```cpp
// inheritance2.h
#include <string>
using namespace std;

enum Discipline { ARCHEOLOGY, BIOLOGY, COMPUTER_SCIENCE };
enum Classification { FRESHMAN, SOPHOMORE, JUNIOR, SENIOR };


// base class
class Person {
     protected:
            string name;    // protected member: derived classes can access
     public:
            Person() { setName(""); }
            Person(string pName) { setName(pName); }
            void setName(string pName) { name = pName; }
            string getName() const { return name; }
};


// Student "is-a" Person
class Student: public Person {
     private:
            Discipline major;
            Person *advisor;
     public:
            // Constructor
            Student(string sname, Discipline d, Person *adv);

            void setMajor(Discipline d) { major = d; }
            Discipline getMajor() const { return major; }
            void setAdvisor(Person *p) { advisor = p; }
            Person *getAdvisor() const { return advisor; }
};


// Faculty "is-a" Person
class Faculty: public Person {
     private:
            Discipline department;
     public:
            // Constructor with initialization list
            Faculty(string fname, Discipline d) : Person(fname) {
                  department = d;
            }

            void setDepartment(Discipline d) { department = d; }
            Discipline getDepartment( ) const { return department; }
};
```

```
// inheritance2.cpp
#include "inheritance2.h"

//*********************************************
// Constructor for the Student class.         *
//*********************************************
Student::Student(string sname, Discipline d, Person *adv)
                 : Person(sname) // Base constructor initialization {
    major = d;
    advisor = adv;
}
```

```
// This program demonstrates the use of the constructor
// initialization list to call a base constructor.

#include "inheritance2.h"
#include <iostream>
using namespace std;

// These arrays of string are used to print values
// of enumerated types.
const string dName[] = { "Archeology", "Biology", "Computer Science" };

const string cName[] = { "Freshman", "Sophomore", "Junior", "Senior" };

int main() {
   // Create  Faculty and Student objects
   Faculty prof("Indiana Jones", ARCHEOLOGY);
   Student st("Sean Bolster", ARCHEOLOGY, &prof);
   cout << "\nProfessor " << prof.getName() << " teaches "
        << dName[prof.getDepartment()] << "." << endl;

   // Get the student's advisor
   Person *pAdvisor = st.getAdvisor();
   cout << st.getName() <<"\'s advisor is "
        << pAdvisor->getName() << "." << endl << endl;

   return 0;
}
```

OUTPUT:

```
Professor Indiana Jones teaches Archeology.
Sean Bolster's advisor is Indiana Jones.
```