CS464/564 Project2
Given: Monday October 2, 2000
Due: Monday October 9, 2000, at the beginning of class (Not October 11!)

Overview:
In this project you will gain some simple experience with CORBA user exceptions and with programming the DII. You may start from the solution code provided by the TA if you prefer, or you may start from your own Project1 code.

Instructions on how to turn it in, and where you can copy theTA's Project1 from, will be forthcoming via email from the TA.

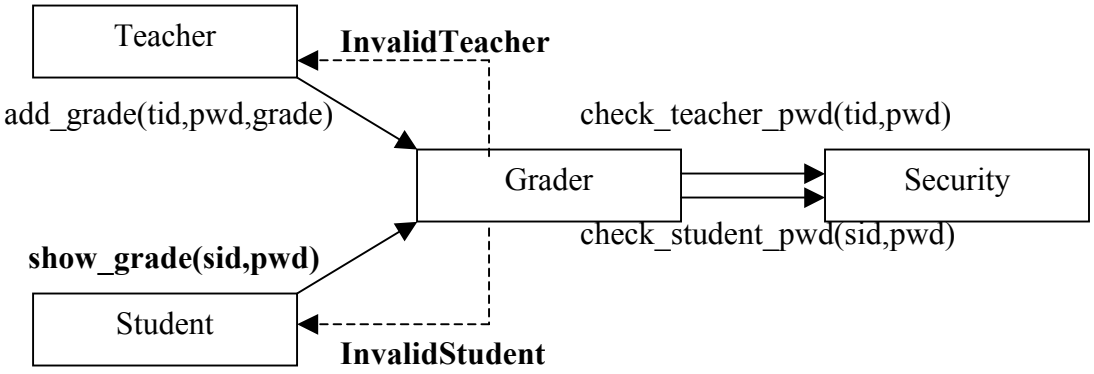
Problem Setting:
Project 2 is based on Project1. In this project, you need to:

- Modify the *Student* to use the Dynamic Invocation Interface (DII)
- Implement two user defined exceptions, *InvalidTeacher* and *InvalidStudent*, which are given in the *Grade.idl* file below. *Grader* raises *InvalidTeacher* when *Security* returns false on *check_teacher_pwd()* and *InvalidStudent* when *Security* returns false on *check_student_pwd()*
- **(For graduate student only)** Graduate Students will have to do this project in two languages, but in the other language from last time for *Student*, *Teacher*, and *Grader*. If somehow this doesn't include both languages (say the *Security* is the only one in Java), they should do one in the other language so that both language are covered.

Grade.idl for Project2:

```
// Grade.idl
module Grade {
    interface Grader {
        exception InvalidTeacher{
                string reason;
        };
        exception InvalidStudent{
                string reason;
        };
        void add_grade(in string tid, in string pwd, in float grade)
                raises (InvalidTeacher);
        float show_grade(in string sid, in string pwd)
                raises (InvalidStudent);
    };
    interface Security {
        boolean check_teacher_pwd(in string tid, in string pwd);
        boolean check_student_pwd(in string sid, in string pwd);
    };
};
```

System Structure:



Notes:

- You can directly use the code in project1 for *Security* because this project doesn't involve any works on that part.
- You may copy anyone's Makefile for any purpose in this class, or any shell scripts it may use (probably must don't). Um, unless the shell script generated your CORBA program….
- **When using the DII, please note that you can not catch any specific user-defined exception. You can only catch UnknownUserException, which is not a problem. For more information, see Chapter 22 of VBJ PROG.**
- If you do the whole project in Java and there are 6 files in total, named respectively as *Student.java*, *Teacher.java*, *Grader.java*, *GraderImpl.java*, *Security.java* and *SecurityImpl.java*, you can directly use the Makefile.java for project1 to build your project2. If you like to organize the files in your own way (say change the file names), it's fine but you might not be able to use the Makefile.java.
- If you do the whole project in C and there are 6 files in total, named respectively as *Student.C*, *Teacher.C*, *Grader.C*, *GradeImpl.h*, *Security.C*. You can directly use the Makefile.cpp for project1 to build your project2. If you like to organize the files in your own way (say change the file names), it's fine but you might not be able to use the Makefile.cpp.
- Please use "osfind –a" to check if there is any *osagent* already running in our lab. If yes, please DONOT fire another *osagent* on your local machine again. **Please note that "ps –aux|grep osagent" won't tell whether there is already an *osagent* running on other machines.** You will find that in most of the cases, there will be at least one *osagent* running on nif-s2, fired by its *root*. It should be enough for you to finish your project2.
- Please **KILL** all your processes before your logout, especially when you use "&" to fire your Server.
- Please **DONOT** lock any of the machines in SNIF lab. Otherwise you should be responsible for any possible loss of your files.

Suggestions:

- Study the DII programming from example: bank_dynamic, under /local/dist_systems/cs564/vbj4_0/examples/basic/bank_dynamic/ and /local/dist_systems/cs564/vbcpp4_0/examples/basic/bank_dynamic/. To start your programming, you can directly copy and modify from its *Client* program. **Please note that the *Server* part of this examples is using DSI, which is not required in Project2.**
- You will also find the following materials are helpful:
    - **Visibroker for Java 4.0: Programmer's Guide** and **Visibroker for C++ 4.0: Programmer's Guide:**
        Ch5 *Handling Exceptions*
        Ch22 *Using the Dynamic Invocation Interface***.**
    - **Visibroker for Java 4.0: Reference Manual** and **Visibroker for C++ 4.0: Reference Manual**

  These materials are available at:
    - ETRL301 (Books)
    - Locally Online (PDF): /local/dist_systems/cs564/documents/vbcpp4_0/ and /local/dist_systems/cs564/documents/vbj4_0/
      Also, downloadable at our course webpage:
      http://www.eecs.wsu.edu/~bakken/464-564-Web.htm
    - Remotely Online (HTML and PDF):
      http://www.inprise.com/techpubs/visibroker/visibroker4/vbcpp40-index.html and
      http://www.inprise.com/techpubs/visibroker/visibroker4/vbj40-index.html