```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration.  It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license .  These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use.  Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//

#ifndef CaloHit_h
#define CaloHit_h 1

#include "G4VHit.hh"
#include "G4THitsCollection.hh"
#include "G4Allocator.hh"

class CaloHit : public G4VHit
{
public:

  CaloHit();
  //  CaloHit(G4String volume);
  CaloHit(G4int DetID);
  ~CaloHit();
  CaloHit(const CaloHit &right);
  const CaloHit& operator=(const CaloHit &right);
  G4int operator==(const CaloHit &right) const;

  inline void *operator new(size_t);
  inline void operator delete(void *aHit);

  void Draw();
  void Print();

private:
  //  G4String VolumeID;
  G4int DetID;
  G4double totalEdep;
  std::map<G4int,G4double> edep;

public:
  //  inline void SetVolumeID(G4String volume)
  inline void SetDetID(G4int det)
  {
    //    VolumeID = volume;
    DetID = det;
  }
  void SetEdep(G4double de,G4int tkID);
  void AddEdep(G4double de,G4int tkID);
  inline G4double GetTotalEdep()
  { return totalEdep; }
  inline std::map<int,double> GetEdepMap(){return edep;}
```



```
  G4double GetEdep(G4int tkID);
  //  inline const G4String GetVolume()
  inline G4int GetVolume()
  { return DetID; }
  //  { return VolumeID; }

};

typedef G4THitsCollection<CaloHit> CaloHitsCollection;

extern G4Allocator<CaloHit> CaloHitAllocator;

inline void* CaloHit::operator new(size_t)
{
  void *aHit;
  aHit = (void *) CaloHitAllocator.MallocSingle();
  return aHit;
}

inline void CaloHit::operator delete(void *aHit)
{
  CaloHitAllocator.FreeSingle((CaloHit*) aHit);
}

#endif
```