SYST 17796

ICE 4

This exercise is to be completed individually during class time. Students are allowed to discuss the exercise and provide assistance to one another but each student is responsible for their own final product. Students not present in class are not eligible for credit unless they are eligible for a specific accommodation which is worked out between the instructor and that student according to class/departmental/college policies.

# Overview

This Exercise will allow students to demonstrate their testing skills by completing the remaining functionality for the password validator code started in class and creating Junit tests to validate the remaining requirements. The submission will consist of the completed PasswordValidator.java source code and the completed PasswordValidatorTest.java source code **with proper design, commenting and coding convention followed**. Students should submit a .zip file **labelled with their name and student number** to the ICE 4 DropBox containing:

* PasswordValidator.java
* PasswordValidtaorTest.java

# Credit/Necessary Information

Complete the code necessary to satisfy the remaining two bolded requirements. This should be accomplished by adding two new static utility methods to your PasswordValidator.java code and calling them from your main method. The requirements for the password validator studied in class are:

* A password is valid if it is at least 8 characters long (completed in class) AND
* **A password is valid if it contains at least one upper case letter AND**
* **A password is valid it contains at least one special character (a special character is defined as being not a letter or a digit so** **HINT: use methods from the Character class in the Java API which can tell you quickly is a given character is a letter or a digit and whether it is uppercase**).

| **Task** |
| --- |
| 1. If you need a base for your code, you can take it from SLATE.   **\*\*you will not need to push anything to the repository for this ICE\*\*** |
| 1. Complete the code in PasswordValidator.java that satisfies the above bolded requirements. |
| 1. Add 6 new JUnit tests which will validate the code you have added (remember good, bad boundary for each test. It is your job to decide what the boundary is). |
| 1. Ensure your code has a good design and follows good coding conventions (with comments). |
| 1. **Zip the two source files** into an archive and label it with your name and student number and submit to the ICE 4 DropBox. |
| 1. **Submissions not in zip format will not be evaluated. Students who do not properly cite their source code by adding themselves as a modifier where appropriate will be assigned a grade of zero.** |

Rubric

|  |  |
| --- | --- |
| **Criteria** | **points** |
| Code in password validator meets the requirements given | 3 |
| 6 Junit test are correctly written and pass when run. | 3 |
| 6 Junit tests cover the good, bad and boundary cases and test what they are supposed to test! | 3 |
| Code is well designed and follows coding standards such as good commenting, naming conventions and spacing. | 1 |