

Assignment #1: Sockets and Remote Class Loading

DUE: 10:30 am, Wednesday, April 14.

Part 0: Configuring Your Account

For this assignment, we will be using Java SDK 1.4. In the Solaris environment, you can check which version you are using with the `-version` option:

```
gamma% java -version
java version "1.4.0_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.0_01-b03)
Java HotSpot(TM) Client VM (build 1.4.0_01-b03, mixed mode)
```

You can configure your account to use Java SDK 1.4 by subscribing to “JDK current” (ie, type “subscribe” at the prompt and follow the menu to select JDK current). After subscribing, you need to logout and log back in before the changes take effect.

Part I: Remote Class Loading

Below is an interface for components that can be used to calculate a maximum number. A client uses this interface by calling the `add()` method for each number to be considered. Then, the `calcMax()` is called to calculate (and return) the maximum of the submitted numbers.

```
public interface FindMaxI {
    public void add (int n);
    public int calcMax ();
}
```

Write a program that tests the functionality of a class that implements the `FindMaxI` interface. Then use the `URLClassLoader` to dynamically load the class definitions of these implementation classes to be tested. You can find 3 classes that you should test at a URL location that will be posted to the newsgroup.

The program that you submit should be called `StdURLLoading` and should take 1 argument: the name of the class to be tested. You may hard-code the URL from which the class should be loaded into the implementation of `StdURLLoading`.

For example, your code would be executed with:

```
java StdURLLoading FM1
```

To complete this part of the lab:

1. Test each of the class implementations from the provided URL and determine which, if any, are correct. For implementations that are not correct, give specific examples of how/where they fail. Turn in a hard copy of your analysis in class.
2. Submit your code using the CSE submit command.

Part II: Sockets and Remote Class Loading

For this part of the assignment, you will write a simple client-server application in which the client application can be dynamically loaded at remote sites and which uses sockets to communicate between the client and the server.

The user interface for interacting with your server should implement the following interface:

```
public interface GradeGetterUI {
    public boolean login();
    public void logout();
    public String getNames();
    public String getGrade();
}
```

The user (ie your TA) will write a program that instantiates an object that implements GradeGetterUI (by dynamically downloading the class from a remote site). This object will hide all the socket communication with the server from the user program.

The server and “UI” classes that you write should agree on the port on which the server will listen for connection requests from the UI.

The behavior expected by your user is as follows. An invocation of getNames() should return a string containing the names of the individuals in your group. The first invocation of getGrade() should return an F, the second a D, then a D+, then a C- and so on. In other words, each invocation should strictly improve your grade. Once the grade is an A, no further improvement is possible. If the user program written by the TA creates a GradeGetterUI, and calls getGrade once, it should obtain an “F” from the getGrade invocation. If the same program is run again (ie runs from scratch, including loading the GradeGetterUI implementation class) it should obtain a “D”. In other words, the state that indicates the grade to be returned must be stored at the server since the GradeGetterUI is not persistent in the same way as the server.

Finally, invocations of getGrade() and getNames() are possible only after a successful (return of true) invocation of login() and before the corresponding invocation of logout().

Completing submission of this part. We will test your completion of this part of the assignment by remotely loading your class that implements the by binding to your server and invoking

the `getGrade()` method. In order for us to know that your server is ready to be tested, you *must* send email to the TA indicating the name of your class that implements `GradeGetterUI` and URL at which this class file is located.

There are 3 windows of opportunity for you to be graded: 8:30 - 9:30 the morning the lab is due, 8:30 - 9:30 am the next day (with a 25% penalty), and 8:30 - 9:30 am the next day (with a 50% penalty). You *MUST* send your email before 8:30 am the morning you wish to be graded. Have your server up and running during the grading window (8:30 - 9:30). Leave it up until you hear back from the TA.

Part III: Web Sockets

Write a (simple) web browser. Your program should take a single command-line argument: the URL of the page to be retrieved:

```
java LittleBrowser http://www.cis.ohio-state.edu/~paolo/index.html
```

The page retrieved should just simply be written to `System.out`. There is no need to process this page or support, for example, following hyperlinks to other pages. You may also assume that the retrieved page will contain only ascii text (no binary image data for example). You may use any of the standard classes in `java.io` and `java.net`.

Submit your code using the CSE submit command.