OOAD Examples

1. Jet Finder
2. Track Fitting
3. Shared Libraries
4. Sequence Diagram
5. Composite Pattern

1 Jet Finder

- Recombination jet finder in e^+e^-
  - Define *distance measure* y_{ij}
  - Define *combination procedure*
- The recombination algorithm:
  - calculate all y_{ij}
  - combine particles with smallest y_{ij} into a pseudo-particle, remove original (pseodo-) particles
  - stop if all $y_{ij} > y_{cut}$ or no (pseodo-) particles left

1 Jet Finder

Algorithm	Distance y_{ij}	Combination
E	$(p_i+p_j)^2/s$	p_i+p_j
JADE E0	$2E_iE_j(1-\cos\Theta_{ij})/s$	p_i+p_j
JADE P0	$(p_i+p_j)^2/s$	$E_k=E_i+E_j;\vec{p}_k=E_k\frac{\vec{p}_i+\vec{p}_j}{\lvert\vec{p}_i+\vec{p}_j\rvert}$
JADE P	$(p_i+p_j)^2/s$	$\vec{p}_k=\vec{p}_i+\vec{p}_j;E_k=\lvert\vec{p}_k\rvert$
Durham	$2\min(E_i^2,E_j^2)(1-\cos\Theta_{ij})/s$	p_i+p_j
Geneva	$\frac{8E_iE_j(1-\cos\Theta_{ij})}{9(E_i+E_j)^2}$	p_i+p_j
LUCLUS	$\frac{2\lvert\vec{p}_i\rvert\lvert\vec{p}_j\rvert\sin(\Theta_{ij}/2)}{(\lvert\vec{p}_i\rvert+\lvert\vec{p}_j\rvert)}$	p_i+p_j

1 Jet Finder

- Design classes for a jet finder package
- Input is vector<FourVector*>
- User can query
  - Number of jets for given y_{cut}
  - Value of y_{cut} when # of jets changes N-1 → N
  - Association of input 4-vectors with jets
  - Jet 4-vectors for given y_{cut}

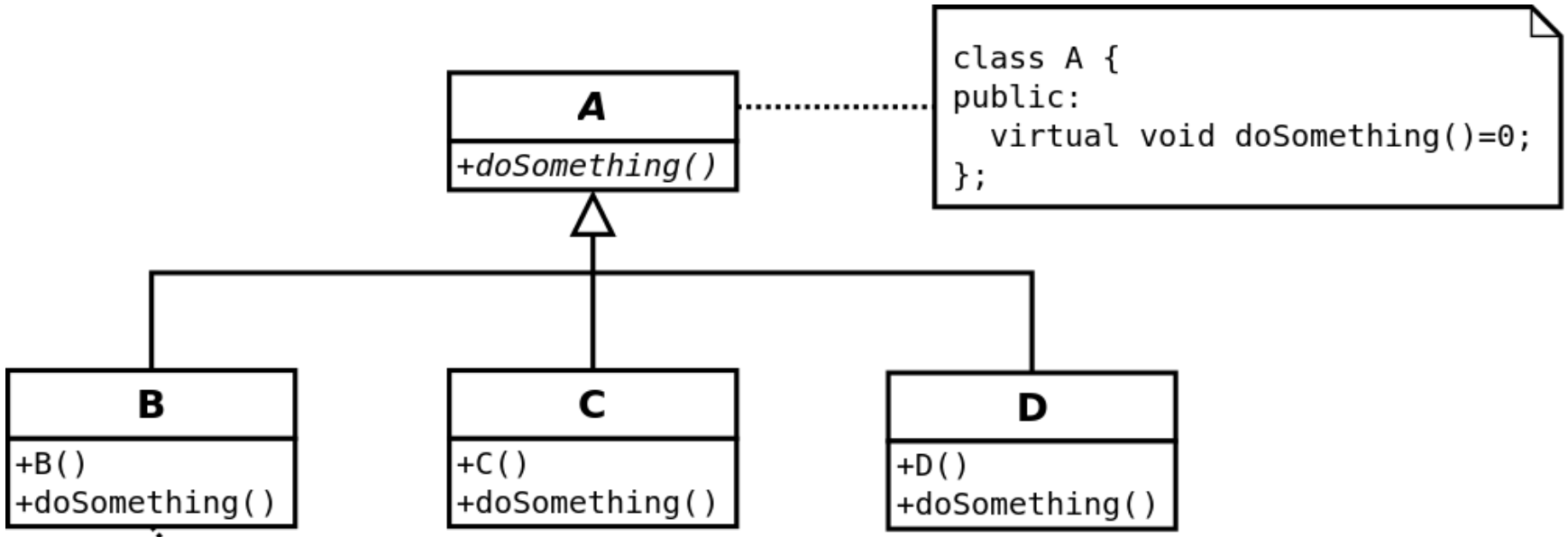
2 Track Fitting

- A typical HEP detector
  - tracking subdetectors with 2D or 3D readout
  - a uniform magnetic field (with small defects)
- Hit finding solved for each subdetector
  - have vector<SVTHit>, vector<DCHHit>
- Want several track fit algorithms, e.g.
  - Simple 5-parameter helix fit
  - a Kalman filter
  - but don't worry now about algorithm details

2 Track Fitting

- Design classes for track fitting code
- Start from hits of individual subdetectors
  - avoid direct coupling of concrete hit classes to track fitting classes
- Output is an object with methods for
  - momentum vector along trajectory
  - error matrix
  - start and end point
  - fit quality
  - hit association

3 Sequence Diagram




```
class B: public A {
public:
  void doSomething() {
    cout<<"I am B"<<endl;
  }
};
```

```
int main () {
  vector<A*> va;
  va.push_back( new B() );
  va.push_back( new C() );
  va.push_back( new D() );
  for( int i=0; i<va.size(); i++ ){
    va[i]->doSomething();
  }
}
```

Draw the sequence diagram showing the actions in main

4 Creating objects from DLLs

- An application (e.g. Athena) loads DLLs
- How can we create objects from the DLL?
  - direct creation not possible → want interchangeable DLLs
- DLL loaded via dlopen at run-time
  - need a mechanism to create objects once a DLL is loaded

5 Composite Pattern

- Calculate systematic uncertainty
  - Different recipies for groups of sources
    - Largest deviation (σ_1, σ_2, σ_3)
    - Average deviation (σ_4, σ_5, σ_6)
  - Add intermediate results in quadrature
- Use Composite pattern
  - Generic solution
  - Construct objects to configure calculation