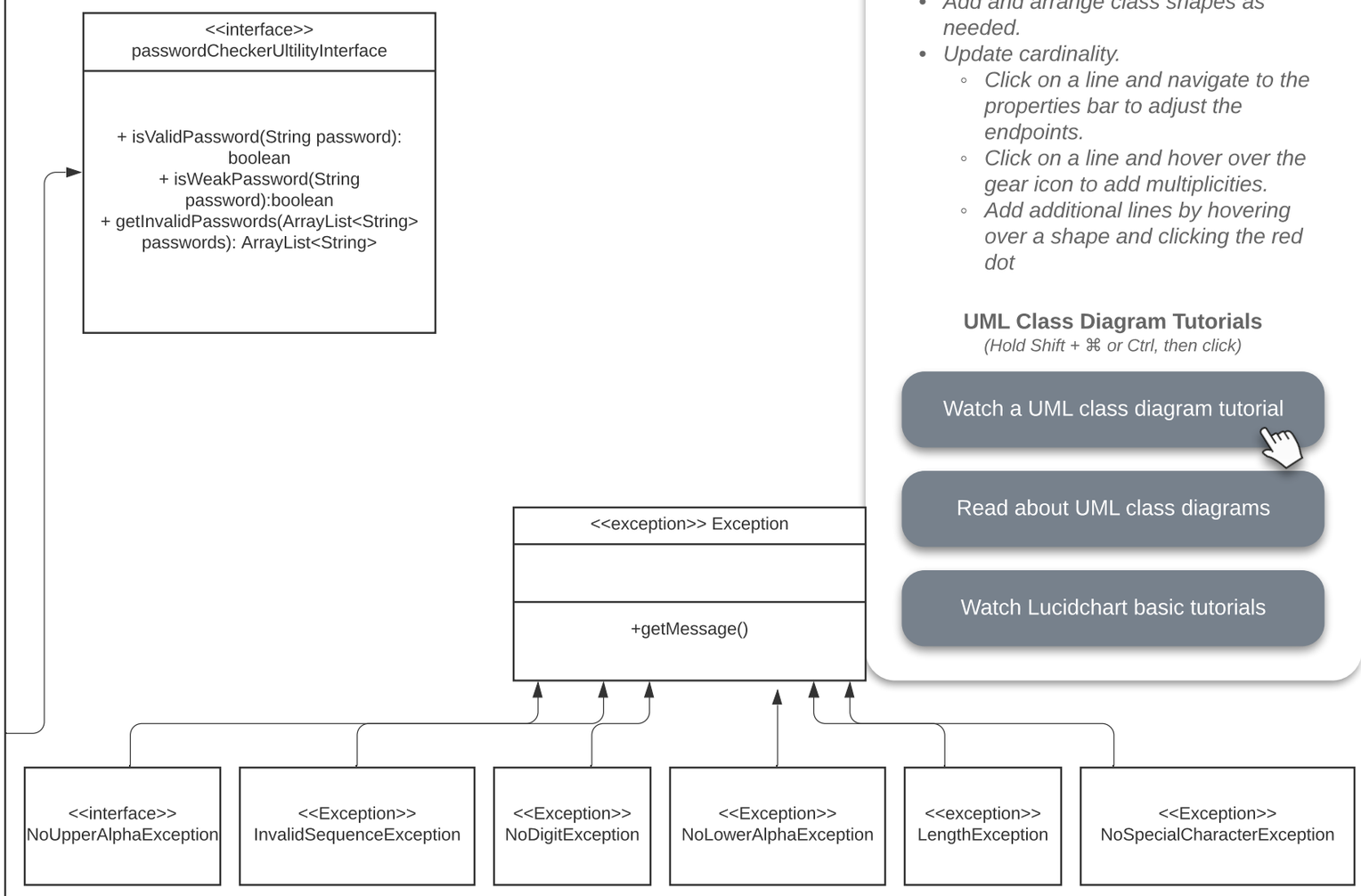
Assignment 1 Design Document

John Mobarry | February 5, 2021

Class PasswordCheckerUtility

-password: String
-illegalPasswords: ArrayList<String>

```
+isValidPassword(String password): boolean
this.password = password;
char first = password.charAt(0);

//checks if password is less than length 6
if (password.length() < 6){throw new LengthException();}

//checks if password has a digit
if (first != digit){
boolean hasDigit = false;
for(int i = 0; i < password.length(); i++){

if(Character.isDigit(password.charAt(i)){
hasDigit = true;
}
}
if(!hasDigit){
throw new NoDigitException();}
}
return true;

}

//checks if password has an uppercase
if(first!= uppercase){
boolean hasUppercase = !password.equals(password.toLowerCase());
if(!hasUppercase){
throw new NoUpperAlphaException();
}
return true;

//checks if a password has a lowercase
if(first!= lowercase){
boolean hasLowerCase = !password.equals(password.toUpperCase());
if(!hasUppercase){
throw new NoUpperAlphaException();
}
return true;
}

//checks if password has to of the same characters in a row
if(first == digit || first == character || first == special character){
for (int i = 0; i < password.length() - 2; i++){

if( password.charAt(i) == password.charAt(i+1) && password.charAt(i) == password.charAt(i+1)){
throw new InvalidSequenceException();}
}
return true;
}

//checks if the password has a special character
if (first == digit || first ==character){
boolean hasSpecial = false;
for(int i = 0; i < password.length(); i++){

if(!Character.isDigit(password.charAt(i) &&!Character.isDigit(password.charAt(i ){
hasSpecial = true;
}
}
if(!hasSpecial){
throw new NoDigitException();}
}
return true;
}


+isWeakPassword(String password): boolean
if (isValidPassword(password) && password.length() >= 6 && password.length() <= 9){
return true;}
else{return false;}

+getInvalidPasswords(ArrayList<String> passwords): ArrayList<String>
illegalPasswords = new ArrayList<String>();
String errorMessage = null;
for( int i = 0; i < passwords.size(); i++){
try{
isValidPassword(passwords.get(i));
}
catch(LengthException e){
errorMessage = passwords.get(i) +  " The password must be 6 characters long";
illegalPasswords.add(errorMessage);
}
catch(NoDigitException e){
errorMessage = passwords.get(i) +  " The password must have a digit";
illegalPasswords.add(errorMessage);
}
catch(InvalidSequenceException e){
errorMessage = passwords.get(i) +  " The password must not have three of the same characters in a row";
illegalPasswords.add(errorMessage);
}
catch(NoSpecialCharacterException e){
errorMessage = passwords.get(i) +  " The password must have a special character";
illegalPasswords.add(errorMessage);
}
catch(NoLowerAlphaException e){
errorMessage = passwords.get(i) +  " The password must have a lower case character";
illegalPasswords.add(errorMessage);
}
catch(NoUpperAlphaException e){
errorMessage = passwords.get(i) +  " The password must have an upper case character";
illegalPasswords.add(errorMessage);
}
}
return illegalPasswords;
}
}
```



Learn about this template

UML class diagrams map out the structure of a particular system by modeling its classes, attributes, operations, and relationships between objects.

To customize this template:

- *Click on any shape and type the information you would like to include.*
- *Add and arrange class shapes as needed.*
- *Update cardinality.*
  - *Click on a line and navigate to the properties bar to adjust the endpoints.*
  - *Click on a line and hover over the gear icon to add multiplicities.*
  - *Add additional lines by hovering over a shape and clicking the red dot*

UML Class Diagram Tutorials

(Hold Shift + ⌘ or Ctrl, then click)

Watch a UML class diagram tutorial

Read about UML class diagrams

Watch Lucidchart basic tutorials

Test Case #	Input	Actual Input	Expected Output	Actual Output	Did the test pass?
1	isWeakPassword(PassWord12)		false		
2	isValidPassword(Pass!Word12)		true		
3	isValidPassword(Passs!Word12)		false		