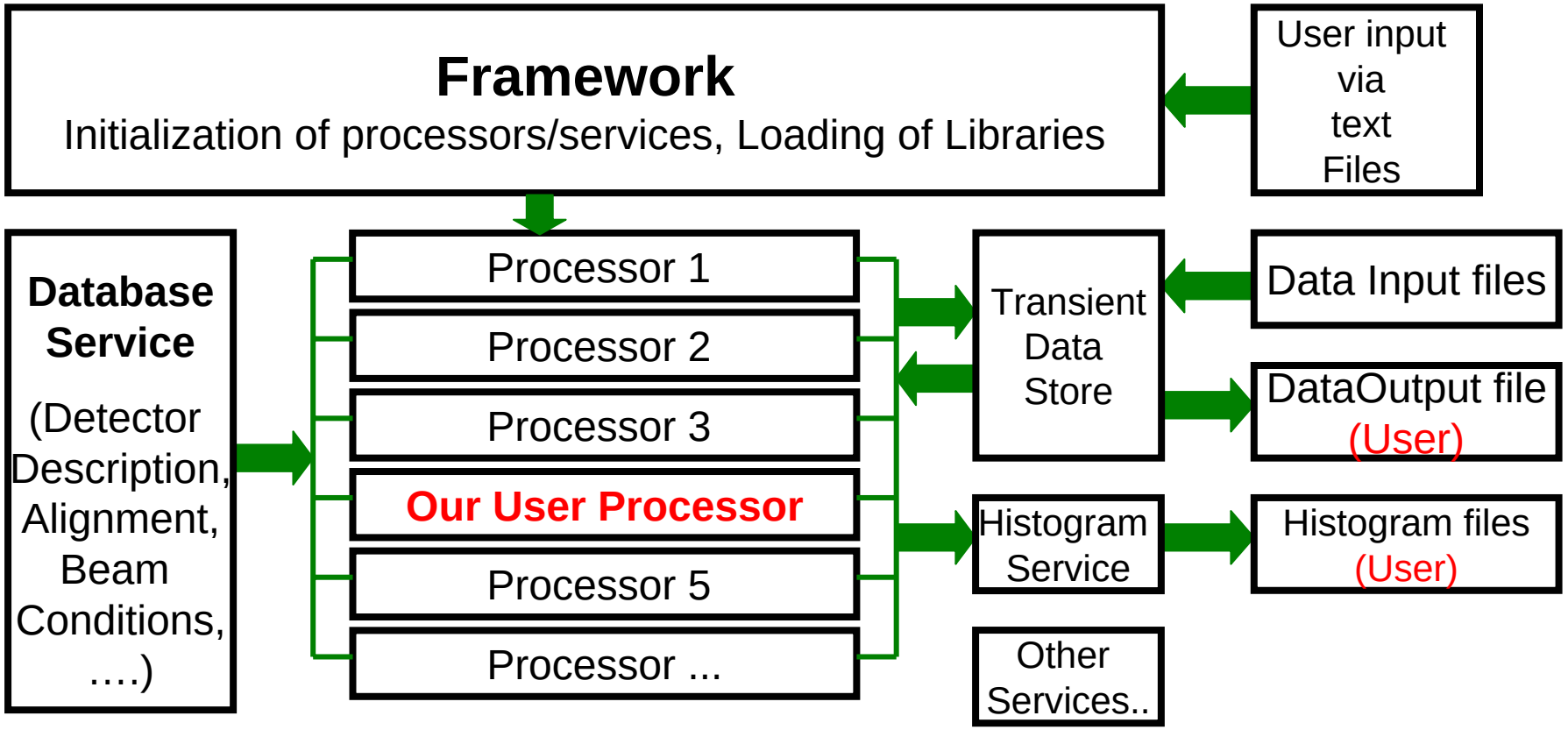
Inheritance is either public (most common), private or protected

(Careful: Without specifying the inheritance type: private is chosen)

	Private Inherit.	Public Inherit.
private data	unusable	unusable
public data	usable / private	public
protected data	usable / private	protected

universitätbonn

Polymorphism



- Transient Data Store does not know about individual data classes. Storage class handles Base class pointers
- Polymorphism: Base class pointer accepts derived classes

universitätbonn

Polymorphism

Polymorphism: Way to treat objects which belong to different classes in a similar way. Derive classes from base class

Define common functionality in base class

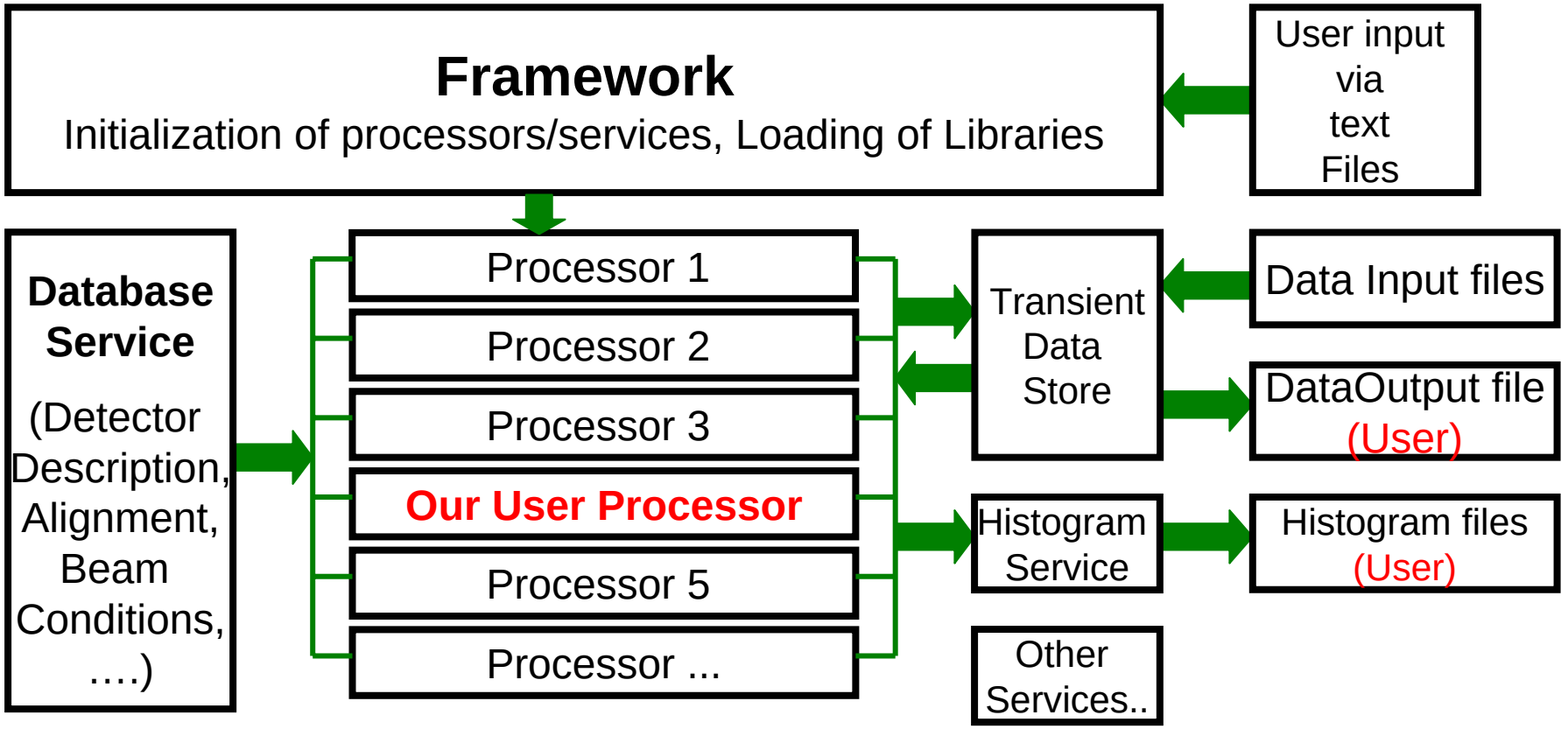
```
class Vehicle{ ..
Print();
.. };  // base class
class Car: public Vehicle {  .... }; // derived class Car

vector <Vehicle*> vehicleList;
vehicleList.push_back(new Car("Ford","Fiesta"));
vehicleList.push_back(new Truck("EightWheel","MyTruck"));
For (int i=0; i<vehicleList.size();i++) vehicleList[0]->Print();
```

Which Print() is executed? Normal case: Vehicle::Print();

Derived class' Print() may be used by applying virtual functions

Templates



- Templates are discussed in exercise

Why copy-and-paste is “evil“

Concepts of Object Orientation designed to avoid blow-up of code

- Classes (“packaging data and functions together“)
- Encapsulation (“shielding your code“)
- Inheritance (“code common to several classes defined in base class “)
- Polymorphism (“classes are handled via interfaces“)
- Templates (“ultimate copy-and-paste killer“)

universitätbonn

Language support of OO features

Feature	C++	Java	Python
Classes	ok	ok	ok
Encapsulation	ok	ok	ok
Inheritance	ok	ok	ok
Polymorphism	ok	ok	ok
Templates	ok	ok	ok

- C++ used by all LHC experiments, B-Fac. and ILC
- No experiment uses C++ exclusively
- Will concentrate on C++ but all examples translatable into other languages (C++, Java, Python, Ruby, ….)

Code navigation exercise I

http://alxr.usatlas.bnl.gov/lxr-stb4/source/atlas/Reconstruction/egamma/egammaPIDTools/src/egammaElectronCutIDTool.cxx

```
  // cuts on TRT
  declareProperty("CutBinEta_TRT",m_CutBinEta_TRT,
                  "Eta binning in TRT");
  // cut on Number of TRT hits
  declareProperty("CutNumTRT",m_CutNumTRT,
                  "cut on Number of TRT hits");
  // cut on Ratio of TR hits to Number of TRT hits
  declareProperty("CutTRTRatio",m_CutTRTRatio,
                  "Cut on Ratio of TR hits to Number of TRT hits");
  // cut on Ratio of TR hits to Number of TRT hits for 90% efficiency
  declareProperty("CutTRTRatio90",m_CutTRTRatio90,
                  "cut on Ratio of TR hits to Number of TRT hits for 90% efficiency");

}

// ================================================================
egammaElectronCutIDTool::~egammaElectronCutIDTool()
{
}

// ================================================================
StatusCode egammaElectronCutIDTool::initialize()
{
  IToolSvc* toolSvc = 0;
  StatusCode sc = service("ToolSvc",toolSvc);
  if ( sc.isFailure() ) {
   ATH_MSG_ERROR("Cannot find ToolSvc! ");
    return StatusCode::FAILURE;
  }
```

declareProperty:
Inherited function from Processor Base Class

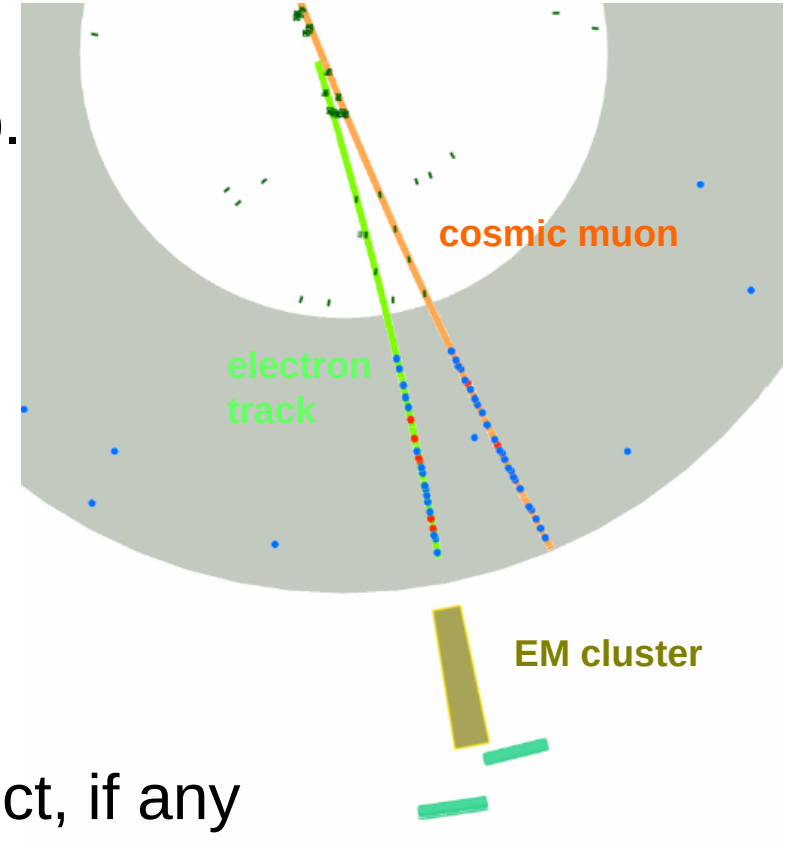
service:
Inherited function from Processor Base Class

Usage of Preprocessor macro for log messages

Electron C++ object

Signature of electron in detector:

- Calo Cluster consistent with e/γ hyp.
- track pointing to cluster
- # high threshold TRT hits on track consistent with e- hyp.
- E/p ~1



Creation of a delta electron in ATLAS Cosmics data (Approved Plot, J.Kraus)

What is an electron object?

- “pointer“ to a cluster
- “pointer“ to a track
- “pointer“ to a $\gamma \rightarrow$ ee conversion object, if any
- functions:
  - GetCluster()
  - GetTrack()
  - GetEOverP()

Electron class source code

atlas/Reconstruction/egamma/egammaEvent/egammaEvent/Electron.h and egamma.h

```
class Electron : public egamma
{
public:
  /** @brief default constructor */
  Electron() :
    IAthenaBarCode(),
    INavigable (),
    I4Momentum (),
    INavigable4Momentum (),
    egamma()
  { };

  /** @brief constructor */
  Electron(unsigned int author) :
    IAthenaBarCode(),
    INavigable (),
    I4Momentum (),
    INavigable4Momentum (),
    egamma(author)
  { };

  /** @brief destructor */
  ~Electron() { };
```

```
class egamma
  : public ParticleImpl<
             egammaNavigation, // not really a terminal
             P4ImplEEtaPhiM >
{

  ////////////////////////////////////////////////////
  // Public typedefs:
  ////////////////////////////////////////////////////
 public:

  // for readability and lazy people
  typedef ParticleImpl< egammaNavigation,
                        P4ImplEEtaPhiM
          > egammaImpl_t;
  typedef egammaImpl_t::navigable_type navigable_type;
  typedef egammaImpl_t::momentum_type  momentum_type;
  typedef egammaImpl_t::particle_type  particle_type;

 ...
private:

 ElementLink<CaloClusterContainer> m_cluster;
 ElementLinkVector<Rec::TrackParticleContainer>
    m_trackParticle;
 ElementLinkVector<VxContainer>  m_conversion;
 ElementLinkVector<egDetailContainer> m_egDetails;
```

universitätbonn

Design Pattern

universitätbonn

Design Patterns

Design pattern pioneered in architecture:

recurring solution to design problems

Introduced by architect C. Alexander, "*A Pattern Language: Towns, Buildings, Construction*. Oxford University Press (1977).

In the 90ies adapted to computer science

Design Pattern Categories

- Creational Patterns
- Structural Patterns
- Behavioral Patterns

expressed in a diagrammatic language (see S. Kluth's lecture tomorrow)

Computer Literature

- “Gang Of Four“ book:

  “Design Patterns, elements of reusable object-oriented software“

  E. Gamma et al., Addison-Wesley 1995

  Description of ~30 patterns in computing, applications + structure + diagramatic description

- F. Buschmann et al., “Pattern oriented software architecture“, Wiley 1996

  (contains blackboard pattern)

universitätbonn

Review of C++ Features and other Issues

universitätbonn

A class definition

```
class HexField : public TPolyLine {

private:
  int    fX;      // X position(column)  in the fPad
  int    fY;      // Y position(line)  in the fPad
  int    fId;     // unique ID
  static const int fNPoints = 7; // number of points in polyline
  bool fNotTaken;
  HexBoard* fBoard; // Canvas  where box is in
  double fxC[fNPoints];    // x position of hexagon corners
  double fyC[fNPoints];    // y position of hexagon corners

public:
  // constructor
  HexField(HexLocation* u, HexBoard *c)
    : fBoard(c), fX(u->x), fY(u->y) {
    DrawField();
  }

  // destructor
  ~HexField() { }

  int GetX(){return fX;}
  int GetY(){return fY;}

  void SetOccupied(int col) {
    fNotTaken = false;
    SetFillColor(col);
    Draw("f");
  }
  void DrawField();

  void ExecuteEvent(int a, int b, int c);

}; // do not forget the semicolon
```

A class definition

```
class HexField : public TPolyLine {

private:
  int     fX;         // X position(column)  in the fPad
  int     fY;         // Y position(line)  in the fPad
  int     fId;        // unique ID
  static const int fNPoints = 7; // number of points in polyline
  bool fNotTaken;
  HexBoard* fBoard; // Canvas  where box is in
  double fxC[fNPoints];     // x position of hexagon corners
  double fyC[fNPoints];     // y position of hexagon corners

public:
  // constructor
  HexField(HexLocation* u, HexBoard *c)
    : fBoard(c), fX(u->x), fY(u->y) {
    DrawField();
  }

  // destructor
  ~HexField() { }

  int GetX(){return fX;}
  int GetY(){return fY;}
  void SetOccupied(int col) {
    fNotTaken = false;
    SetFillColor(col);
    Draw("f");
  }
  void DrawField();

  void ExecuteEvent(int a, int b, int c);

}; // do not forget the semicolon
```

Class name

List of inheritance, might be empty

class body starts here, enclosed by { }

Everything behind this is private: invisible from outside

Data, convention used here: all class data start with f…

A constant which is common to all objects of this class → thus „static“

Everything behind this is public: visible from outside

← constructor

Init List, in which other constructors are called, might be empty

← destructor

Complete declaration of function

Type

Function name

Argument list enclosed by ( )

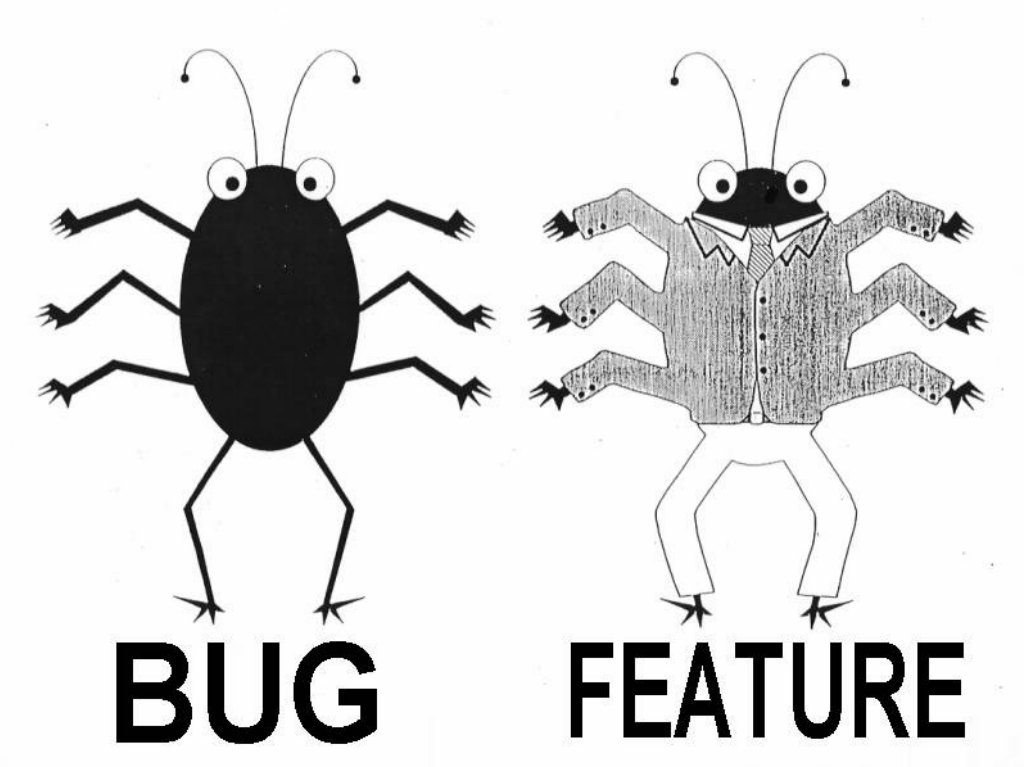
Function Body enclosed by { }, no semicolon

Forward declaration of function, no body, has semicolon

Empty argument list

universitätbonn

Bugs Top-10 list (C++)



universitätbonn

Worst Bugs top-10 list

#10 bad usage of char*

```
char* name="HAL";
char* input[20];
cin >>input;
if (input == name) cout << "Good Morning" << endl;
```

#9 errors in an if-clause bool statement

```
int a,b;
if ( a = b || a =!3 ) { ...
```

#8 wrong cast of an object

```
double d    = GetValue();
double* poi = &d;
int* poi2   = (int*) poi;
```

#7 STL-containers with a faulty less-operator

```
Set < Pair<int>, MyLess > s;
Bool MyLess(const Pair<int>& p1, const Pair<int>& p2){
 return (p1.first < p2.first && p1.second < p2.second;}
```

#6 accessing an array out of its boundaries

```
aclass a[4];
aclass* ap = a[3];
ap++; ap->Print();
```

universitätbonn

Worst Bugs top-10 list

#5 overwriting a class variable with a local variable

```
class aclass {
double fDat, fDat2;
SetDat(int i){ double fDat = i; }
};
```

#4 deleting a pointer which is used elsewhere

```
aclass* ap = &a[2];
checkPointer(ap);
ap->Print();
```

```
void checkPointer(aclass* ap){
  delete ap;
  return;}
```

#3 a pointer to a stack object as a function return value

```
aclass* GetAclass(int i){
  aclass a;
  a.SetDat(i);
  return &a;}
```

#2 using uninitialized class data

```
class aclass {
double fDat, fDat2;
aclass(){ double fDat = 0.; }
};
```

#1 the bug no-one has thought of yet

universitätbonn

End of Lecture 1

universitätbonn

BACKUP