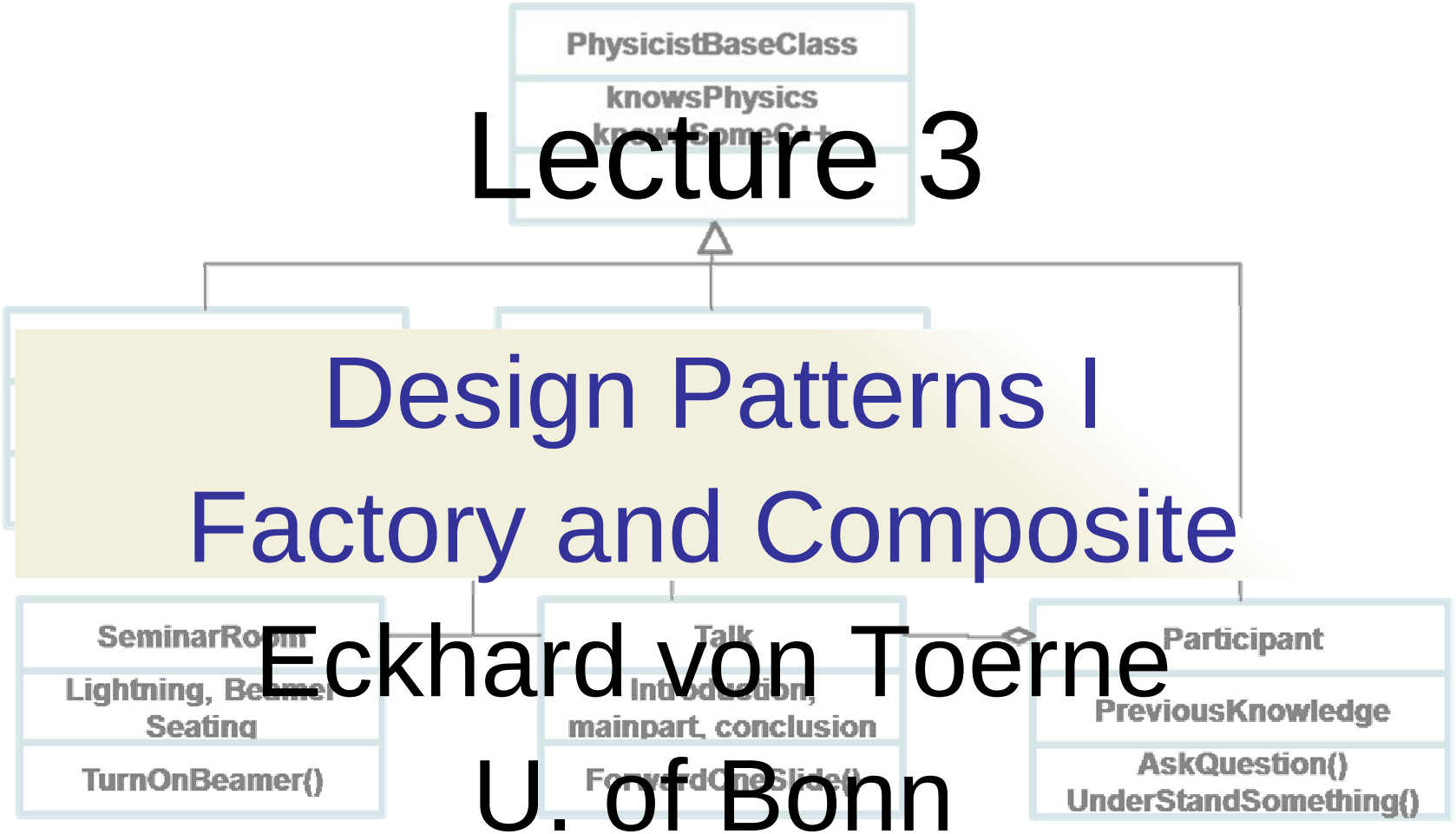
universitätbonn

Helmholtz Workshop

PHYSICS AT THE TERA SCALE Helmholtz Alliance

Advanced Methods of Software Programming



Lecture 3

Design Patterns I
Factory and Composite

Eckhard von Toerne
U. of Bonn

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

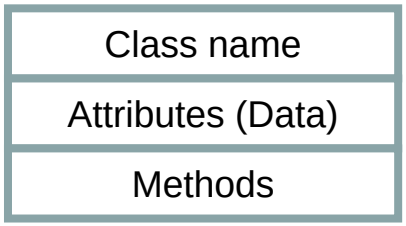
Design Pattern Categories

- Creational Patterns
- Structural Patterns
- Behavioral Patterns

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

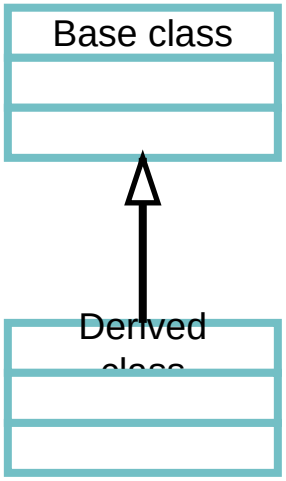
Summary
Syntax of UML Class Diagrams

Class representation

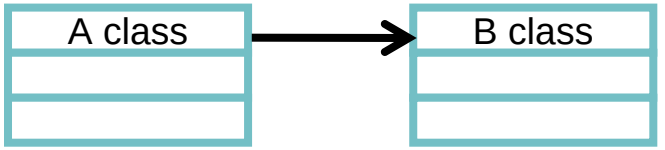
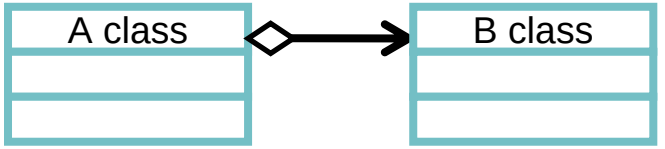
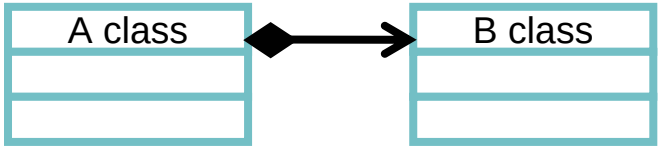
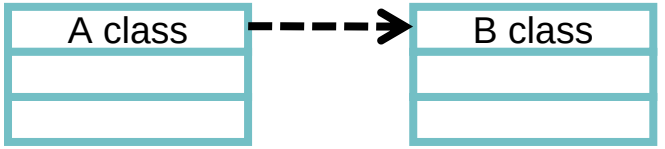


Relations among classes

Generalization (Inheritance)



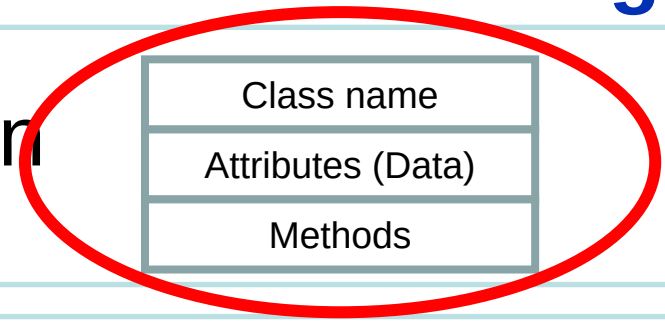
Association

Association (A knows B)	
Aggregation (B part of A)	
Composition **(B integral part of A)**	
Instantiation (A creates B)	

universitätbonn

Summary
Syntax of UML Class Diagrams

Class representation

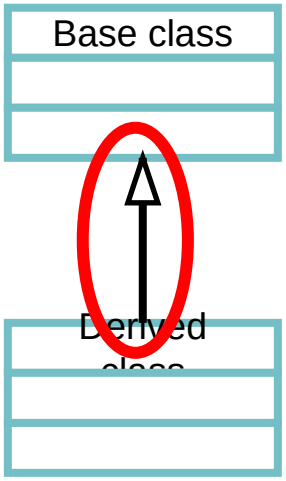


Relations among classes

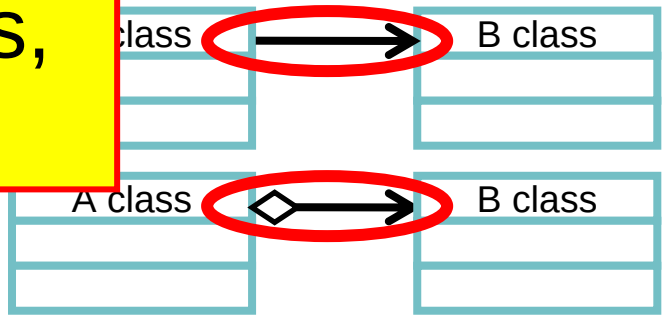
Generalizati… (Inheritance)

…ciation

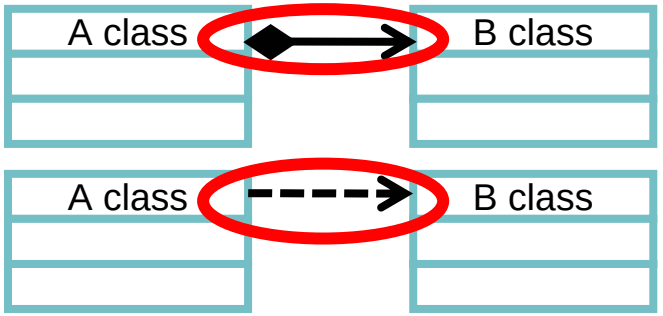
In Summary:
- 1 type of boxes,
- 5 types of arrows



Aggregation
(B part of A)



Composition
(B integral part of A)



Instantiation
(A creates B)

universitätbonn

Design Pattern Factory

universitätbonn

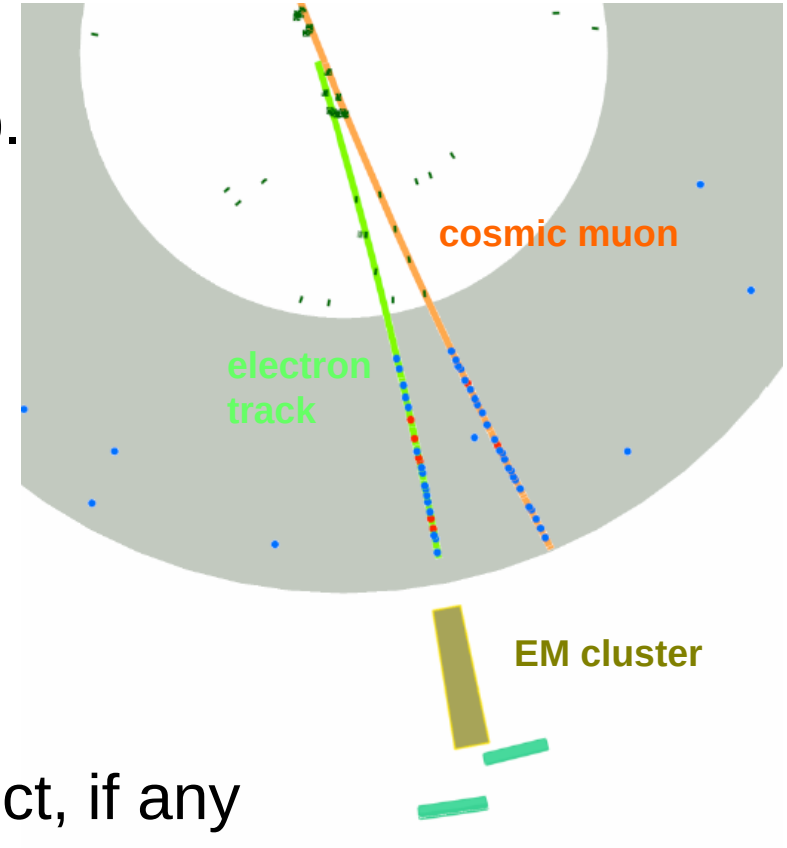
Motivation for Factory Pattern

How is an object stored in a file and read from a file?

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()

Writing objects to file

Break down objects into writeable pieces:

- int, float, string,
- arrays/vectors of writeable types
- Non-trivial: how to write pointers to objects

  → address of objects is dynamic, will be invalid when read

Usual procedure for writing pointers:

- each object gets unique ID
- write unique ID instead of pointer

- Basic problem here: how to decide if an object is no longer needed (persistency problem)

  → garbage collector

```
class Electron {
Cluster*  fCluster;
Track*    fTrack
Int       fUniqueID
} ;
```

```
Electron::Write(ostream out)
 {
out << fUniqueID;
out << fCluster->GetUniqueID();
out << fTrack->GetUniqueID()
} ;
```

universitätbonn

Reading objects from file

- Reading of basic objects: trivial
- Reading of pointers: convert ID back into pointer
- Creation of object usually divided into
  - creation of empty object
    object type unknown at compilation time, but known at execution time
    inherent problem solved by factory pattern
  - read information from file into empty object
- How to synchronize the creation and filling of objects
- How to avoid empty objects

Side remark: **ROOT offers functionality to store any object in ROOT files**

```
class Electron {
Cluster*  fCluster;
Track*    fTrack
Int       fUniqueID
} ;
```

```
Electron::Read(istream in){
in >> fUniqueID;
in >>clusterID;
in >> trackID;
// next:
// convert ID into pointers
}
```

```
ReadFile(istream in)
 {
string typename;
in << typname
BaseObject* obj =
   factory.Create(typname)
obj->Read(in);
}
```

A Toy Example: Objects stored in a file (in ascii)

```
--BEGINOFFILE
4                                   Number of objects in file
CalorimeterCellCollection           data object type stored as a string
101   3                             (UniqueID of collection, Number of entries in collection)
102   214205   44.506               (UniqueID of cell, DetectorID and pulseheight)
103   234756   15.533               (UniqueID, DetectorID and pulseheight)
104   234757   23.003               (UniqueID, DetectorID and pulseheight)

TrackerHitCollection
105   1                             (UniqueID, Number of entries in collection)
106   100787  59.284                (UniqueID, DetectorID and pulseheight)

CaloClusterCollection               (UniqueID, Number of entries in collection)
107   1
108   2  103 104                    (UniqueID, Number of Cells in cluster, list of CaloCell UniqueIDs)

ElectronCollection
109   1                             (UniqueID, Number of entries in collection)
110   108 106                       (UniqueID, CaloCluster UniqueID, Track Unique ID)
--EndOfFile
```

A Toy Example: Objects stored in a file (in ascii)

```
--BEGINOFFILE
4                                   Number of objects in file
CalorimeterCellCollection           data object type stored as a string
101   3                             (UniqueID of collection, Number of entries in collection)
102   214205   44.506               (UniqueID of cell, DetectorID and pulseheight)
103   234756   15.533               (UniqueID, DetectorID and pulseheight)
104   234757   23.003               (UniqueID, DetectorID and pulseheight)

TrackerHitCollection
105   1                             (UniqueID, Number of entries in collection)
106   100787   59.284               (UniqueID, DetectorID and pulseheight)

CaloClusterCollection               (UniqueID, Number of entries in collection)
107   1
108   2   103 104                   (UniqueID, Number of Cells in cluster, list of CaloCell UniqueIDs)

ElectronCollection
109   1                             (UniqueID, Number of entries in collection)
110   108 106                       (UniqueID, CaloCluster UniqueID, Track Unique ID)
--EndOfFile
```

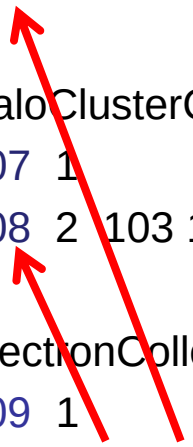
A Toy Example: Objects stored in a file (in ascii)

```
--BEGINOFFILE
4                                   Number of objects in file
CalorimeterCellCollection           data object type stored as a string
101   3                             (UniqueID of collection, Number of entries in collection)
102   214205   44.506               (UniqueID of cell, DetectorID and pulseheight)
103   234756   15.533               (UniqueID, DetectorID and pulseheight)
104   234757   23.003               (UniqueID, DetectorID and pulseheight)

TrackerHitCollection
105   1                             (UniqueID, Number of entries in collection)
106   100787   59.284               (UniqueID, DetectorID and pulseheight)

CaloClusterCollection               (UniqueID, Number of entries in collection)
107   1
108   2   103 104                   (UniqueID, Number of Cells in cluster, list of CaloCell UniqueIDs)

ElectronCollection
109   1                             (UniqueID, Number of entries in collection)
110   108 106                       (UniqueID, CaloCluster UniqueID, Track Unique ID)
--EndOfFile
```



universitätbonn

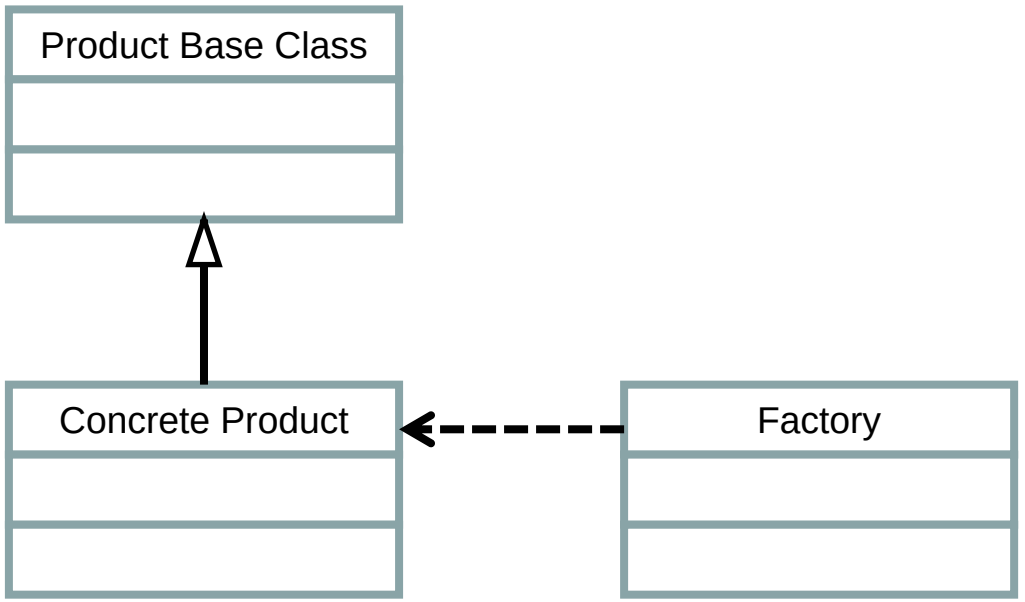
Factory Method Pattern

(Creational pattern)

universitätbonn

UML Diagram

FactoryMethod
Class Diagram



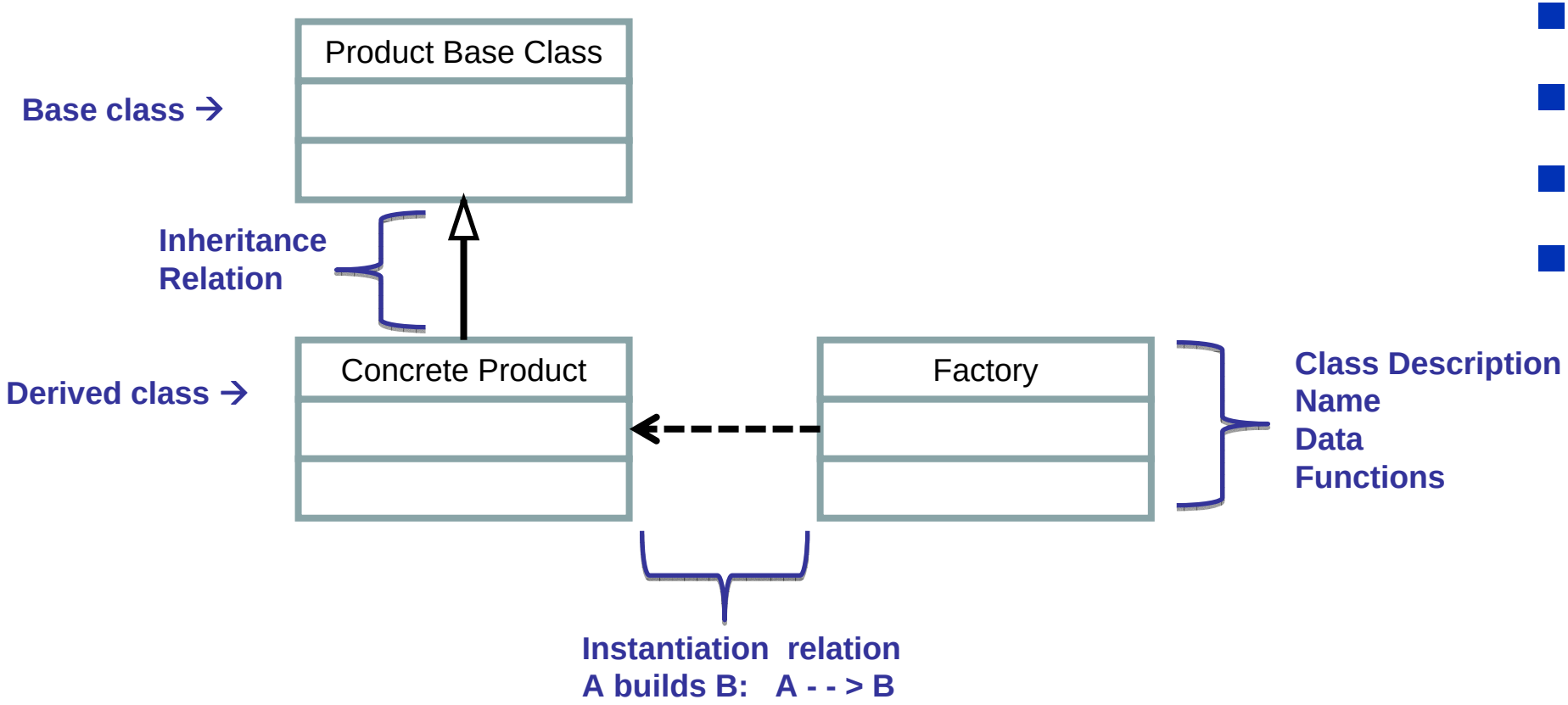
Unified Modeling Language: pictorial language used to model object oriented software

UML defines multitude of diagram types (see e.g. http://www.omg.org/spec/UML/2.2/)

In the following: only **Class diagrams** in a slightly modified version used (to be compatible with Design Pattern examples)

FactoryMethod Diagram

UML Class Diagram



universitätbonn

Factory Method Pattern Example:

Creator of Data Objects in Analysis Framework

The Creator of DataObjects

Reading Objects from a file:

```
string typname;
in >> typname;
DataObject* dat = fCreator.Create(typname); // Create invokes
   // correct constructor for derived DataObject
dat->Read(in);

DataObject* Create(string typname)
{
     if (typname == "DICalorimeterHit" ) {
       DICalorimeterHit* poi = new DICalorimeterHit();
       return dynamic_cast<DataItem*>( poi );
     }
     else if (typname == "DIEventNumber" ) {
       DIEventNumber* poi = new DIEventNumber();
       return dynamic_cast<DataItem*>( poi );
     }
     else if (typname == "DITrackerHit" ) {
       DITrackerHit* poi = new DITrackerHit();
       return dynamic_cast<DataItem*>( poi );
     }
} // Factory design pattern (explicit implementation)
```

** dynamic_cast <type*> checks at run type if conversion is valid and only then returns a pointer of said type.

Who creates the creator?

A Meta-Solution to create Factory class

Factory class generated at compilation time from all available Data items, automatically adding any new DataItem classes.

scripts/creator.sh (a bash shell script)

```
FILENAME="DataItemCreator.h"
FILENAMECXX="DataItemCreator.cxx"
#create list of header files in which DATA classes are
HEADERFILES=`ls DI*.h`
CLASSLIST=`echo $HEADERFILES | sed s/.h//g`

cat <<EOF > $FILENAMECXX
#include "DataItemCreator.h"
EOF
for i in $HEADERFILES; do
    echo "#include \"$i\"" >> $FILENAMECXX
done
cat <<EOF >> $FILENAMECXX

DataItem* DataItemCreator::Create(TString name){

EOF
for i in $CLASSLIST; do
    echo "     if (name == \"$i\" ) {" >> $FILENAMECXX
    echo "       $i* poi = new $i();" >> $FILENAMECXX
    echo "       return dynamic_cast<DataItem*>( poi );" >> $FILENAMECXX
    echo "     }" >> $FILENAMECXX
done
cat <<EOF >> $FILENAMECXX
}
EOF
```

`command ` inserts result of a shell command into the script

Prints everything up to EOF into file

Prints everything up to EOF into file (append mode)

Loop over string list

universitätbonn

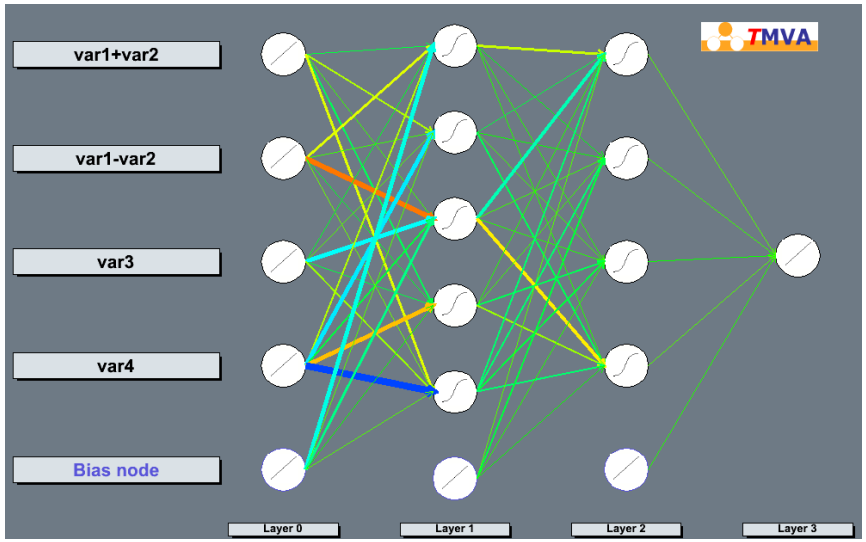
Factory Method Pattern Example:

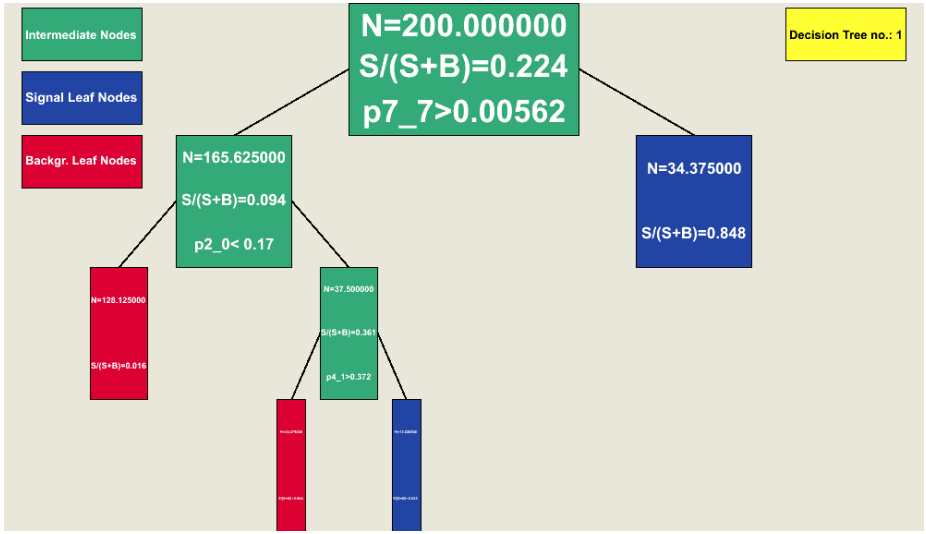
TMVA Method Factory

universitätbonn

TMVA

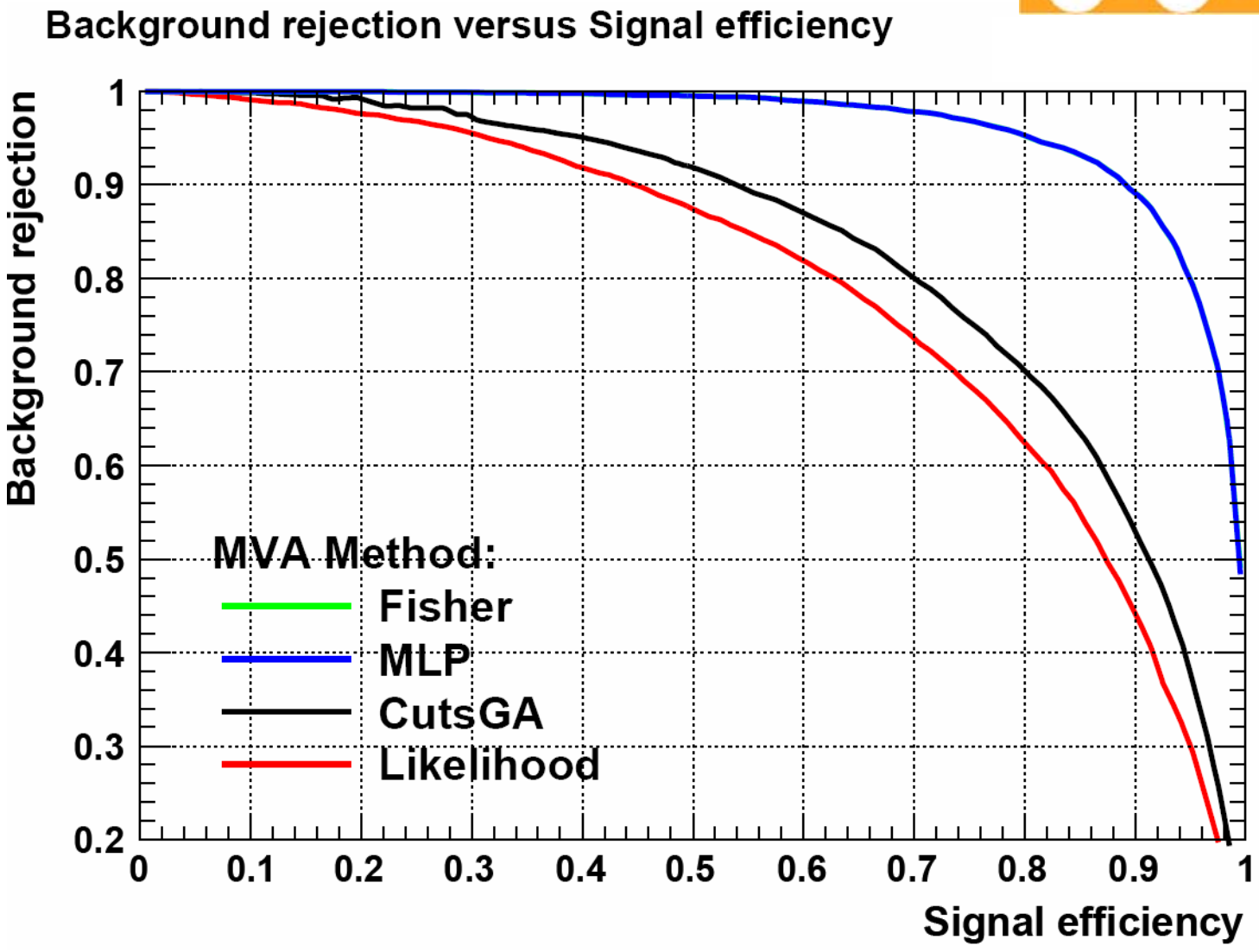
TMVA





Toolkit for **M**ulti**V**ariate **A**nalysis (A. Höcker, P. Speckmayer, J. Stelzer, J. Therhaag, H. Voss, E.v.Törne)

- ~15 different methods for classification or regression applications accessible via one Interface (the factory)
- Methods: ANN, BDT, linear classifiers, likelihoods, support vector machines, …
- Factory: creates methods and builds data sets
- Comparison of Methods in identical framework



- Comparison of Classification methods

TMVA factory

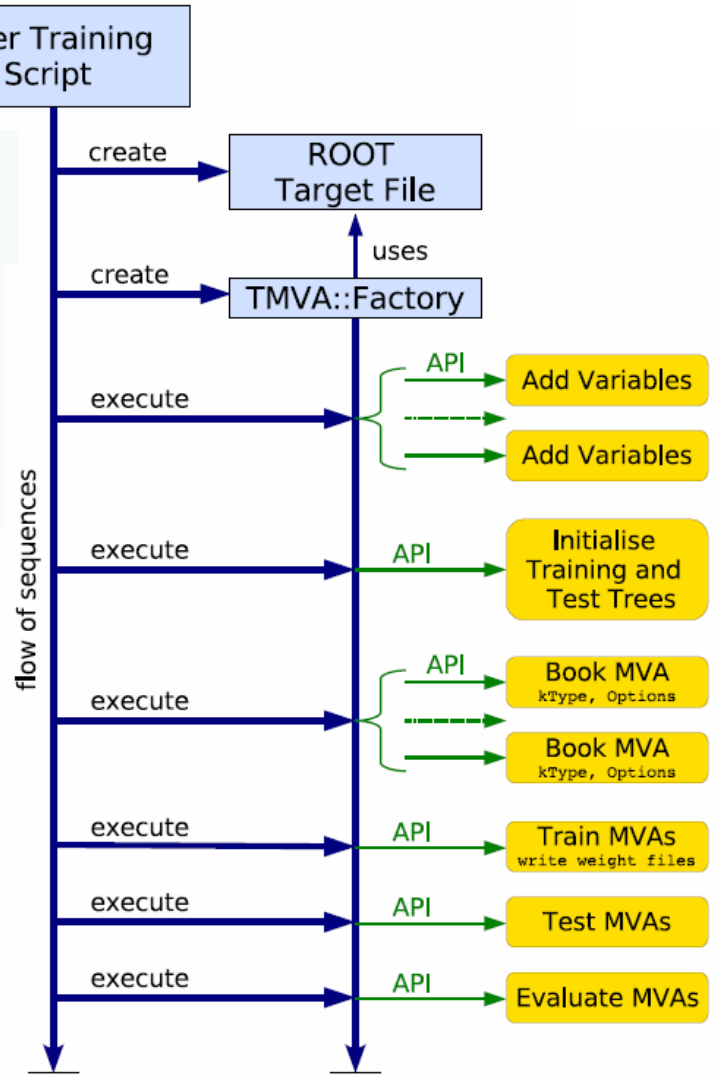
TMVA factory (J. Stelzer, A. Höcker)

Source code example:

```
factory->BookMethod( TMVA::Types::kMLP, "MLPBFGS",
    "H:!V:NeuronType=tanh:VarTransform=N:NCycles=600:HiddenLayers=N+5:\
     TestRate=5:TrainingMethod=BFGS" );

// Support Vector Machine
factory->BookMethod( TMVA::Types::kSVM, "SVM", "Gamma=0.25:Tol=0.001" );

// Boosted Decision Trees with adaptive boosting
factory->BookMethod( TMVA::Types::kBDT, "BDT",
    "!H:!V:NTrees=400:nEventsMin=400:MaxDepth=3:BoostType=AdaBoost:\
     SeparationType=GiniIndex:nCuts=20:PruneMethod=NoPruning" );
```



- Implementation follows A.Alexandrescu (Modern C++ Design),
- Factory contains (almost) no references to individual classes.
- Instead method registers itself with a singleton instance of a method repositorium.
- Registration is wrapped in a preprocessor macro.
- The neccessary code reduces to:

REGISTER_METHOD(“MyTMVAMethod“)

placed in front of MVA class declaration

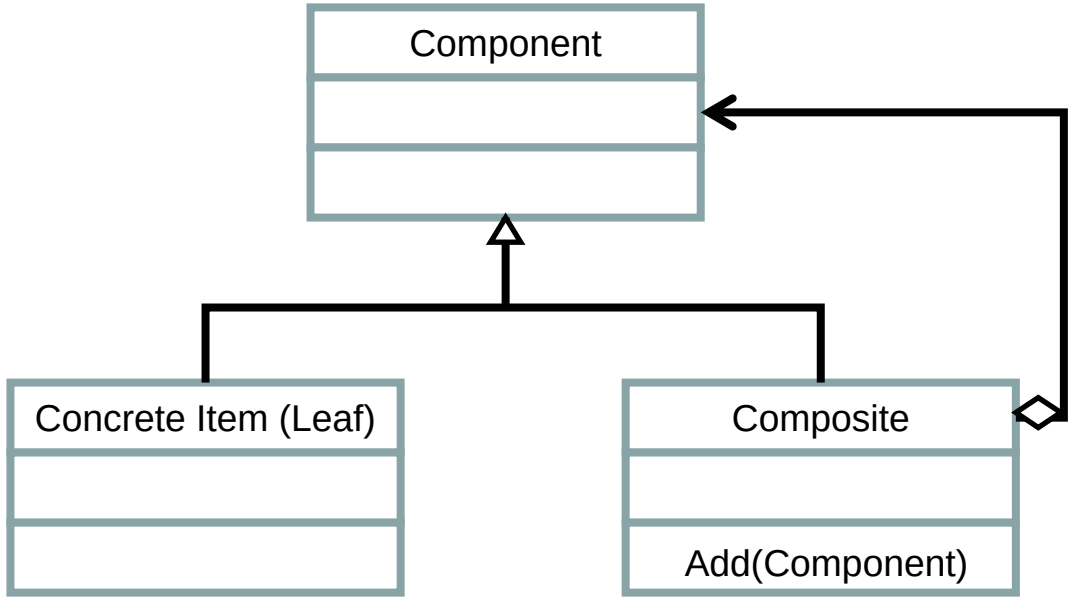
universitätbonn

Composite Pattern

(Structural pattern)

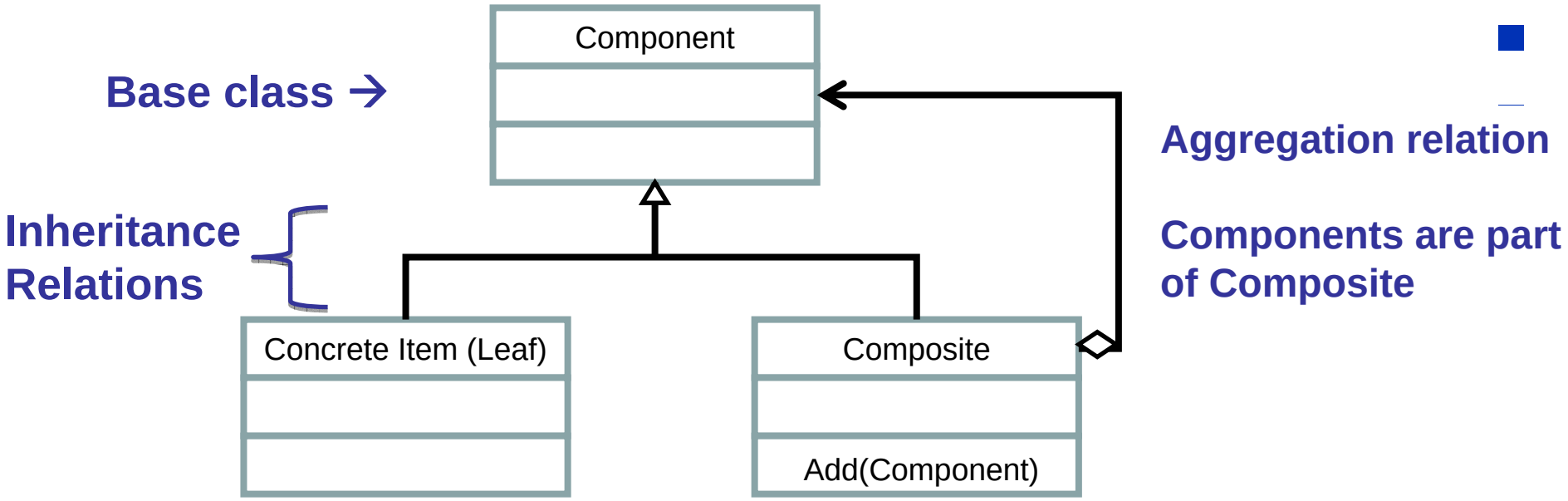
universitätbonn

Composite Class Diagram



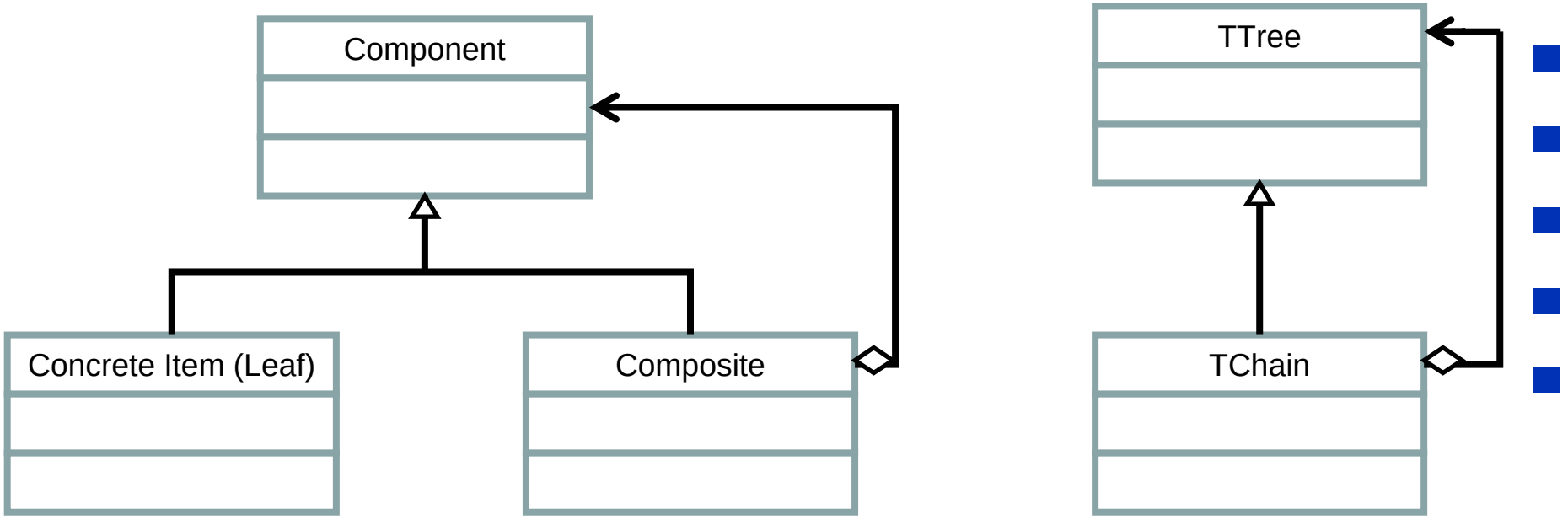
- Component: abstract class (interface)
- Basic objects: concreate Leaf(s)
- Composite: aggregations of components
- Use Composite::Add to add concrete items or composites
- Will discuss this pattern in detail in the next exercise

Composite Class Diagram



- Component: abstract class (interface)
- Basic objects: concreate Leaf(s)
- Composite: aggregations of components
- Use Composite::Add to add concrete items or composites
- Will discuss this pattern in detail in the next exercise

TTree and TChain



TChain in ROOT follows **not** Composite pattern

→Problem in maintainability (care necessary when adding new functions to TTree)

→Historical reasons,

→problems when trying to chain trees in the same file.

Summary and Conclusion

- Design pattern frequently appear in HEP source code
- Important to know when using and escpecially when designing object oriented code
- All examples given are in C++, design patterns are not language specific

universitätbonn

BACKUP

universitätbonn

HistogramService

How to make the histogram handling more efficient

- At the moment each histogram is handled at least in three far-away points in your code
  - Definition
  - Filling
  - Writing
- Real world example: turn this into a one point access using a service class