CS464/564 Project 1
Assigned: Monday Sep. 18, 2000
Due: Wednesday Sep 27, 2000

Overview

In this project you will create two simple servers and two simple clients. You will submit the source code in a manner to be described in forthcoming email, and will also give a short demonstration of your running program to the TA at a time to be set later.

Problem setting:

Design and implement a simple CORBA system, based on Visibroker for Java/C++ 4.0. Suppose we have 4 kinds of objects in our system: Teacher, Student, Grader and Security.

- The Security object receives requests from Grader. It maintains a list of valid Teacher ID (tid) and password (pwd), a list of valid Student ID(sid) and password (pwd). It checks if the pwd is correct.
- The Grader object receive request from Teacher and Student. It will ask Security to check the pwd. The Grader maintains a simple float variable: grade. If the pwd for Teacher is correct, it will update grade. If the pwd for Student is correct, it will retrieve grade to the Student.
- The Teacher object send request to Grader.
- The Student object send request to Grader.

The IDL interface Grade.idl is given like:

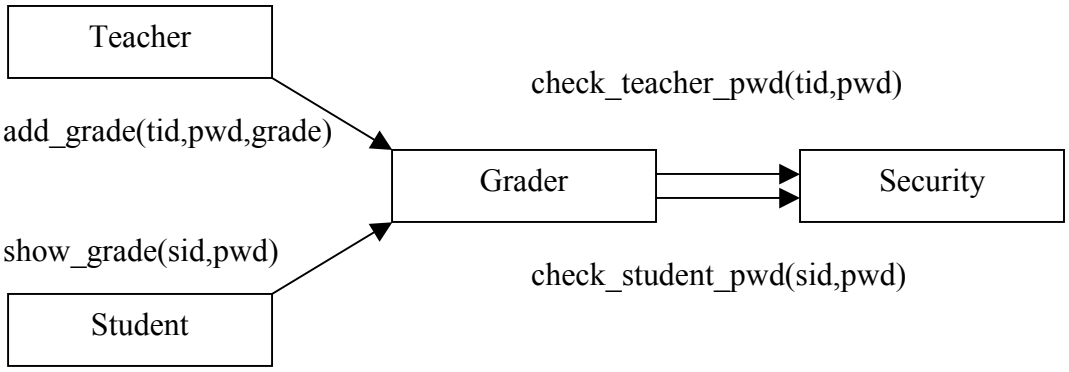
```
// Grade.idl

module Grade {
        interface Grader {
                boolean add_grade(in string tid, in string pwd, in float grade);
                float show_grade(in string sid, in string pwd);
        };
        interface Security {
                boolean check_teacher_pwd(in string tid, in string pwd);
                boolean check_student_pwd(in string sid, in string pwd);
        };
};
```

where: tid denotes teacher ID #, and sid denotes Student ID #.

Note: we will not use user defined exceptions (i.e., ones defined in IDL).

However, you should catch all CORBA system exceptions any time you do a CORBA call – see the online VisiBroker examples and the CORBA-1 lecture for details.

The structure of the system is like:



Background Information:
PLEASE DON'T LOG IN NIF-S2.EECS.WSU.EDU.
Visibroker for Java 4.0 is installed in
nif-s2.eecs.wsu.edu:/local/dist_systems/cs564/vbj4_0.
Visibroker for C++ 4.0 is installed in
nif-s2.eecs.wsu.edu:/local/dist_systems/cs564/vbcpp4_0.

For Java:

1) If you want to use Visibroker for Java 4.0, follow the steps:

1. login any machine in ETRL301, using your own EECS account and pwd.
2. type "env | grep SHELL" to check what kind of shell you are using.
3. if you are using C-SHELL ("SHELL=/bin/csh" will be displayed), type "source /local/dist_systems/cs564/startjava.csh" to setup necessary environments.
4. if you are using BASH-SHELL ("SHELL=/bin/bash") or K_SHELL ("SHELL=/bin/ksh"), type "source /local/dist_systems/cs564/startjava.sh".
5. Now, you can program!….
6. Most likely, this project involves 7 files:
   Grade.idl, Student.java, Teacher.java, Security.java, SecurityImpl.java, Grader.java, GraderImpl.java
7. Build your Java SRC. (Details see **How to build my Java SRC?**)
8. Type "ps –aux | grep osagent" to see if there is any OSAGENT already running.
9. If no, type "osagent &" to fire an OSAGENT.
10. Using "vbj XXXX" to interpret your class file.

2) How to build my Java SRC?

1. Create a Makefile.java file under your SRC directory, say ~user/cs564/project1/java

Makefile.java:

```
.SUFFIXES: .java .class .idl .module
.java.class:
        vbjc $<
.idl.module:
        idl2java $<
        touch $@
default: all
clean:
        rm -rf Grade
        rm -f *.class *.tmp *.module *~
IDLS = \
        Grade.idl
MODULES = $(IDLS:.idl=.module)
SRCS = \
        Student.java \
        Teacher.java \
        Grader.java \
        Security.java
CLASSES = $(SRCS:.java=.class)
all:    $(MODULES) $(CLASSES)
```

2. Type "make –f Makefile.cpp" to build all executable file.
3. Your class files are "Security.class" "Grader.class" "Student.class" and "Teacher.class"

For C++:

1) If you want to use Visibroker for C++ 4.0, follow the steps:
login any machine in ETRL301, using your own EECS account and pwd.
type "env |grep SHELL" to check what kind of shell you are using.
if you are using C-SHELL ("SHELL=/bin/csh" will be displayed), type "source /local/dist_systems/cs564/startcpp.csh" to setup necessary environments.
if you are using BASH-SHELL ("SHELL=/bin/bash") or K_SHELL ("SHELL=/bin/ksh"), type "source /local/dist_systems/cs564/startcpp.sh".
Now, you can program!….
Most likely, this project involves 6 files:
Grade.idl, Student.C, Teacher.C, GradeImpl.h Grade.C Security.C
Build your C++ SRC (Details see **How to build my C++ SRC?**).
Type "ps –aux | grep osagent" to see if there is any OSAGENT already running.
If no, type "osagent &" to fire an OSAGENT.
Using "server" to fire a server.
Using "client" to fire a client.

2) How to build my C++ SRC?

1. Create a Makefile.cpp file under your SRC directory, say ~user/cs564/project1/cpp

Makefile.cpp:

```
include /local/dist_systems/cs564/vbcpp4_0/example/stdmk
EXE = Teacher  Student  Grader  Security
all: $(EXE)
clean:
-rm -f core *.o *.hh *.cc $(EXE)
-rm -rf SunWS_cache

#
# "Grade" specific make rules
#
Grade_c.cc: Grade.idl
        $(ORBCC) Grade.idl
Grade_s.cc: Grade.idl
        $(ORBCC) Grade.idl
Teacher: Grade_c.o Teacher.o
        $(CC) -o Teacher Teacher.o Grade_c.o \
                $(LIBPATH) $(LIBORB) $(STDCC_LIBS)
Student: Grade_c.o Student.o
        $(CC) -o Student Student.o Grade_c.o \
                $(LIBPATH) $(LIBORB) $(STDCC_LIBS)
Grader: Grade_s.o Grade_c.o Grader.o
        $(CC) -o Grader Grader.o Grade_s.o Grade_c.o \
                $(LIBPATH) $(LIBORB) $(STDCC_LIBS)
Security: Grade_s.o Grade_c.o Security.o
        $(CC) -o Security Security.o Grade_s.o Grade_c.o \
                $(LIBPATH) $(LIBORB) $(STDCC_LIBS)
```

2. Type "make –f Makefile.cpp" to build all executable file.

3. Your executable files are "Security" "Grader" "Student" and "Teacher"