Analysis

Geant4 CERN Tutorial

Wednesday, 17 February 2010

In This Lecture

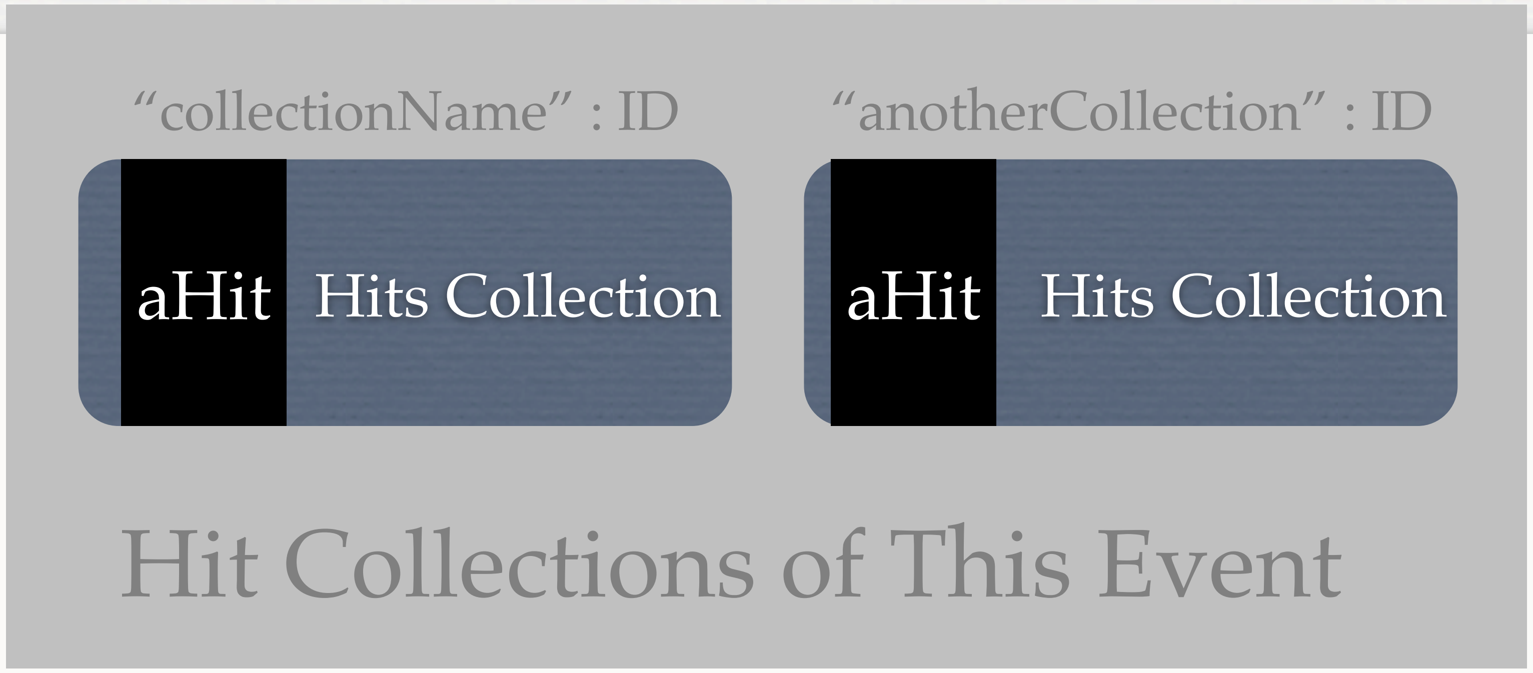
- How to retrieve Hits

Important: this material refers to the "Sensitive Detector" and "Hits" lectures

Retrieving Hits

- In previous lecture we have created and filled hits
- We now want to retrieve back the hit container and show how to do analysis of the collected data
  - For this exercise we show a simple example: fill histograms
- Note: Geant4 toolkit does not provide an internal analysis framework, you are free to use your favorite (examples: AIDA, ROOT, Mathematica, Excel,...)
  - You need to “integrate” your analysis framework with your application. We use ROOT as an example

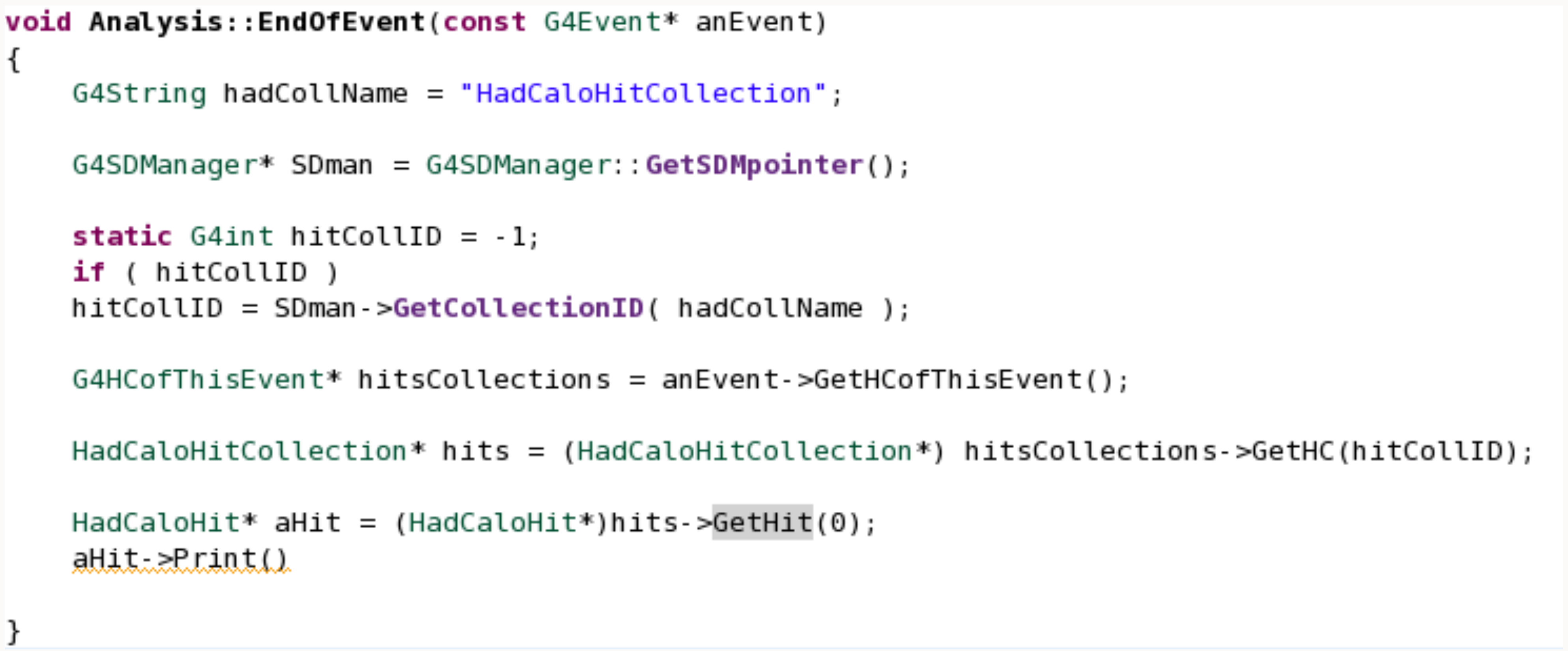
Retrieving Hits



- Reminder: Hits are stored in a hits collection and added to this event's hit collection
- In this example we have (a non G4) user class called Analysis; we implement in this class the retrieval of hits and filling of histograms

Retrieving Hits

1. Remember: hits collections have names, but you need the (G4 assigned) unique ID
2. Given collection name ask G4 for the ID (time consuming, do once!)
3. Get the hit collection

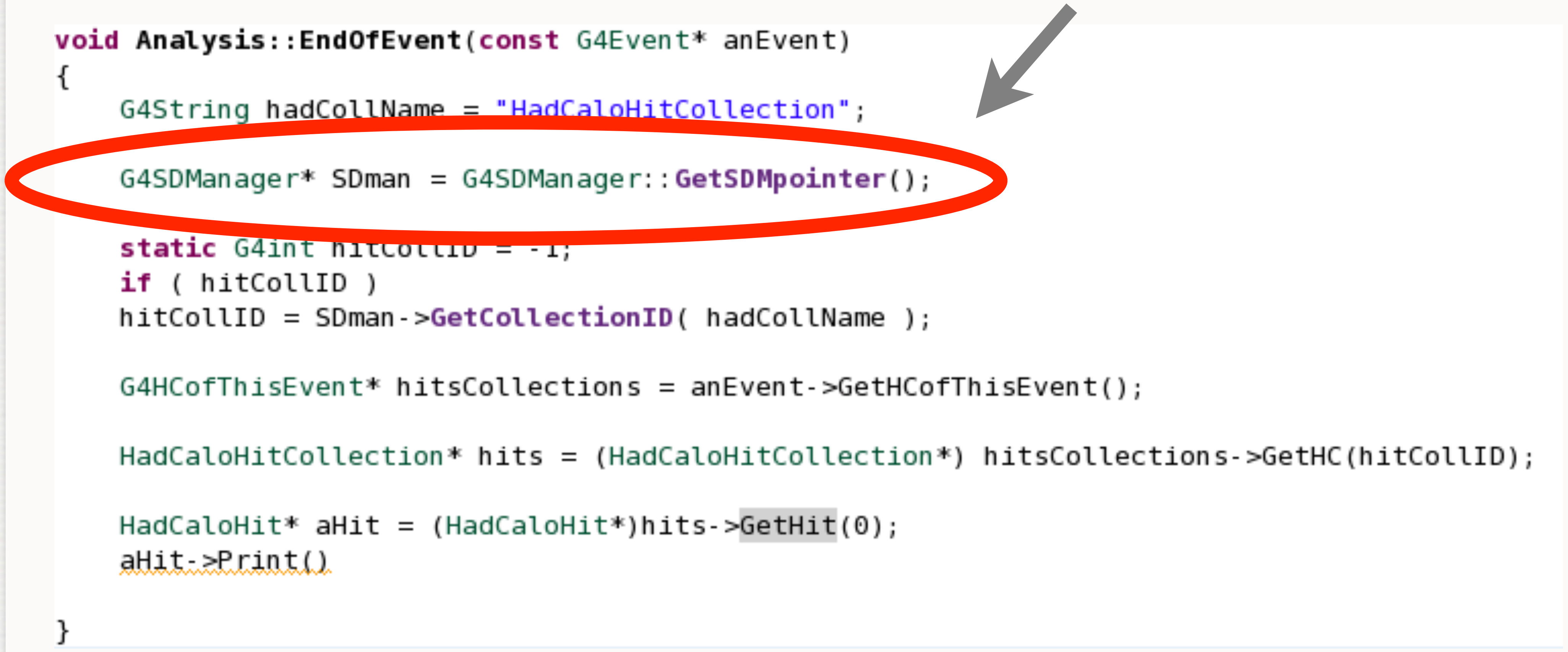


```
void Analysis::EndOfEvent(const G4Event* anEvent)
{
    G4String hadCollName = "HadCaloHitCollection";

    G4SDManager* SDman = G4SDManager::GetSDMpointer();

    static G4int hitCollID = -1;
    if ( hitCollID )
    hitCollID = SDman->GetCollectionID( hadCollName );

    G4HCofThisEvent* hitsCollections = anEvent->GetHCofThisEvent();

    HadCaloHitCollection* hits = (HadCaloHitCollection*) hitsCollections->GetHC(hitCollID);

    HadCaloHit* aHit = (HadCaloHit*)hits->GetHit(0);
    aHit->Print()

}
```

Retrieving Hits

1. Remember: hits collections have names, but you need the (G4 assigned) unique ID
2. Get the ID from collection name (time consuming, do once!)
3. Get the hit collection

Get a pointer to the Sensitive Detector manager

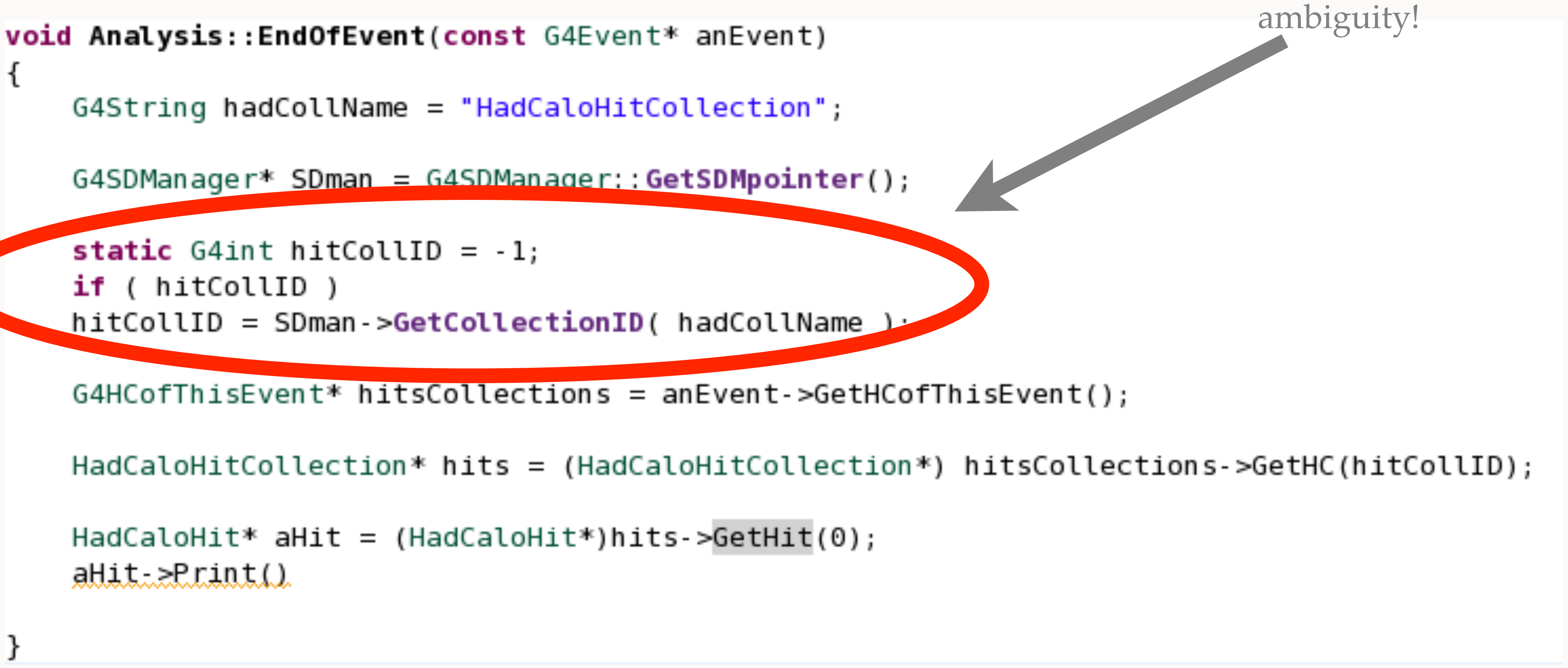


```
void Analysis::EndOfEvent(const G4Event* anEvent)
{
    G4String hadCollName = "HadCaloHitCollection";

    G4SDManager* SDman = G4SDManager::GetSDMpointer();

    static G4int hitCollID = -1;
    if ( hitCollID )
    hitCollID = SDman->GetCollectionID( hadCollName );

    G4HCofThisEvent* hitsCollections = anEvent->GetHCofThisEvent();

    HadCaloHitCollection* hits = (HadCaloHitCollection*) hitsCollections->GetHC(hitCollID);

    HadCaloHit* aHit = (HadCaloHit*)hits->GetHit(0);
    aHit->Print()

}
```

Retrieving Hits

1. Remember: hits collections have names, but you need the (G4 assigned) unique ID
2. Get the ID from collection name (time consuming, do once!)
3. Get the hit collection

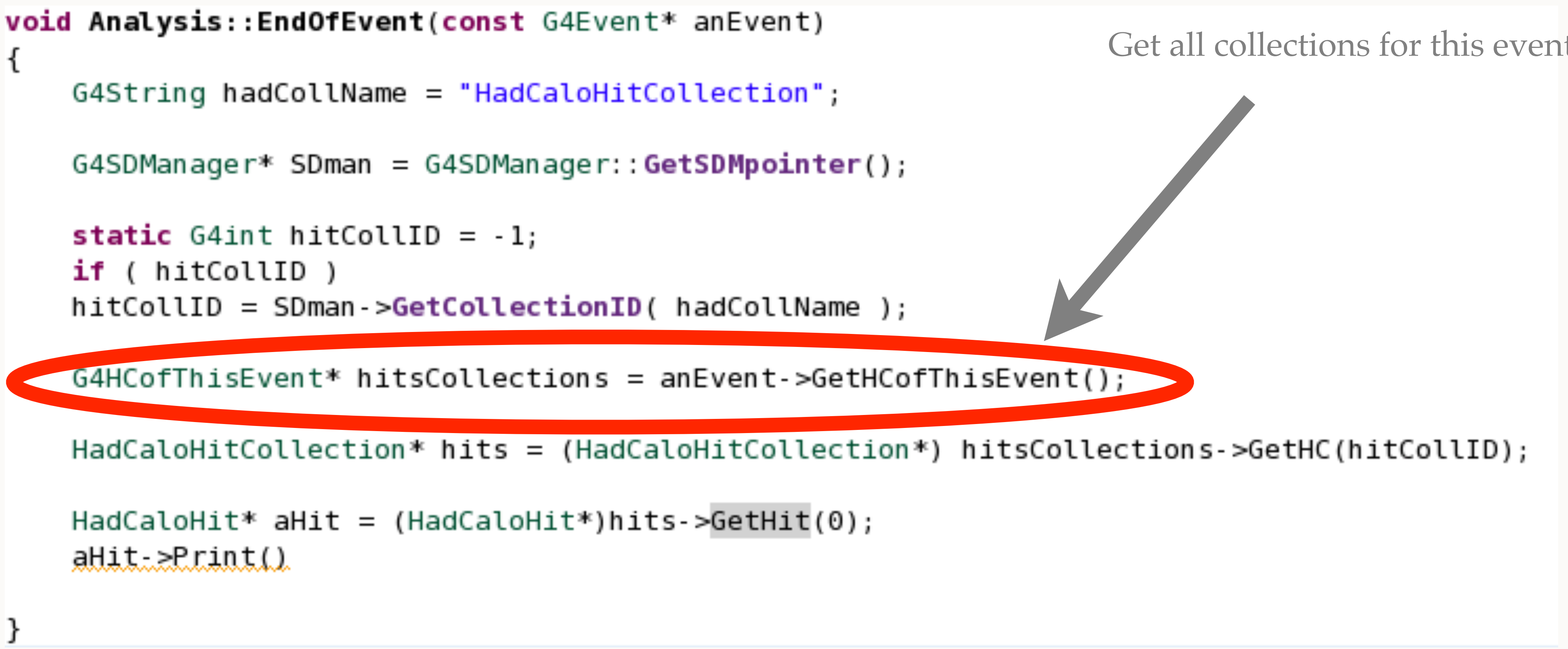
Get collection ID; very important: the collection name is:
/HadCalo/HadCaloHitCollection
this works only because there is only one HC and there is no ambiguity!



```
void Analysis::EndOfEvent(const G4Event* anEvent)
{
    G4String hadCollName = "HadCaloHitCollection";

    G4SDManager* SDman = G4SDManager::GetSDMpointer();

    static G4int hitCollID = -1;
    if ( hitCollID )
    hitCollID = SDman->GetCollectionID( hadCollName );

    G4HCofThisEvent* hitsCollections = anEvent->GetHCofThisEvent();

    HadCaloHitCollection* hits = (HadCaloHitCollection*) hitsCollections->GetHC(hitCollID);

    HadCaloHit* aHit = (HadCaloHit*)hits->GetHit(0);
    aHit->Print()

}
```

Retrieving Hits

1. Remember: hits collections have names, but you need the (G4 assigned) unique ID
2. Get the ID from collection name (time consuming, do once!)
3. Get the hit collection



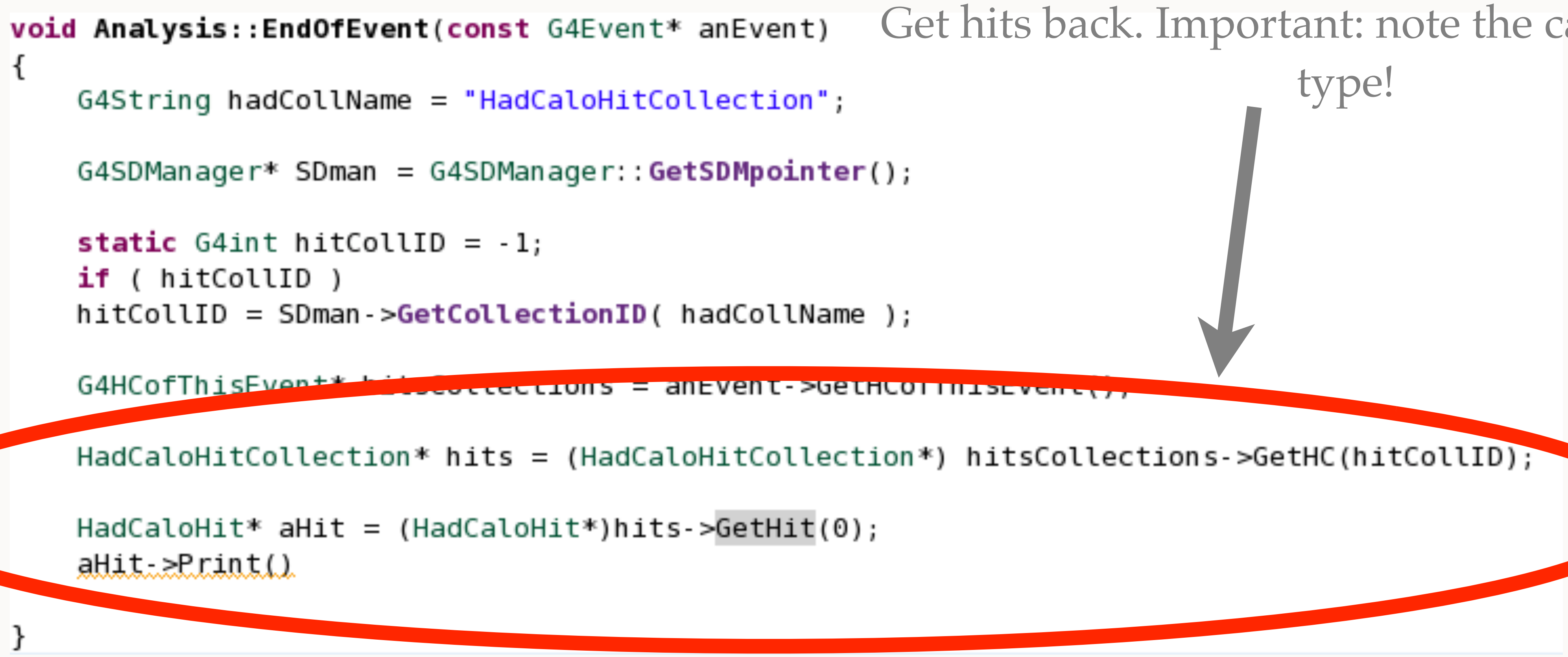
```
void Analysis::EndOfEvent(const G4Event* anEvent)
{
    G4String hadCollName = "HadCaloHitCollection";

    G4SDManager* SDman = G4SDManager::GetSDMpointer();

    static G4int hitCollID = -1;
    if ( hitCollID )
    hitCollID = SDman->GetCollectionID( hadCollName );

    G4HCofThisEvent* hitsCollections = anEvent->GetHCofThisEvent();

    HadCaloHitCollection* hits = (HadCaloHitCollection*) hitsCollections->GetHC(hitCollID);

    HadCaloHit* aHit = (HadCaloHit*)hits->GetHit(0);
    aHit->Print()

}
```

Get all collections for this event

Not shown here: always check pointers to be !
=0, a zero pointer means: "not found"

Retrieving Hits

1. Remember: hits collections have names, but you need the (G4 assigned) unique ID
2. Get the ID from collection name (time consuming, do once!)
3. Get the hit collection



```
void Analysis::EndOfEvent(const G4Event* anEvent)
{
    G4String hadCollName = "HadCaloHitCollection";

    G4SDManager* SDman = G4SDManager::GetSDMpointer();

    static G4int hitCollID = -1;
    if ( hitCollID )
    hitCollID = SDman->GetCollectionID( hadCollName );

    G4HCofThisEvent* hitsCollections = anEvent->GetHCofThisEvent();

    HadCaloHitCollection* hits = (HadCaloHitCollection*) hitsCollections->GetHC(hitCollID);

    HadCaloHit* aHit = (HadCaloHit*)hits->GetHit(0);
    aHit->Print()

}
```

Get hits back. Important: note the cast to correct type!

Exercise shows how to "loop" on all hits

Adding Analysis Toolkits

- To add an analysis toolkit you will need to:
  - Modify makefiles to instruct compilation that you want “extra” functionalities
  - Add appropriate headers in source code
  - Use toolkit functionalities to produce histograms/ntuple etc
- Exercises show hot to use ROOT
- To use AIDA see $G4INSTALL/examples/extended/analysis
- Warning: commercial products’ licenses in general do not allow for (open and free) integration in your application. Instead write out a simple text file that can be “read in” by analysis toolkit (ASCII, CSV, XML, ...)

Exercise

- task4d
  - Retrieve hits for analysis
  - Add ROOT toolkit to application
  - Fill histograms
- cd g4course/task4/task4d

A Note

- Analysis class is not part of Geant4, thus the interface methods (Analsysis::PrepareNewEvent, ::EndOfEvent, etc) are not called by Geant4 "kernel". We use corresponding UserAction classes to call this methods

- You may also implement histogram filling and analysis code directly in UserAction, however it is always a good idea to "isolate" the analysis code in one (or more) well defined classes. If the complexity of your application increases it will be much easier to adapt the simulation and analysis code to your new needs

Ntuples

- task2a contains the code in src/RootSaver.cc to write a ROOT ntuple from hits and digits