Password Analyzer

You do not need to worry about Escape sequences (for example, \n, \\, \". These will not be in the password.)

You're working for a growing e-commerce web site, which has become a popular target for thieves. The thieves gain access to customers' accounts by guessing passwords, which are all too often trivial (such as "secret", "password", and "1234"). If your customers used better passwords, your company would have less trouble with fraudulent purchases. You've been tasked with creating a password analyzer, which will inform a customer about the relative strength of their choice of password. A "strong" password is one that is hard to guess – either by sheer length, or by using a combination of letters, numbers and symbols.

In this problem you are to complete several methods in the `PasswordAnalyzer` class. The methods are `isValid`, `getLengthBonus`, `getUpperCaseBonus`, `getDigitBonus`, `getSymbolBonus`, `getCombinationBonus`, `getPoints` and the `getRating` method.

Please see the end of this problem for several java library methods you may find useful in creating your solution to this problem.

The `isValid` method has a single `String pw` parameter and returns a `boolean`. `isValid` returns true if `pw` is a valid password and returns false if `pw` is not a valid password.
A password is valid if it has all of the following characteristics:

- Contains 8 or more characters.
- Contains NO spaces
- Contains at least one of the following
  - Both UPPER case and lower case letters
  - Or both a letter and a number
  - Or both a letter and a symbol (A symbol is any character that is not a letter and not a digit/number)

The following code shows the results of the `isValid` method.

The following code	Returns
`PasswordAnalyzer pwa = new PasswordAnalyzer();`	
`pwa.isValid("pass word");`	`false`
`pwa.isValid("computer123456789ABC!!!!!");`	`true`

The `getLengthBonus` method has a single `String pw` parameter and returns an `int`. `getLengthBonus` returns the number of bonus points `pw` is entitled due to its length. If a `pw` is not valid password, zero length bonus points are awarded. The possible bonus points are awarded as follows:

- Length bonus: (total max = 25 points)
  *when calculating the length bonus, consecutive letters count as one letter; for example
  letter has length 5, *bookkeeper* has length 7 and *brrrr111* has length 3. (mixed case (Rr) is not tested).
  - +2 for each additional character(s) that makes the length greater than 8 up to and including the length of 15. Max of 14 (7 * 2) bonus points.
  - +1 for one additional character(s) beyond 15 up to a max of 11 additional bonus points.

The following code shows the results of the `getLengthBonus` method.

The following code	Returns
`PasswordAnalyzer pwa = new PasswordAnalyzer();`	
`pwa.getLengthBonus("brrrItIscold!!!!!");`	`6 = 3*2`
`pwa.getLengthBonus("computer123456789ABC!!!!!");`	`20 = 7*2+ 6`
`pwa.getLengthBonus("<#CS$$2004?>");`	`4 = 2*2`
`pwa.getLengthBonus("i.luv.programming");`	`15 = 7 * 2 + 1`
`pwa.getLengthBonus("<2015@Wittry@Contest>");`	`19 = 7 * 2 + 5`

The `getUpperCaseBonus` method has a single `String pw` parameter and returns an `int`. `getUpperCaseBonus` returns the number of bonus points `pw` is entitled from the Upper Case letters `pw` contains. If a `pw` is not valid password, zero Upper Case letters bonus points are awarded. The possible bonus points are awarded as follows:

- Upper case bonus: (total max = 10 points)
  (Repeated Upper case letters are treated as separate letters (example "PASS" is 4 upper case letters)
  - +1 for each upper case letter (max 5 points)
  - +1 for each non upper case letter (lower case, numbers or symbols, max 5 points)

The following code shows the results of the `getUpperCaseBonus` method.

The following code	Returns
`PasswordAnalyzer pwa = new PasswordAnalyzer();`	
`pwa.getUpperCaseBonus("<#CS$$2004?>");`	`7 = 2 + 5`
`pwa.getUpperCaseBonus("UPPERlower");`	`10 = 5 + 5`
`pwa.getUpperCaseBonus("CS12345678");`	`7 = 2 + 5`
`pwa.getUpperCaseBonus ("i.luv.programming");`	`5 = 0 + 5`
`pwa.getUpperCaseBonus ("<2015@Wittry@Contest>");`	`7 = 2 + 5`

The `getDigitBonus` method has a single `String pw` parameter and returns an `int`. `getDigitBonus` returns the number of bonus points `pw` is entitled from the digits `pw` contains. If a `pw` is not valid password, zero digit bonus points are awarded. The possible bonus points are awarded as follows:

- Digit (number) Bonus (total max = 15 points)
  - +1 for each digit contained in the password. (The digits 10 counts as two numbers. Max of 10 points)
  - +1 for each non digit (lower/upper case symbols, or symbols, max 5 points)

The following code shows the results of the `getDigitBonus` method.

The following code	Returns
`PasswordAnalyzer pwa = new PasswordAnalyzer();`	
`pwa.getDigitBonus("@Digit27");`	`7 = 2 + 5`
`pwa.getDigitBonus("<#CS$$2004?>");`	`9 = 4 + 5`
`pwa.getDigitBonus("i.luv.programming");`	`5 = 0 + 5`
`pwa.getDigitBonus("<2015@Wittry@Contest>");`	`9 = 4 + 5`

The `getSymbolBonus` method has a single `String pw` parameter and returns an `int`. `getSymbolBonus` returns the number of Symbol points `pw` is entitled from the Symbols `pw` contains. If a `pw` is not valid password, zero Symbol bonus points are awarded. The possible bonus points are awarded as follows:

- Symbol Bonus (total max = 15 points)
  A symbol is any character that is not a letter and not a digit/number)
  - +1 for each Symbol contained in the password. (max 10 points)
  - +1 for each non Symbol (lower/upper case letter, or digit, max 5 points)

The following code shows the results of the `getSymbolBonus` method.

The following code	Returns
`PasswordAnalyzer pwa = new PasswordAnalyzer();`	
`pwa.getSymbolBonus("@Symbol!!");`	`8 = 3 + 5`
`pwa.getSymbolBonus("<#CS$$2004?>");`	`11 = 6 + 5`
`pwa.getSymbolBonus("i.luv.programming");`	`7 = 2 + 5`
`pwa.getSymbolBonus("<2015@Wittry@Contest>");`	`9 = 4 + 5`

The `getCombinationBonus` method has a single `String pw` parameter and returns an `int`. `getCombinationBonus` returns the number of Combination points `pw` is entitled from the Combinations `pw` contains. If a `pw` is not valid password, zero Combination bonus points are awarded. The possible bonus points are awarded as follows:

Only valid passwords are eligible to receive bonus points. The possible bonus points are described below.

- Combination Bonus (total max = 10 points)
  A combination is defined to be any of the following
  - Letter (upper of lower case) followed by a digit/number
  - letter (upper of lower case) followed by a symbol
  - Digit/number followed by a symbol
  - Symbol followed by a digit/number

  The password containing “A7#” has two combinations
  - +1 for each combination in the password. (max 10 points)

The following code shows the results of the `getCombinationBonus` method.

The following code	Returns
`PasswordAnalyzer pwa = new PasswordAnalyzer();`	
`pwa.getCombinationBonus("@Symbol!!");`	`1`
`pwa.getCombinationBonus("<#CS$$2004?>");`	`3`
`pwa.getCombinationBonus("i.luv.programming");`	`2`
`pwa.getCombinationBonus("<2015@Wittry@Contest>");`	`4`

The `getPoints` method has a single `String pw` parameter and returns an `int`. `getPoints` returns the total number of points `pw` is entitled. The total points pw is entitled is calculated by starting with a score of 50 points and then adding all the bonus points. The minimum score of all passwords (including invalid passwords) is 50.

The following code shows the results of the `getPoints` method.

The following code	Returns
`PasswordAnalyzer pwa = new PasswordAnalyzer();`	
`pwa.getPoints("word"));`	`50`
`pwa.getPoints("<#CS$$2004?>"));`	`72 = 50 + 4 + 7 + 9 + 11 + 3`
`pwa.getPoints("i.luv.programming");`	`84 = 50 + 15 + 5 + 5 + 7 + 2`
`pwa.getPoints("<2015@Wittry@Contest>");`	`98 = 50 + 19 + 7 + 9 + 9 + 4`

The `getRating` method has a single `String pw` parameter and returns a `String`. `getRating` returns the rating of the password `pw`. Passwords that are not valid receive a rating of "`rejected`". Valid password will be given one of five different ratings. Valid password with 70 or fewer points received a rating of "`unacceptable`". Passwords receiving more than 70 points and 80 points or less receive a rating of "`weak`". Passwords receiving more than 80 points and 95 points or less receive a rating of "`average`". Passwords receiving more than 95 points and 115 points or less receive a rating of "`good`". Passwords receiving more than 115 points receive a rating of "`excellent`".

The following code shows the results of the `getRating` method.

The following code	Returns
`PasswordAnalyzer pwa = new PasswordAnalyzer();`	
`pwa.getRating("word");`	`"rejected"`
`pwa.getRating("i.luv.programming");`	`"average"`
`pwa.getRating("<2015@Wittry@Contest>");`	`"good"`

The following `Character` static methods may be useful:

`static Boolean`	**`isDigit`**`(char ch)`
`static Boolean`	**`isLetter`**`(char ch)`
`static Boolean`	**`isLetterOrDigit`**`(char ch)`
`static Boolean`	**`isLowerCase`**`(char ch)`
`static Boolean`	**`isUpperCase`**`(char ch)`

Some sample usages of these methods are listed below with the return value

Method call	return value
`Character.isDigit``("3".charAt(0))`	**`true`**
`Character.isLetter` `("3".charAt(0))`	**`false`**
`Character.isLetterOrDigit` `("3".charAt(0))`	**`true`**
`Character.isLetterOrDigit` `("A".charAt(0))`	**`true`**
`Character.isLowerCase` `("A".charAt(0))`	**`false`**
`Character.isUpperCase` `("A".charAt(0))`	**`true`**