**Name: Hasib Rostaiee**

**Class: CMSC203- 32715**

**Instructor: Farnaz Eivazi**

**Date: 03/12/2020**

**Assignment: #3 – Design and Test Cases**

|  |
| --- |
| **CryptoJUnitTest** |
|  |
| +setUp( ) : void |
| +tearDown( ): void |
| +testStringBounds( ): void |
| +testEnreypCaesar( ): void |
| +testDecryptCaesar( ): void |
| +testEncryptBellaso( ): void |
| +testDecryptBellaso( ): void |

|  |
| --- |
| **CryptoManagerTest** |
|  |
| +main() : void |

|  |
| --- |
| **CryptoManager** |
| -LOWER\_BOUND: char |
| -UPPER\_BOUND: char |
| -RANGE: int |
| +stringInBounds(plainText: String): boolean |
| +encryptCaesar(plainText: String, key: int): String |
| +encryptBellaso(plainText: String, bellasoStr: String): String |
| +decryptCaesar(encryptedText: String, key: int): String |
| +decryptBellaso(encryptedText: String, bellasoStr: String): String |

|  |
| --- |
| **FXDriver** |
|  |
| +main() : void |
| +start( ) : void |

|  |
| --- |
| **FXMainPane** |
| -decryption: Button |
| -exitButton: Button |
| -enryption: Button |
| -test: Button |
| -clearButton: Button |
| -plainTextTextField: TextField |
| -inputForEncryptionTextField: TextField |
| -encryptedStringTextField3: TextField |
| -decryptedTextField4: TextField |
| -plainTextLabel: Label |
| -descriptionForInputLabel: Label |
| -encryptedLabel3: Label |
| -decriptedLabel4: Label |
| -blankLabel1: Label |
| -blankLabel2: Label |
| -blankLabel3: Label |
| -blankLabel4: Label |
| -radioButton1: radioButton |
| -radioButton2: radioButton |
| -shiftInt: int |
| +FXMainPane() |

**Pseudocodes:**

* **Method: public static boolean stringInBounds(String plainText)**
  1. *Initiate return variable IsInbound and set it to true b default*
  2. *Get character of the string which is passed to the method*
  3. *If the character is not within the range, set return value to false*
  4. *Continue step 3 for all characters in the text.*
  5. *Return the IsInbound variable.*
* **public static String encryptCaesar(String plainText, int key)**
  1. *If the Key passed in the method greater than the UPPER\_BOUND.value, keep decreasing the key by range value (here , 64) until it’s less than UPPER\_BOUND.*
  2. *Define return variable (encrypted) and by default set to empty*
  3. *Get character of the string passed to the method*
  4. *Add the above character value to the key value*
  5. *If the result of step 4 is more than the UPPER\_BOUND value, keep decreasing it until it’s less than UPPER\_BOUND*
  6. *Add the results from step 4 or 5 into encrypted variable*
  7. *Continue steps 3-6 for all characters of the string passed in the method*
  8. *Return encrypted value*
* **public static String encryptBellaso(String plainText, String bellasoStr)**
  1. *Define return variable (encrypted) and by default set to empty*
  2. *If length of bellaso string (key), is less than the length of the text passed in the method, get the difference of the length and add characters of beginning of key to the end of it until length of key string become same as the length of plain text string.*
  3. *For each character of the plain text passed, in the method, add the ASCII code of it to the ASCII code of character of key for the same index.*
  4. *If the above value become more than UPPER\_BOUND, keep decreasing the value by range number until it become less than UPPER\_BOUDN.*
  5. *Change the result back to character and add it to encrypted variable*
  6. *Return encrypted variable.*
* **public static String decryptCaesar(String encryptedText, int key)**
  1. *If the Key passed in the method greater than the UPPER\_BOUND.value, keep decreasing the key by range value (here , 64) until it’s less than UPPER\_BOUND.*
  2. *Define return variable (decrypted) and by default set to empty*
  3. *Get character of the text to be decrypted, and subtract the key value*
  4. *If the result value becomes less than LOWER\_BOUND, increase it by range value to be greater than LOWER\_BOUND*
  5. *Add the character of step 4 to the decrypted variable*
  6. *Return decrypted variable*
* ***public static String decryptBellaso(String encryptedText, String bellasoStr)***
  1. *If length to the key string is less than encrypted text,*
  2. *add the characters from beginning at the end of key string until the length become same as the encrypted text length*
  3. *Get character of the decrypted text and subtract the character of key string*
  4. *If the results become less than LOWER\_BOUND, increase it by RANGE value.*
  5. *Add the results to the return variable “encrypted” which is initially set to empty.*
  6. *Repeat teste step 2-4 for all characters of the encrypted text passed to the method.*

**Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Encryption/ Decryption Method | Inputs | | Encrypted Result | Decrypted Results (of encrypted ) | Pass / Fail |
| **Text** | **Key** |
| Caesar cipher | TEST TEXT | 36 | 8)78D8)<8 | TEST TEXT | **PASS** |
| Caesar cipher | NEW TEST | 900 | RI[$XIWX | NEW TEST | **PASS** |
| Bellaso cipher | TEST | ABCD | UGVX | ABCD | **PASS** |
| Bellaso cipher | TESTING 2 | ABC | UGVUKQH"5 | TESTING 2 | **PASS** |
| Bellaso cipher | TEST3 | KEY IS LONG | \_J,4< | TEST3 | **PASS** |
| Caesar cipher | {INVALID} | 64 | Text Contains Invalid Character\* | | **PASS** |
| Bellaso cipher | {INVALID} | ABC | Text Contains Invalid Character\* | | **PASS** |
| Bellaso cipher | VALID TEXT | {INVALID KEY} | Key Contains Invalid Character\* | | **PASS** |

*\*This will be printed in “Encrypted” field, to avoid a run-time error if invalid texts are passed.*