1. A class to encode texts Specification: Realize a Java class to represent encrypted texts. The encryption of the text is obtained by replacing each character with the character whose code is equal to the code of the character to encode augmented by an integer number representing the encryption key.

The functionalities of encrypted texts are:

1. creation of a new object that represents a text encrypted with a given encryption key;
2. creation of a new object that represents a text that is not encrypted (yet);
3. return of the encrypted text;
4. return of the decrypted text, provided the correct encryption key is provided;
5. verification of the correctness of an encryption key;
6. Modification of the encryption key; this is possible only if the correct key is provided.

The class EncryptedText: realization of the methods We concentrate now on the single methods and realize their bodies. To do so, we make use of two auxiliary methods, encode and decode, analogous to the methods for encoding and decoding text according to a given key that we have already seen.

**package** com.strings.demo;

**public** **class** EncryptedText {

**private** **int** key;

**private** String text;

// constructors

**public** EncryptedText(String nonEncryptedText) {

key = 0;

text = nonEncryptedText;

}

**public** EncryptedText(String nonEncryptedText, **int** key) {

**this**.key = key;

text = *encode*(nonEncryptedText,key);

}

**public** String getEncryptedText() {

**return** text;

}

**public** String getDecryptedText(**int** key) {

**if** (key == **this**.key)

**return** *decode*(text, key);

**else**

**return** **null**;

}

**public** **boolean** isKey(**int** candidateKey) {

**return** candidateKey == key;

}

**public** **void** setKey(**int** key, **int** newKey) {

**if** (key == **this**.key) {

**this**.key = newKey;

text = *encode*(*decode*(text,key),newKey);

}

}

// auxiliary methods

**private** **static** String encode(String text, **int** key) {

String resText;

**char** c;

**int** ci;

resText = "";

**for** (**int** i = 0; i < text.length(); i++) {

c = text.charAt(i);

ci = (**int**)c;

ci = ci + key;

c = (**char**)ci;

resText = resText + String.*valueOf*(c);

}

**return** resText;

}

**private** **static** String decode(String text, **int** key) {

String resText;

**char** c;

**int** ci;

resText = "";

**for** (**int** i = 0; i < text.length(); i++) {

c = text.charAt(i);

ci = (**int**)c;

ci = ci - key;

c = (**char**)ci;

resText = resText + String.*valueOf*(c);

}

**return** resText;

}

}

The class EncryptedTextMain: example of a client We realize the class ClientEncryptedText, which contains a methdo main that uses the class EncryptedText:

**package** com.strings.demo;

**public** **class** EncryptedTextMain {

**public** **static** **void** main(String[] args) {

EncryptedText t = **new** EncryptedText("Hello friend ...", 10);

System.***out***.println("Encrypted: " + t.getEncryptedText());

System.***out***.println("Decrypted: " + t.getDecryptedText(10));

t.setKey(10,20);

**if**(t.getDecryptedText(10) == **null**)

System.***out***.println("We can not decrypt the string.");

System.***out***.println(t.getDecryptedText(20));

}

}

1. Write a program that prints length of a given string.
2. Write a program to accept a string and print that string. If string is empty throw a user defined exception.
3. Write a program to print the accepted string changed to uppercase.
4. Write a program to print the accepted string changed to lowercase.
5. Write a program that reads two strings from input and check if they are equal or equal with case ignored or they are not equal.
6. Write a program to accept two strings from the keyboard and outputs if one is part of the other.
7. Write a program to accept two strings from the keyboard and outputs if one word is prefix of the other.
8. Write a program to accept two strings from the keyboard and outputs if one word is suffix of the other.
9. Write a program that prints the index and the character for an accepted word.
10. Write a program that takes two strings and prints where in the first string the second string starts.

String1: This is a test

String2: is

‘is’ can be found at index 2.

1. Write a program that counts how many ‘e’ s and how many ‘u’ s are there in the following text.

“JUnit 4 has been here for quite a long time, and there are numerous tests written in junit 4. JUnit Jupiter needs to support those tests as well. For this purpose, the JUnit Vintage sub-project was developed.”

1. Modify the above program to print how many times each letter appear in the above given text.
2. Write a method that reverses a string.
3. Write a method which checks if a word is a palindrome.