**Purpose**

* To validate user input
* To use Toast notification messages
* To implement an event handler for a Button

**Functional Requirements for Your Login Example App**

Now it's time to add functionality and data validation to your *Login Example* app.

As savvy Android developers, we should always consider how to best validate all data entered by the user.

Steps to follow for input validation:

* select (or make) the proper UI <View>
* set any properties of the <View> that will help with validation
* implement, in Java code, the remaining validation rules not covered by the previous 2 points

Extend the behaviour of your *Login Example* app, and implement data validation.

**Log In Button**

When the user taps the **Log In** button:

* *validate* the username input; refer to the section *Input Validation* (below) for addition information
* *validate* the password input; see *Input Validation* (below)
* *success* - if and only if ( iff ) all input is valid, then display a Toast message (with a long duration); display the username and password

**Input Validation**

Validate the user input for username and password.

**username Validation**

The username is considered valid if each of the following conditions is true:

* the username is mandatory
* the length of the username must be *exactly* 8 characters in length
* each of the first 2 positions can only be lower-case alphabetic characters (a-z)
* positions 3 and 4 can either be a lower-case alphabetical character (a-z) OR a digit (0-9)
* positions 5 thru 8 (inclusive) must be digits (0-9)

**Note**: these validation rules are based on the ITS policy regarding student usernames

If any one of the above conditions is false, the input for the username is invalid. Implement the following:

* for the username <TextView>  :: use setError( ) to set an appropriate error message
* for the username <TextView> :: request focus (requestFocus())

Examples with explanation:

* aaaa0000 (valid)
* zzzz9999 (valid)
* AAAA0000 (invalid; upper-case)
* ?0000000 (invalid; non-alphabetical character)
* aa000000 (valid)
* abc12345 (valid)
* aaaa12345 (invalid; too long)
* abcdefgh (invalid; positions 5 thru 8 are not digits)
* a (invalid; too short)

**password Validation**

The password is considered valid if each of the following conditions is true:

* the password is mandatory
* the length of the password is greater than or equal to 5 ( >= 5 )

If any one of the above conditions is false, the input for the username is invalid. Implement the following:

* for the password <TextView>  :: use setError( ) to set an appropriate error message
* for the password <TextView> :: request focus (requestFocus())

**How to Implement**

Good news! Java has 2 built-in classes to help you implement the above rules for validating the username and password. The classes are:

* String ( <https://docs.oracle.com/javase/8/docs/api/index.html?java/lang/String.html> )
* Character ( <https://docs.oracle.com/javase/8/docs/api/index.html?java/lang/Character.html> )

**Class String**

Java's String class represents character strings. For this activity, you will need to *get* the String values for the username and password.

To get the number of characters in a String, use the length( ) instance method.

For example,

String course = new String( "CST8227" );

System.out.println( "The length() is: " + course.length() );

Displays this output: The length() is: 7

The method charAt( int index ) is a convenient method to get the single character at position index. Remember, Java begins indexing at 0 (zero), just like Javascript.

The charAt( ) instance method returns a char, which is one of Java's primitive data types.

For example,

String course = "CST8227";

System.out.println( "charAt(0): " + course.charAt(0) );

Displays the output: charAt(0): M

To get the sub-string of a string, use String's subString( int beginIndex, int endIndex ) instance method.

For example,

String course = "CST8227";  
String courseCode = course.substring(0, 3);  
String courseNumber = course.substring(3, course.length());  
System.out.print( "Course Code: " + courseCode);  
System.out.println( "\tCourse Number: " + courseNumber);

Displays this output:

Course Code: CST <tab> Course Number: 8227

**Class Character**

Java's Character class wraps a value of the primitive type char in an object.

The class method (i.e. static) isLowerCase( char ch ) returns boolean true if the character ch is a lowercase letter (a-z); otherwise, false is returned.

For example,

String course = "CST8227";

System.out.println( "is the 1st character position a lowercase letter? " + Character.isLowerCase(course.charAt(0)) );

System.out.println( "is the 4th character position a lowercase letter? " + Character.isLowerCase(course.charAt(3)) );

Displays this output:

is the 1st character position a lowercase letter? false

is the 4th character position a lowercase letter? false

Of interest... Character.isUpperCase(course.charAt(0)) returns true.

The class method (i.e. static) isDigit( char ch ) returns boolean true if the character ch is a digit (0-9); otherwise, false is returned.

For example,

String course = "CST8227";

System.out.println( "is the 1st character position a digit? " + Character.isDigit(course.charAt(0)) );

System.out.println( "is the 4th character position a digit? " + Character.isDigit(course.charAt(3)) );

Displays this output:

is the 1st character position a digit? false

is the 4th character position a digit? true

**Deliverables**

1. MainActivity.java
2. APK file for your *Login Example* app

**To Find Your APK File**

Here's how I find my APK file...

The name of my Android Studio project is: **LoginExample**

My APK file is found at:

<PATH TO YOUR ANDROID STUDIO PROJECTS>/LoginExample/app/build/outputs/apk/app-debug.apk