CS360 Assignment 3
Password Tester

Date Assigned: Tuesday, February 24, 2004
Date Due: Tuesday, March 9, 2004
Total Points: 50

Introduction

The objective of this assignment is to write the classes necessary to implement a password checker. You need to implement these classes in such a way that you can then use the password tester with any of your future Java applications or applets.

Goals

The goal of this assignment is to write a class, extend this class, write an interface and extend one of the Java API classes. Specifically, you will need to understand the concepts of:

- Inheritance
- Polymorphism
- Event Handling
- Interfaces

Assignment

You should write this assignment one step at a time, testing your classes at every step. None of the classes are long, but they do use the advanced concepts of inheritance and polymorphism.

It is important when writing all the classes in this assignment to adhere to the names that I give you. This is necessary in order for your classes to be able to work with the test applets and applications that I will give you.

Class `PasswordList`

This class contains an array of strings, which are the valid passwords. The length of this array is determined from the array of strings that is passed to the constructor. This class should contain:

- Constructor: `PasswordList( String[] )` that sets the instance variable to the passed variable.
- Method: `boolean match( String )` that compares every valid password in the instance variable array, to the string passed to the method. If a match is found then the method should return true, otherwise it should return false.

Once you have completed this class, test it by creating an object in another class. This is not a necessary step, but it will make your life easier later on.

Class `PasswordTries`

This class extends (or inherits from) the class PasswordList. This method will contain an instance variable to store the maximum number of tries allowed. It will also contain:

- Constructor: `PasswordTries( String[], int )`, where the array of strings is the valid passwords and the integer is the maximum number of tries.
- Method: `boolean match( String )` that overrides the `match` method in the superclass `PasswordList`. This method should return false if the maximum tries is reached, otherwise it will call the method in the superclass.

You may need an additional variable in this class to store the current try number. Remember to use constants if necessary. Again, you should test this class before moving on.

Interface `PasswordListener`

This interface should contain two methods:

- `void passwordCorrect()`
- `void passwordIncorrect()`

These methods will do specific things depending on the class that implements the interface. For testing purposes, assume that they will display a message depending on whether the password is correct or not.

Class `PasswordField`

This class is your specific version of a text field that performs password validation whenever the user hits enter. The class should extend `TextField` to make use of existing text field methods and implement `ActionListener` to catch the event of the user hitting enter.

The class should contain the following instance variables:

- An object of `PasswordList`. Remember, following the rules of inheritance, this object could also hold a `PasswordTries` object.
- An array of objects of `PasswordListener`. Assume a maximum of 20.

It should contain the following methods and constructors:

- Constructor: `PasswordField( int, PasswordList )` where the integer is the width of the text field and `PasswordList` is the list of valid passwords. This constructor will need to call the superclass constructor, set the `PasswordList`, add an action listener to catch when the user hits enter (`addActionListener(this)`), and make sure that the characters displayed in the text field are * (hint, use the `setEchoChar` method in the `TextArea` class).
- Method: `addPasswordListener( PasswordListener )` adds the passed `PasswordListener` to the array of `PasswordListeners`.

- Method: `actionPerformed( ActionEvent )` compares the text in this textfield with the valid passwords. If a match is found then the `ActionListeners` should call the method `passwordCorrect`, otherwise they should call the method `passwordIncorrect`.

Testing Your Classes

I will place a couple of test files in the CS360 public folder. These test files will only work if you name your classes and methods as I have specified.

Submitting Your Programs

Save both of your files in a folder called (your PUNET ID) and place the folder in the CS360 drop folder by 9.25 AM on the assignment due date. You do not need to submit the .class files. Just submit your .java files.