

```
1: ///////////////////////////////////////////////////  
2: // Grade.idl, Troy  
3:  
4:  
5: module Grade {  
6:   interface Grader {  
7:     boolean add_grade(in string tid, in string pwd, in float grade);  
8:     float show_grade(in string sid, in string pwd);  
9:   };  
10:  interface Security {  
11:    boolean check_teacher_pwd(in string tid, in string pwd);  
12:    boolean check_student_pwd(in string sid, in string pwd);  
13:  };  
14: };  
15:
```

```
1: //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
2: // Grader.C, Troy
3:
4: #include "GradeImpl.h"
5:
6: USE_STD_NS
7: int main(int argc, char* const* argv)
8: {
9:     try {
10:        CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
11:
12:        CORBA::Object_var obj = orb->resolve_initial_references("RootPOA");
13:        PortableServer::POA_var rootPOA = PortableServer::POA::_narrow(obj);
14:
15:        CORBA::PolicyList policies;
16:        policies.length(1);
17:        policies[(CORBA::ULong)0] = rootPOA->create_lifespan_policy(PortableServer::PERSISTENT);
18:
19:        PortableServer::POAManager_var poa_manager = rootPOA->the_POAManager();
20:        PortableServer::POA_var myPOA = rootPOA->create_POA("zhan_grader_poa",
21:                                                         poa_manager,
22:                                                         policies);
23:
24:        GraderImpl graderServant;
25:
26:        PortableServer::ObjectId_var graderId =
27:            PortableServer::string_to_ObjectId("zhan_Grader");
28:        myPOA->activate_object_with_id(graderId, &graderServant);
29:
30:        poa_manager->activate();
31:
32:        CORBA::Object_var reference = myPOA->servant_to_reference(&graderServant);
33:        cout << reference << " is ready" << endl;
34:
35:        orb->run();
36:    } catch(const CORBA::Exception& e) {
37:        cerr << e << endl;
38:        return 1;
39:    }
40:    return 0;
41: }
42:
43:
44:
45:
46:
```

```
1: ////////////////////////////////////////////////////
2: // Security.C, Troy
3:
4: #include "GradeImpl.h"
5:
6: USE_STD_NS
7: int main(int argc, char* const* argv)
8: {
9:     try {
10:
11:         CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
12:
13:         CORBA::Object_var obj = orb->resolve_initial_references("RootPOA");
14:         PortableServer::POA_var rootPOA = PortableServer::POA::_narrow(obj);
15:
16:         CORBA::PolicyList policies;
17:         policies.length(1);
18:         policies[(CORBA::ULong)0] = rootPOA->create_lifespan_policy(PortableServer::PERSISTENT);
19:
20:         PortableServer::POAManager_var poa_manager = rootPOA->the_POAManager();
21:         PortableServer::POA_var myPOA = rootPOA->create_POA("zhan_security_poa",
22:                 poa_manager,
23:                 policies);
24:
25:         SecurityImpl securityServant;
26:         PortableServer::ObjectId_var securityId =
27:             PortableServer::string_to_ObjectId("zhan_security");
28:         myPOA->activate_object_with_id(securityId, &securityServant);
29:         poa_manager->activate();
30:
31:         CORBA::Object_var reference = myPOA->servant_to_reference(&securityServant);
32:         cout << reference << " is ready" << endl;
33:         orb->run();
34:     } catch(const CORBA::Exception& e) {
35:         cerr << e << endl;
36:         return 1;
37:     }
38:     return 0;
39: }
```

```
1: //////////////////////////////////////
2: // Student.C, Troy
3:
4: #include "Grade_c.hh"
5:
6: USE_STD_NS
7:
8: int main(int argc, char* const* argv)
9: {
10:     try {
11:
12:         CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
13:         PortableServer::ObjectId_var graderId =
14:             PortableServer::string_to_ObjectId("zzhan_Grader");
15:
16:         Grade::Grader_var grader =
17:             Grade::Grader::_bind("/zzhan_grader_poa", graderId);
18:
19:         const char* sid = argc > 1 ? argv[1] : "1001";
20:         const char* pwd = argc > 2 ? argv[2] : "test";
21:
22:         CORBA::Float grade = grader->show_grade(sid,pwd);
23:         if (grade<0)
24:             cout<<"Student(CPP): Invalid SID or Password"<<endl;
25:         else
26:             cout << "Student(CPP): The grade of: "<<sid<<" is: "<<grade<<endl;
27:     } catch(const CORBA::Exception& e) {
28:         cerr << e << endl;
29:         return 1;
30:     }
31:     return 0;
32: }
```

```
1: //////////////////////////////////////
2: // Teacher.C, Troy
3:
4: #include "Grade_c.hh"
5: #include <stdlib.h>
6: #include <math.h>
7:
8: USE_STD_NS
9: int main(int argc, char* const* argv)
10: {
11:     try {
12:         CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
13:         PortableServer::ObjectId_var graderId =
14:             PortableServer::string_to_ObjectId("zzhan_Grader");
15:
16:         Grader_var grader =
17:             Grade::Grader::_bind("/zzhan_grader_poa", graderId);
18:
19:         const char* tid = argc > 1 ? argv[1] : "9999";
20:         const char* pwd = argc > 2 ? argv[2] : "test";
21:         const char* str_grade = argc > 3 ? argv[3] : "85.50";
22:         float grade=(float)atof(str_grade);
23:
24:         CORBA::Boolean ret = grader->add_grade(tid,pwd,grade);
25:         if (!ret)
26:             cout<<"Teacher(CPP): Invalid TID or Password."<<endl;
27:         else
28:             cout << "Teacher(CPP): The grade is updated to: "<<grade<<endl;
29:     } catch(const CORBA::Exception& e) {
30:         cerr << e << endl;
31:         return 1;
32:     }
33:     return 0;
34: }
```

```
1: ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
2: // GradeImpl.h, Troy
3:
4: #include "Grade_s.hh"
5: #include <math.h>
6: #include <string.h>
7:
8: USE_STD_NS
9: class GraderImpl : public virtual POA_Grade::Grader{
10: private:
11:     CORBA::Float _grade;
12: public:
13:     GraderImpl():_grade(0.0){}
14:
15:     CORBA::Boolean add_grade(const char* tid, const char *pwd, CORBA::Float grade){
16:     CORBA::Boolean ret=false;
17:     try {
18:         PortableServer::ObjectId_var securityId =
19:         PortableServer::string_to_ObjectId("zzhan_security");
20:
21:         Grade::Security_var security =
22:         Grade::_Security::_bind("/zzhan_security_poa", securityId);
23:
24:         CORBA::Boolean result = security->check_teacher_pwd(tid,pwd);
25:         if (!result)
26:             cout<<"Grader(CPP): Invalid TID or Password."<<endl;
27:         else{
28:             cout << "Grader(CPP): Teacher: "<<tid<<" logged in. The grade is updated to: "<<grade<<endl;
29:             _grade=grade;
30:             ret=true;
31:         }
32:     }
33:     catch(const CORBA::Exception& e) {
34:         cerr << e << endl;
35:         return false;
36:     }
37:     return ret;
38: }
39:
40: CORBA::Float show_grade(const char* sid, const char *pwd) {
41:     CORBA::Float ret=-1.0;
42:     try {
43:         PortableServer::ObjectId_var securityId =
44:         PortableServer::string_to_ObjectId("zzhan_security");
45:         Grade::Security_var security =
46:         Grade::_Security::_bind("/zzhan_security_poa", securityId);
47:         CORBA::Boolean result = security->check_student_pwd(sid,pwd);
48:         if (!result)
49:             cout<<"Grader(CPP): Invalid TID or Password."<<endl;
```

```
50:     else{
51:         cout << "Grader(CPP): Student: "<<sid<<" logged in. The grade: "<<_grade<<" is retrieved."<<endl;
52:         ret= _grade;
53:     }
54: }catch(const CORBA::Exception& e) {
55:     cerr << e << endl;
56:     return -1.0;
57: }
58:     return ret;
59: }
60: };
61:
62: #define TROY_MAX 100
63: class SecurityImpl : public POA_Grade::Security{
64: private:
65:     char* _tid[TROY_MAX];
66:     char* _t_pwd[TROY_MAX];
67:     int _t_length;
68:
69:     char* _sid[TROY_MAX];
70:     char* _s_pwd[TROY_MAX];
71:     int _s_length;
72: public:
73:     SecurityImpl(){
74:         _tid[0]="9999";
75:         _t_pwd[0]="test";
76:         _tid[1]="8888";
77:         _t_pwd[1]="test";
78:         _tid[2]="7777";
79:         _t_pwd[2]="test";
80:         _t_length=2;
81:
82:         _sid[0]="1001";
83:         _s_pwd[0]="test";
84:         _sid[1]="2002";
85:         _s_pwd[1]="test";
86:         _sid[2]="3003";
87:         _s_pwd[2]="test";
88:         _s_length=2;
89:     }
90:
91:     CORBA::Boolean check_teacher_pwd(const char* tid, const char* pwd) {
92:         int i=0,j=0;
93:         int found=0;
94:         while ((i<=_t_length)&&(!found)){
95:             if ((j=strcmp(_tid[i],tid))==0)
96:                 found=1;
97:             else
98:                 i++;
```

```
99:     }
100:     if(i>_t_length){
101:         _tid[i]=(char *)malloc(sizeof(char)*strlen(tid));
102:         strcpy(_tid[i],tid);
103:         _t_pwd[i]=(char *)malloc(sizeof(char)*strlen(pwd));
104:         strcpy(_t_pwd[i],pwd);
105:         _t_length++;
106:         cout<<"Security(CPP): Created TID "<<tid<<" : "<<pwd<<endl;
107:         return 1;
108:     }
109:     if((j=strcmp(_t_pwd[i],pwd))==0){
110:         cout<<"Security(CPP): Authentication succeeded on TID:"<<tid<<endl;
111:         return 1;
112:     }else{
113:         cout<<"Security(CPP): Authentication failed on TID:"<<tid<<endl;
114:     }
115:     return 0;
116: }
117:
118: CORBA::Boolean check_student_pwd(const char* sid, const char* pwd) {
119:     int i=0,j=0;
120:     int found=0;
121:     while ((i<=_s_length)&&(!found)){
122:         if ((j=strcmp(_sid[i],sid))==0)
123:             found=1;
124:         else
125:             i++;
126:     }
127:     if(i>_s_length){
128:         _sid[i]=(char *)malloc(sizeof(char)*strlen(sid));
129:         strcpy(_sid[i],sid);
130:         _s_pwd[i]=(char *)malloc(sizeof(char)*strlen(pwd));
131:         strcpy(_s_pwd[i],pwd);
132:         _s_length++;
133:         cout<<"Security(CPP): Created SID "<<sid<<" : "<<pwd<<endl;
134:         return 1;
135:     }
136:     if((j=strcmp(_s_pwd[i],pwd))==0){
137:         cout<<"Security(CPP): Authentication succeeded on SID:"<<sid<<endl;
138:         return 1;
139:     }else{
140:         cout<<"Security(CPP): Authentication failed on SID:"<<sid<<endl;
141:     }
142:     return 0;
143: }
144:
145: };
```