Inputs and outputs to and from most applications are likely to be in text. The Java **String** class (4.4, 10.10-11) provides structures and procedures to manipulate characters. This assignment includes exercises that get input from keyboards, pointers, and files. This input is often in alphanumeric format, even if integers or other numbers are expected. Java classes like Scanner, System.in, and System.out parse input text in strings and compose output strings, although some types are automatically converted to Strings for output.

The material on Strings is scattered throughout the text.  The first mention is in Chapter 1 and then is further developed until Section 4.4.  There the String class is introduced with its methods most often used but exercises in which strings play a major part do not occur until Chapter 6. This assignment calls for exercises from Chapter 6 and 12 but Chapter 10 also has material for Strings. You may need to go back to Chapter 4 for review.

Often the first input to a program is a user ID and a password.  A common security and validity check is that a password must satisfy certain conditions which makes a repeat loop necessary to allow for mistakes.  An example programs without the validity checks is shown in the accompanying video using **JOptionPane** classes.

Some defensive programming assumes that all users are stupid, mischievous, and/or downright evil.  Input validation is always necessary and allowing choices from menus or other displays will likely reduce errors.  It is often best if users do not have to remember and/or type text in response to input prompts.  Contrary to what it says in Exercise 12.28 do not ask the user to enter the file path and name, this assignment requires the use of **JFileChooser** classes (see instructions below) to specify the input file.

Java also has a **Password** Class which can be used in the password program.  However, it uses various methods of the JOptionPane class which are not considered in the 10th edition. It captures the password as a char[] array so each character can be examined separately. The example program in the video does not do this, it uses String class methods.  One useful link to the Password class is [http://docs.oracle.com/javase/tutorial/uiswing/components/passwordfield.html. (Links to an external site.)Links to an external site. (Links to an external site.)Links to an external site.](http://docs.oracle.com/javase/tutorial/uiswing/components/passwordfield.html)

To read a text file you'll need to get into Sections 12.10-11.

|  |  |  |
| --- | --- | --- |
| Assignment 5 | 11th Edition | 10th Edition |
| CS | 6.18, 6.23, 12.13; see Section 10.10.4 for regular expression | 6.18, 6.23, 12.13; see Section 10.10.4 for regular expression |
| MIS | 6.18, 6.23, 12.13; see Section 10.10.4 for regular expression | 6.18, 6.23, 12.13; see Section 10.10.4 for regular expression |

For this assignment, exercises 6.18, 12.13, and 14.13 should use exception handling for valid or invalid passwords and if the user selects a file path and name for which the file does not exist. As shown in the video remember to tell the user the password rules before the input and/or in the error messages if the password is not valid. For exercises 12.13 the program should allow the user to specify the .txt file to be examined.  A required way to do that is to invoke [JFileChooser classes (Links to an external site.)Links to an external site.](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFileChooser.html" \t "_blank) which will throw an exception if the file cannot be found. Your program should use try-catch blocks to handle it.

Exercise 6.23 should have **two repeat loops**, the second nested in the first. The outer loop contains code that asks the user for the input string or to exit while the inner loop asks for a particular character to be found. The first prompt should ask for the String from which multiple characters can be searched one at a time in the second loop. When the user is done with searching for characters the program should prompt for a new String or ask if the user is done.  Remember to be very specific about what constitutes a valid String.  It is you who chooses to allow only letters and/or numbers, or also allow printable punctuation marks.  The String validation methods must check that.

Your submissions will be checked by entering a variety of valid and invalid input, so starting with this assignment it is required that you structure the program so that it returns to the input section regardless of prior input. It should have an explicit input for the user to leave the program without which the program should continue to loop.

Remember:

1. All programs need a brief introduction printed for the user saying what the program will do.
2. The prompts to the user for the inputs required for the program need to say what each input is (*e.g.*,letter, word, number phrase), its units, and how it should be formatted (*e.g.,* 3 numbers separated by spaces).
3. Write the program with a repeat loop so that when the first set of input is done with the programs ask the user if it should run again for more, presumably different input. There should be an option to exit or to repeat the program with new input.