```
// ===============================================================================
//
// Developed by Bryan V. Egner, Darren E. Holland, and Julie V. Logan
// Modified by Darren Holland 2020-11-02
// ===============================================================================
//
// This file is the main Geant4 .cc file for the RSM Geant4 Package. Compiling
this
// code for a design will calculate and return the mass.
// ===============================================================================
//
#include "B4DetectorConstruction.hh"
#include "B4aActionInitialization.hh"
#include "G4SystemOfUnits.hh"
#ifdef G4MULTITHREADED
#include "G4MTRunManager.hh"
#else
#include "G4RunManager.hh"
#endif
#include "G4UImanager.hh"
#include "G4UIcommand.hh"
#include "FTFP_BERT.hh"    // Photon PHysics -->  Recommended for HEP, Bertinia
Cascade
#include "Randomize.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"
#include <math.h>
#include <cmath>
#include <stdio.h>
#include <string>
#include <sstream>
#include <stdlib.h>
#include <time.h>
#include <vector>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <iomanip>
#include "Settings.hh"

// ===============================================================================
//
// Load any default command options
// ===============================================================================
//
using namespace std;

namespace {
  void PrintUsage() {
    G4cerr << " Usage: " << G4endl;
    G4cerr << " myMesh [-m macro ] [-u UIsession] [-t nThreads]" << G4endl;
    G4cerr << "   note: -t option is available only for multi-threaded mode."
           << G4endl;
  }
}

// ===============================================================================
//
// Begin Main Program
// ===============================================================================
//
int main(int argc,char** argv)
{
    // Evaluate arguments:
      // Load any macros:
    if ( argc > 7 ) {
      PrintUsage();
      return 1;
    }
    G4String macro;

    // Set default number of threads to use:
    #ifdef G4MULTITHREADED
      G4int nThreads = 4;
    #endif
    for ( G4int i=1; i<argc; i=i+2 ) {
        if      ( G4String(argv[i]) == "-m" ) macro = argv[i+1];
    #ifdef G4MULTITHREADED
        else if ( G4String(argv[i]) == "-t" ) {
          nThreads = G4UIcommand::ConvertToInt(argv[i+1]);
        }
    #endif
        else {
          PrintUsage();
          return 1;
        }
    }

    // Construct the default run manager:
    #ifdef G4MULTITHREADED
      auto runManager = new G4MTRunManager;
      if ( nThreads > 0 ) {
        runManager->SetNumberOfThreads(nThreads);
      }
    #else
      auto runManager = new G4RunManager;
    #endif

    // Set mandatory initialization classes for run manager:
    // Register detector:
    auto detConstruction = new B4DetectorConstruction();
    runManager->SetUserInitialization(detConstruction);

    // Register the physics list:
    auto physicsList = new FTFP_BERT(0);  // Photon Physics List
    runManager->SetUserInitialization(physicsList);

    // Register the user action initialization - SetUserAction of
    // PrimaryGeneratorAction, RunAction, EventAction, SteppingAction
    // are found in the Build function of this class
    auto actionInitialization = new B4aActionInitialization();
    runManager->SetUserInitialization(actionInitialization);

    // Get the pointer to the User Interface manager:
    auto UImanager = G4UImanager::GetUIpointer();

    // Initialize run manager
    UImanager->ApplyCommand("/run/initialize");
    //
=============================================================================== //
    // Job termination:
    // Free the store: user actions, physics_list and detector_description are
    // owned and deleted by the run manager, so they should not be deleted
    // in the main() program !
    //
=============================================================================== //
    delete runManager;
}
```