COP5007: Java Programming

Programming Project # 1

See Assignment for due date

Project Outcomes:

Create a Java application that uses:

- decision constructs
- looping constructs
- more than one class and has multiple objects

Prep Readings:

Zybooks chapter 1 - 7

Background Information:

The Unified Modeling Language (UML) provides a useful notation for designing and developing object-oriented software systems. One of the basic components of the UML is a class diagram, which are used to depict the attributes and behaviors of a class. A basic class diagram (as shown in the figure below) has three components. The first is the class name. The second component includes the class's attributes or fields. Each attribute is followed by a colon (:) and its data type. The third component includes the class's behaviors or methods. If the method takes parameters, their types are included in parentheses. Each behavior is also followed by a colon (:) and its return type. If the return value of a method is void, the return type can be omitted. For more information on the UML, refer to http://www.uml.org/.

Project Requirements:

1) This project will use The Vigenere Cipher to encrypt passwords.

2) Vigenere Cipher
   a) Simple letter substitution ciphers are ones in which one letter is substituted for another. Although their output looks impossible to read, they are easy to break because the relative frequencies of English letters are known.

   b) The Vigenere cipher improves upon this. They require a key word and the plain text to be encrypted.

c) Example:

Suppose you have a key word: *house*

We would encrypt a message as follows:

The first letter would be encrypted using these correspondences:
original letter: a b c d e f g h i j k l m n o p q r s t u v w x y z
substitute letter: **h** i j k l m n o p q r s t u v w x y z a b c d e f g

The second letter would be encrypted using these correspondences:
original letter: a b c d e f g h i j k l m n o p q r s t u v w x y z
substitute letter: **o** p q r s t u v w x y z a b c d e f g h i j k l m n

The third letter would be encrypted using these correspondences:
original letter: a b c d e f g h i j k l m n o p q r s t u v w x y z
substitute letter: **u** v w x y z a b c d e f g h i j k l m n o p q r s t

etc

Once all characters in the key are used, the offset repeats starting over with the Key's first letter.

3) The cipher for this program is a little more complex as you must include punctuation so the password may be any of the characters in the ASCII table from character 33 the exclamation point '*!*' to character 122 the lower case '*z*'.

4) User Class
   a) The class that contains user information including the username, password, encrypted password and key (see UML class diagram below)

User
username : String clearPassword : String encryptedPassword : String key : String
+User() +User(String, String, String) +getUserName() : String +setUserName(String) +getClearPassword(): String +setClearPassword(String) +getEncryptedPassword():String +getKey(): String +setKey(String) -encrypt() +toString():String

   b) Constructors
      i) Default constructor sets all fields to the empty String ("")

      ii) Parameterized constructors

         (1) takes in the username, clearPassword and the key

         (2) calls the private method encrypt to encrypt the clearPassword using the Vigenere Cypher.

   c) Methods
      i) Accessor and mutator methods as listed in the Class Diagram

         (1) The setClearPassword method calls the private method encrypt to encrypt the clearPassword using the Vigenere Cypher.
         (2) Note no mutator for the enxyrptedPassword as that is only set or changed via the encrypt method.
      ii) encrypt method

         (1) Private method – only accessible to other methods of the User class.

(1) Uses the Vigenere Cypher to encrypts the clearPassword instance variable using the key

(2) Stores the encrypted password in encryptPassword instance variable.

ii) toString – returns a nicely formatted String representing the user to include username, encrypted password, clear password and key such as: *Jsmith]Umka\f^ password house*

5) UserTester

    a) The purpose of this class is to thoroughly test all the methods and constructors in the User class.

    b) Methods can be tested directly or indirectly

    c) Indirect testing is testing a method by calling another method such as

        i) If the constructor’s calls the mutator methods when setting instance variables rather than setting them directly.

        ii) If the toString method calls accessor method rather than accessing instance variables directly

    d) No user input, just create variables and/or hardcode values

        i) To simplify the program no error handling required just make sure you enter the data as specified

            (1) All passwords are 8 legal characters (as outlined in the requirements)

            (2) All keys are 5 characters

6) Programming/Client Communication
    a) When developing a program requirement are rarely perfect, ambiguities, omissions and errors are usually present.

    b) To keep this communication organized please post to the appropriate Programming Project Topic any question you may have. I will answer those question there and thus we will have a written record.

Submission Requirements:

Your project must be submitted using the instructions below. Any submissions that do not follow the stated requirements will not be graded.

1. You should have the following files for this assignment:
    - User.java - The User class
    - UserTester.java – The testing class for this project
    - The javadoc files for the User class (Do not turn in)
        - User.html

2. Remember to compile and run your program one last time before you submit it. If your program will not compile, the graders will not be responsible for trying to test it.
3. Follow the submission requirements posted on elearning.

Important Notes:

1. Projects will be graded on whether they correctly solve the problem, and whether they adhere to good programming practices.
2. Projects must be submitted by the time specified on the due date. Projects submitted after that time will get a grade of zero.
3. Please review UWFs academic conduct policy. Note that viewing another student's solution, whether in whole or in part, is considered academic dishonesty. Also note that submitting code obtained through the Internet or other sources, whether in whole or in part, is considered academic dishonesty. All programs submitted will be reviewed for evidence of academic dishonesty, and all violations will be handled accordingly.