```
//////////////////////////////////////////////////////////
// This class has been automatically generated on
// Mon Nov 25 10:41:51 2013 by ROOT version 5.34/09
// from TTree EventTree/The Tree with the variable used to performe the calculat
ion of energy deposition on the HEPD detector
// found on file: Electron5MeV_4M.root
//////////////////////////////////////////////////////////

#ifndef AnalyzerMulti_h
#define AnalyzerMulti_h

#include <TROOT.h>
#include <TChain.h>
#include <TFile.h>
#include <TH1F.h>
#include <TH2F.h>
#include <TProfile.h>
#include <vector>

// Header file for the classes stored in the TTree if any.

/*
#include "/Users/ambroglini/Simulation/CSES/hepd/source/dataformats/include/HEPD
RootEvent.hh"
#include "/Users/ambroglini/Simulation/CSES/hepd/source/dataformats/include/Root
Track.hh"
#include "/Users/ambroglini/Simulation/CSES/hepd/source/dataformats/include/Calo
RootHit.hh"
#include "/Users/ambroglini/Simulation/CSES/hepd/source/dataformats/include/Trac
kerRootHit.hh"
*/
#include "/wduser/sw64/hepd/hepd/source/dataformats/include/HEPDRootEvent.hh"
#include "/wduser/sw64/hepd/hepd/source/dataformats/include/RootTrack.hh"
#include "/wduser/sw64/hepd/hepd/source/dataformats/include/CaloRootHit.hh"
#include "/wduser/sw64/hepd/hepd/source/dataformats/include/TrackerRootHit.hh"



#include <TObject.h>
#include <TVector3.h>


class AnalyzerMulti {
public :
   TTree          *fTree;   //!pointer to the analyzed TTree or TChain
   Int_t           fCurrent; //!current Tree number in a TChain
   TProfile*          layerDepthElectron;
   TProfile*          layerDepthProton;
   TProfile*          energyAVGRangeElectron;
   TProfile*          energySUMRangeElectron;
   TProfile*          energyAVGRangeProton;
   TProfile*          energySUMRangeProton;
   // Declaration of leaf types
   HEPDRootEvent   *Event;

   // List of branches
   TBranch        *b_Event;   //!

   Bool_t checkPos;
   Bool_t checkTheta;
   Double_t Xlimit,Ylimit,Thetalimit;

   float theERange;
   std::vector<float> electronLayerMin;
   std::vector<float> electronLayerMax;
   std::vector<float> electronEnergy;
   std::vector<float> electronAcceptance;

   std::vector<float> protonLayerMin;
```

```
   std::vector<float> protonLayerMax;
   std::vector<float> protonEnergy;
   std::vector<float> protonAcceptance;

   Int_t numbCaloLayer,numbLayerScint,numbLayerCrystal;

   AnalyzerMulti();
   virtual ~AnalyzerMulti();
   virtual void     SetFile(TString fileName);
   virtual Int_t    GetEntry(Long64_t entry);
   virtual Long64_t LoadTree(Long64_t entry);
   virtual void     InitFullHisto();
   virtual void     CloseFullHisto();
   virtual void     Init(TTree *tree);
   virtual void     LoopElectron(float ERange);
   virtual void     LoopProton(float ERange);
   virtual Bool_t   Notify();
   virtual void     Show(Long64_t entry = -1);
   virtual Double_t ComputeAngle(std::vector<TrackerRootHit>& myTkHit);
   virtual Double_t ComputeAngleWithSmearing(std::vector<TrackerRootHit>& myTkHi
t,Double_t delta);
   virtual void     SetThetaAcceptance(Double_t theta); //in degree
   virtual void     SetAcceptanceWindows(Double_t X, Double_t Y); //in millimite
rs
   virtual void     SetCalorimeterConfiguration(Int_t nCalo,Int_t nScint,Int_t n
Crystal);
};

#endif

#ifdef AnalyzerMulti_cxx
AnalyzerMulti::AnalyzerMulti() : fTree(0)
{
  checkPos=false;
  Xlimit = 0;
  Ylimit = 0;
  numbCaloLayer=20;
  numbLayerScint=20;
  numbLayerCrystal=0;
}
AnalyzerMulti::~AnalyzerMulti()
{
   if (!fTree) return;
   delete fTree->GetCurrentFile();
}

void AnalyzerMulti::SetFile(TString fileName)
{
  // if parameter tree is not specified (or zero), connect the file
  // used to generate this class and read the Tree.
  TFile *f = (TFile*)gROOT->GetListOfFiles()->FindObject(fileName);
  if (!f || !f->IsOpen()) {
    f = new TFile(fileName);
  }
  fileName.Append(":/HEPD");
  TDirectory * dir = (TDirectory*)f->Get(fileName);
  dir->GetObject("EventTree",fTree);
  Init(fTree);

}


Int_t AnalyzerMulti::GetEntry(Long64_t entry)
{
// Read contents of entry.
   if (!fTree) return 0;
   return fTree->GetEntry(entry);
}
Long64_t AnalyzerMulti::LoadTree(Long64_t entry)
```

```
{
// Set the environment to read one entry
   if (!fTree) return -5;
   Long64_t centry = fTree->LoadTree(entry);
   if (centry < 0) return centry;
   if (fTree->GetTreeNumber() != fCurrent) {
      fCurrent = fTree->GetTreeNumber();
      Notify();
   }
   return centry;
}

void AnalyzerMulti::Init(TTree *tree)
{
  // The Init() function is called when the selector needs to initialize
  // a new tree or chain. Typically here the branch addresses and branch
  // pointers of the tree will be set.
  // It is normally not necessary to make changes to the generated
  // code, but the routine can be extended by the user if needed.
  // Init() will be called many times when running on PROOF
  // (once per file to be processed).

  // Set branch addresses and branch pointers
  Event = 0;
  if (!tree) return;
  fTree = tree;
  fCurrent = -1;

  fTree->SetBranchAddress("Event", &Event, &b_Event);
  Notify();
}

Bool_t AnalyzerMulti::Notify()
{
   // The Notify() function is called when a new file is opened. This
   // can be either for a new TTree in a TChain or when when a new TTree
   // is started when using PROOF. It is normally not necessary to make changes
   // to the generated code, but the routine can be extended by the
   // user if needed. The return value is currently not used.

   return kTRUE;
}

void AnalyzerMulti::Show(Long64_t entry)
{
// Print contents of entry.
// If entry is not specified, print current entry
   if (!fTree) return;
   fTree->Show(entry);
}
#endif // #ifdef AnalyzerMulti_cxx
```