Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергеніці

Домашнє завдання № 14

з курсу: "Розробка програмного забезпечення мобільних пристроїв"

**Виконав:**  
студентка 4-го курсу,  
групи ТІ-01  
Круть Катерина Олександрівна

**Перевірив:**

Недашківський О.Л.

Київ 2023/2024

**Домашнє завдання № 14**

**Завдання:**

1. Написати програму, що замінює підстрічку в файлі заданою стрічкою

2. Написати програму, що копіює зміст текстового файлу в інший з розширення

«.bak». Назва файлу вводиться користувачем.

3. Написати програму, що відкриває файл на читання і формує два інших файли.

Перший файл формується з непарних стрічок початкового файлу, а інший з парних.

4. Здійснити шифрування файлу використовуючи зсув символів.

5. Здійснити розшифрування файлу зашифрованого вище

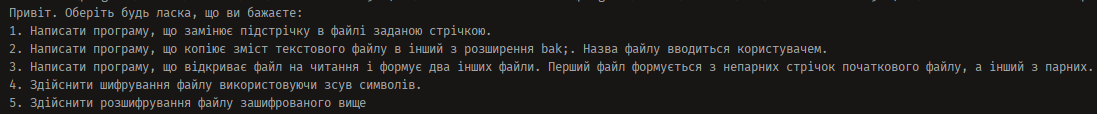
**Хід виконання:**

Під час виконання програми, користувач працює з меню, яке надає змогу вибрати опцію до виконання (номер завдання).

**Main.kt**

fun main() {  
 do {  
 when(*menu*()) {  
 1 -> FileWorker().fileReplace()  
 2 -> FileWorker().bak()  
 3 -> FileWorker().oddAndEvenFiles()  
 4 -> FileWorker(1).cipher()  
 5 -> FileWorker(2).cipher()  
 }  
 *println*("Для завершення введіть 'q', інакше натисніть Enter")  
 } while (*readLine*() != "q")  
}  
  
*/\*\*  
 \* Головне меню програми  
 \*/*fun menu(): Int {  
 *println*("""  
 1. Заміна substr в файлі  
 2. Копіювання файлу з розширенням ".bak"  
 3. Непарні та парні строки  
 4. Шифрування файлу  
 5. Розшифрування файлу""".*trimIndent*())  
  
 return *generateSequence* **{** *readLine*()?.*toIntOrNull*()?.*takeIf* **{ it** in 1..5 **}  
 }**.*first*()  
}

Виконання:



Для роботи з файлами було створено клас «FileWorker» в якому було реалізавно всі методи для виконання завдання:

* створення нового файлу
* відкриття файлу
* читання тексту із файлу
* заміна тексту в файлі

**FileWorker.kt**

import java.io.File  
  
*/\*\*  
 \* Повний шлях до папки resources з файлами для тестування  
 \*/*const val *ROOT\_URL* = "/Users/katiakrut/IdeaProjects/KotlinProjects/Homework14/src/main/resources"  
  
  
*/\*\*  
 \* Клас для роботи з файлами  
 \*/*class FileWorker(private val choose: Int = 0) {  
  
 */\*\*  
 \* Замінює підстрічку в файлі заданою стрічкою.  
 \* Якщо файл не існує, повертає "File does not exist" повідомлення.  
 \*/* fun fileReplace() {  
 val file = openFile() ?: return *println*("File does not exist")  
 *println*("Введіть підстрічку для заміни: ")  
 val txt1 = *readLine*() ?: ""  
 *println*("Введіть підстрічку, на яку замінити: ")  
 val txt2 = *readLine*() ?: ""  
 file.*writeText*(file.*readText*().*replace*(txt1, txt2))  
 *println*("Done!")  
 }  
  
 */\*\*  
 \* Створює копію файлу з розширенням .bak.  
 \* Якщо файл не існує, повертає "File does not exist" повідомлення.  
 \* Якщо файл з такою назвою вже існує, повертає "#ERROR - файл вже є!" повідомлення.  
 \*/* fun bak() {  
 val file = openFile() ?: return *println*("File does not exist")  
 *println*("Enter file name!")  
 val newFile = File("$*ROOT\_URL*/${*readLine*()}.bak")  
 if (newFile.createNewFile()) {  
 newFile.*writeText*(file.*readText*())  
 *println*("Done!")  
 } else {  
 *println*("#ERROR - файл вже є!")  
 }  
 }  
  
 */\*\*  
 \* Формує два нових файли: один з непарних рядків вихідного файлу, інший - з парних.  
 \* Якщо вихідний файл не існує, повертає "File does not exist" повідомлення.  
 \*/* fun oddAndEvenFiles() {  
 val file = openFile() ?: return *println*("File does not exist")  
 val lines = file.*readLines*()  
 val oddLines = lines.*filterIndexed* **{** index, \_ **->** index % 2 == 0 **}** val evenLines = lines.*filterIndexed* **{** index, \_ **->** index % 2 != 0 **}** File(newFile()).*writeText*(oddLines.*joinToString*("\n"))  
 File(newFile()).*writeText*(evenLines.*joinToString*("\n"))  
 *println*("Done!")  
 }  
  
 */\*\*  
 \* Шифрує або розшифровує вміст файлу використовуючи ключ.  
 \* Якщо файл не існує, повертає "File does not exist" повідомлення.  
 \** ***@param*** *choose Визначає режим роботи: 1 для шифрування, 2 для розшифровки.  
 \*/* fun cipher() {  
 val file = openFile() ?: return *println*("File does not exist")  
 *println*("Enter key. Example: 'key'")  
 val key = *readLine*() ?: ""  
 val lines = file.*readLines*()  
 val processedLines = when (choose) {  
 1 -> lines.*filter* **{ it**.*isNotEmpty*() **}**.*map* **{** Cipher().encryption(**it**, key) **}** 2 -> lines.*filter* **{ it**.*isNotEmpty*() **}**.*map* **{** Cipher().decryption(**it**, key) **}** else -> lines  
 }  
 File(newFile()).*writeText*(processedLines.*joinToString*("\n"))  
 *println*("Done!")  
 }  
  
 */\*\*  
 \* Відкриває файл для читання.  
 \* Якщо файл не існує, повертає `null` і виводить повідомлення про помилку -- "#ERROR - файл не існує".  
 \** ***@return*** *Об'єкт File | `null`.  
 \*/* private fun openFile(): File? {  
 *println*("Opening file!\nВедіть filename без формату: ")  
 val fileName = "$*ROOT\_URL*/${*readLine*()}.txt"  
 return File(fileName).*takeIf* **{ it**.exists() **}** ?: *run* **{** *println*("#ERROR - файл не існує")  
 null  
 **}** }  
  
 */\*\*  
 \* Створює новий файл і повертає шлях до нього.  
 \* Якщо файл вже існує, виводить повідомлення про помилку -- "#ERROR - файл вже є!".  
 \** ***@return*** *Шлях до створеного файлу.  
 \*/* private fun newFile(): String {  
 *println*("Введіть filename без формату:")  
 val fileName = "$*ROOT\_URL*/${*readLine*()}.txt"  
 *println*(File(fileName).*takeIf* **{ it**.createNewFile() **}**?.*let* **{** "File created!" **}** ?: "#ERROR - файл вже є!")  
 return fileName  
 }  
}

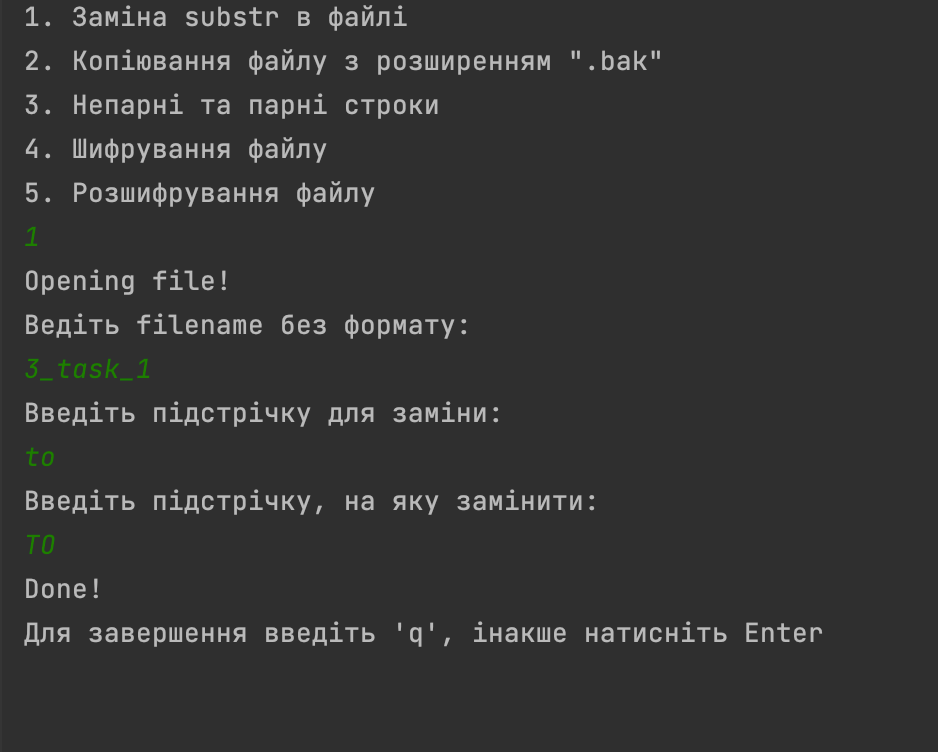
**Завдання 1:** Написати програму, що замінює підстрічку в файлі заданою стрічкою.

fun fileReplace() {  
 val file = openFile() ?: return *println*("File does not exist")  
 *println*("Введіть підстрічку для заміни: ")  
 val txt1 = *readLine*() ?: ""  
 *println*("Введіть підстрічку, на яку замінити: ")  
 val txt2 = *readLine*() ?: ""  
 file.*writeText*(file.*readText*().*replace*(txt1, txt2))  
 *println*("Done!")  
}

Виконання:

В папці «resourses» було створено текстовий 3\_task\_.txt файл із текстом:

After a while finding that nothing more  
at once but alas for poor Alice when she got  
little golden key and when she went back to  
reach it she could see it quite plainly through  
one of the legs of the table but it was too  
with trying THE poor little thing sat down and cried



Результат:

After a while finding that nothing more  
at once but alas for poor Alice when she got  
little golden key and when she went back TO  
reach it she could see it quite plainly through  
one of THE legs of the table but it was TOo  
with trying the poor little thing sat down and cried

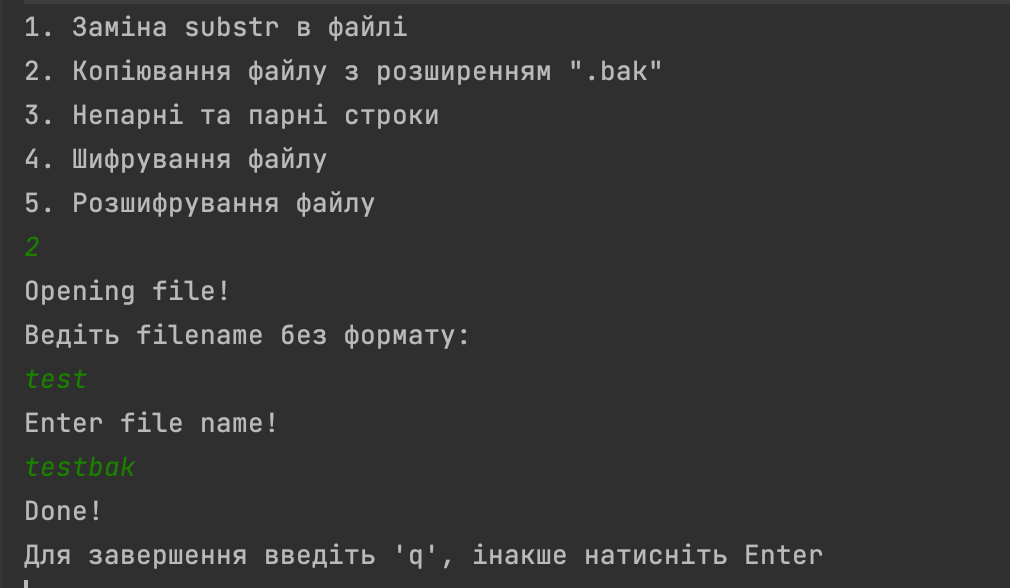
**Завдання 2:** Написати програму, що копіює зміст текстового файлу в інший з розширення «.bak». Назва файлу вводиться користувачем.

fun bak() {  
 val file = openFile() ?: return *println*("File does not exist")  
 *println*("Enter file name!")  
 val newFile = File("$*ROOT\_URL*/${*readLine*()}.bak")  
 if (newFile.createNewFile()) {  
 newFile.*writeText*(file.*readText*())  
 *println*("Done!")  
 } else {  
 *println*("#ERROR - файл вже є!")  
 }  
}

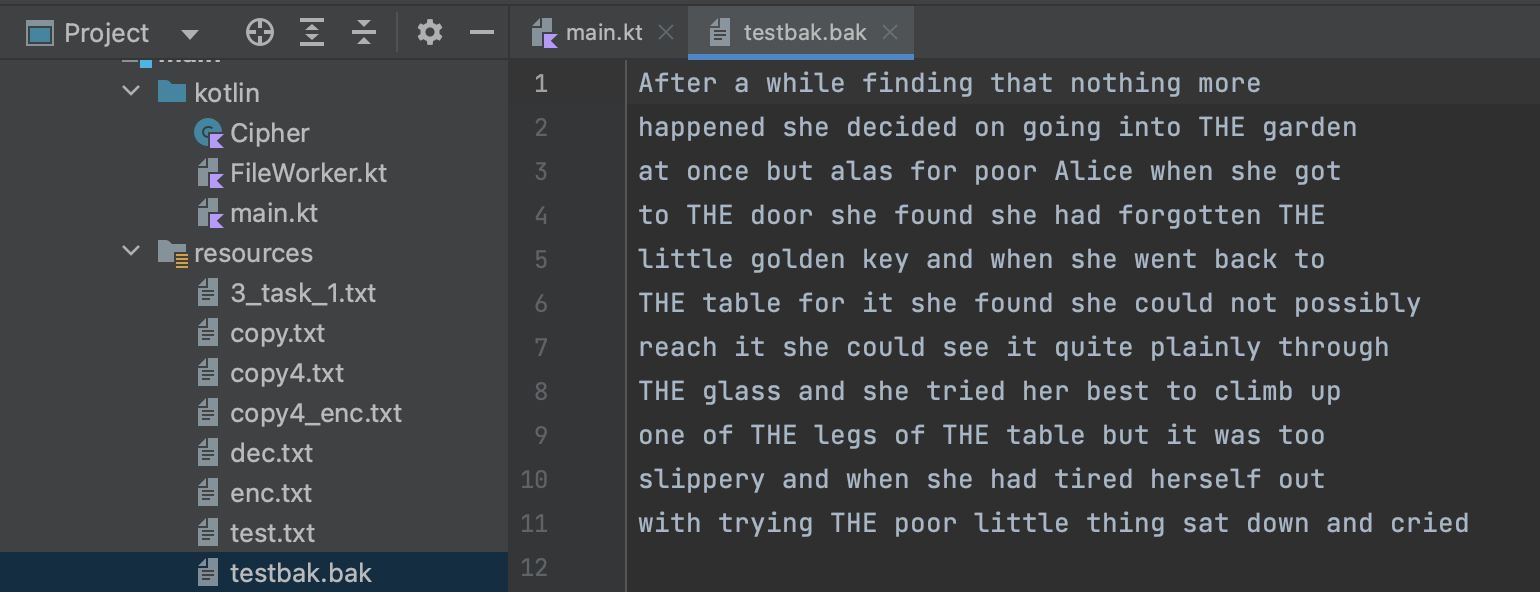
Виконання:

В папці «resourses» було створено текстовий файл із текстом

After a while finding that nothing more  
happened she decided on going into THE garden  
at once but alas for poor Alice when she got  
to THE door she found she had forgotten THE  
little golden key and when she went back to  
THE table for it she found she could not possibly  
reach it she could see it quite plainly through  
THE glass and she tried her best to climb up  
one of THE legs of THE table but it was too  
slippery and when she had tired herself out  
with trying THE poor little thing sat down and cried



Результат:



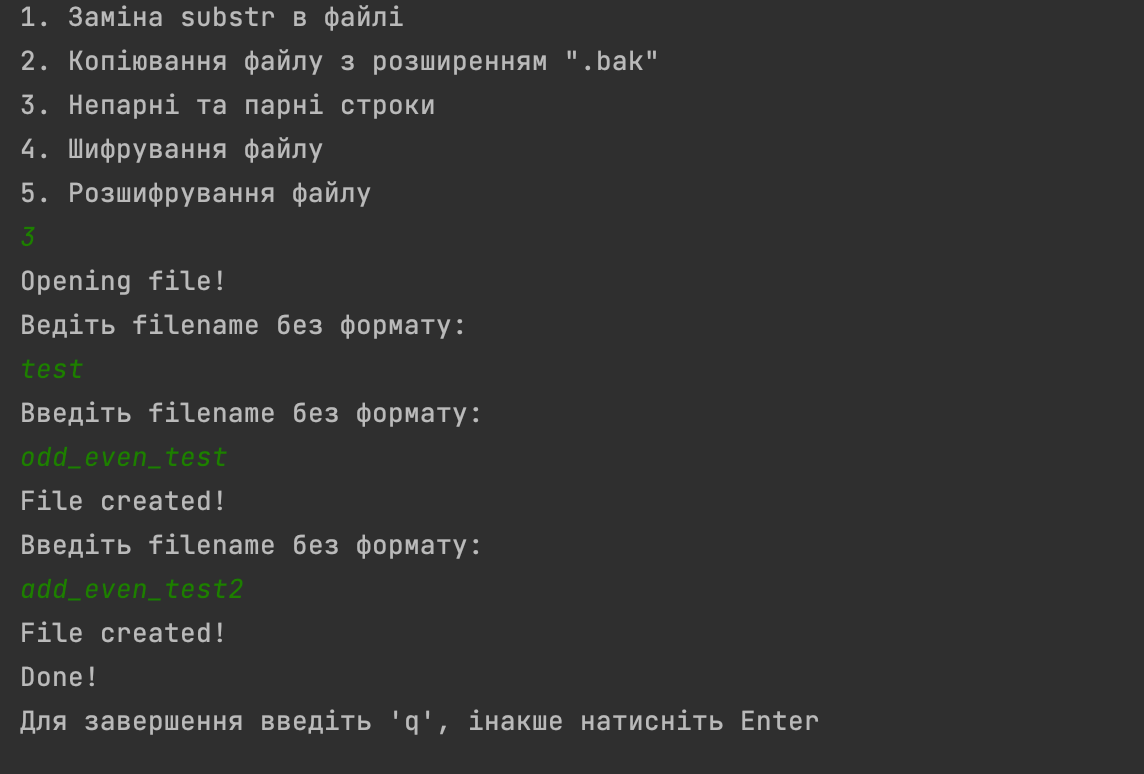
**Завдання 3:** Написати програму, що відкриває файл на читання і формує два інших файли. Перший файл формується з непарних стрічок початкового файлу, а інший з парних.

fun oddAndEvenFiles() {  
 val file = openFile() ?: return *println*("File does not exist")  
 val lines = file.*readLines*()  
 val oddLines = lines.*filterIndexed* **{** index, \_ **->** index % 2 == 0 **}** val evenLines = lines.*filterIndexed* **{** index, \_ **->** index % 2 != 0 **}** File(newFile()).*writeText*(oddLines.*joinToString*("\n"))  
 File(newFile()).*writeText*(evenLines.*joinToString*("\n"))  
 *println*("Done!")  
}

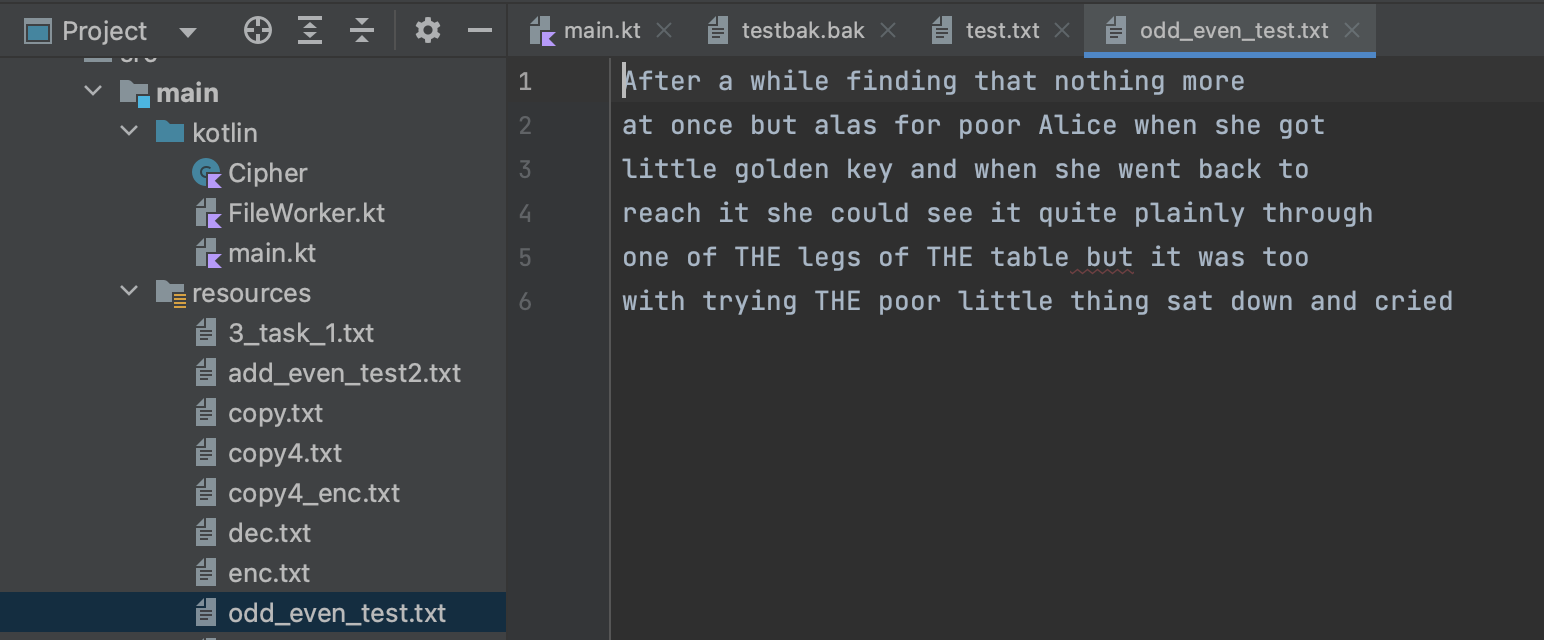
Виконання:

В папці «resourses» було створено текстовий файл із текстом:

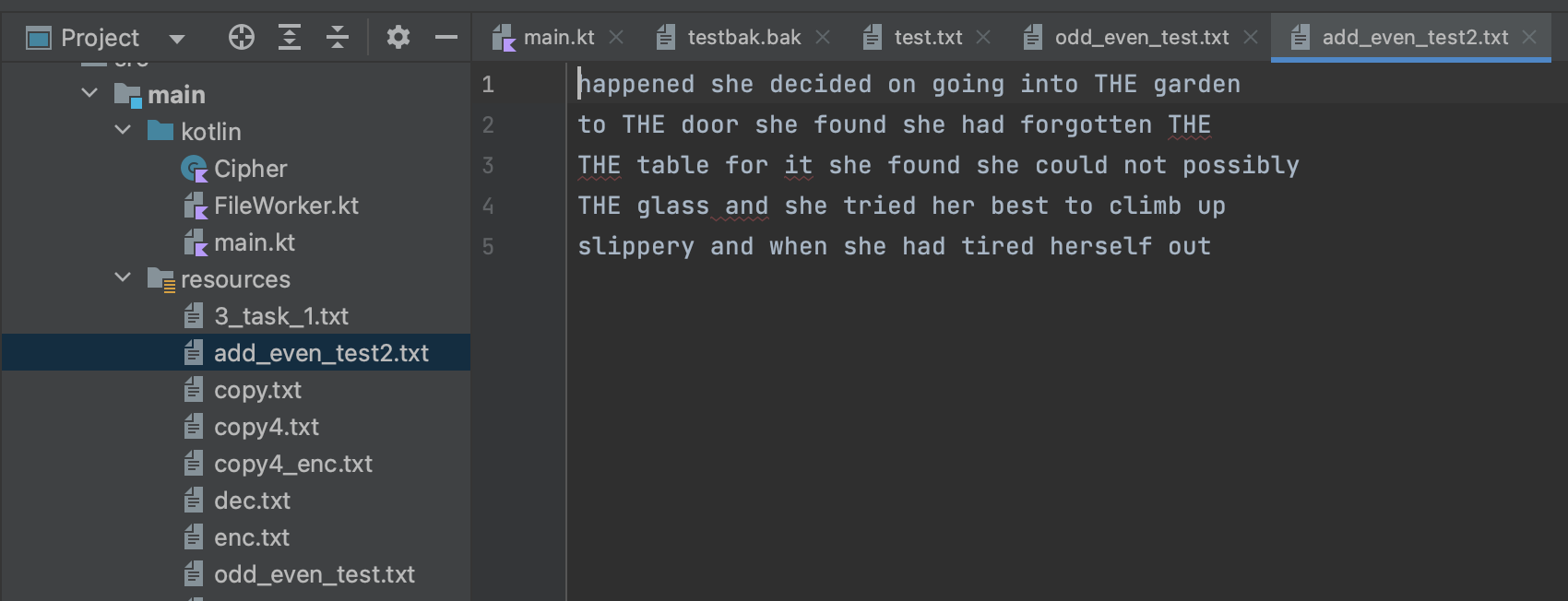
After a while finding that nothing more  
happened she decided on going into THE garden  
at once but alas for poor Alice when she got  
to THE door she found she had forgotten THE  
little golden key and when she went back to  
THE table for it she found she could not possibly  
reach it she could see it quite plainly through  
THE glass and she tried her best to climb up  
one of THE legs of THE table but it was too  
slippery and when she had tired herself out  
with trying THE poor little thing sat down and cried



Файл 1:



Файл2 :



**Завдання 4:** Здійснити шифрування файлу використовуючи зсув символів.

Для реалізації цього завдання та завдання 5 було обрано шифрування методом Віженера. Та реалізован спеціальний клас для цього завдання:

*/\*\*  
 \* Клас для шифрування та розшифрування тексту методом Віженера  
 \*/*class Cipher(private var text: String = "", private var key: String = "") {  
 private val alf = alphabet()  
  
 */\*\*  
 \* Шифрує вхідний текст за допомогою ключа  
 \*  
 \** ***@param*** *inputText Текст, який потрібно зашифрувати  
 \** ***@param*** *inputKey Ключ для шифрування  
 \** ***@return*** *Зашифрований текст  
 \*/* fun encryption(inputText: String, inputKey: String): String {  
 key = inputKey  
 val textNumbers = convertToNumbers(inputText.*lowercase*())  
 val keyNumbers = convertToNumbers(generateBigKey(inputText.length))  
  
 textNumbers.*indices*.*forEach* **{** i **->** textNumbers[i] += keyNumbers[i] **}** return *buildString* **{** textNumbers.*forEach* **{** value **->** val adjustedValue = if (value >= alf.length) value - alf.length else value  
 append(alf[adjustedValue])  
 **}  
 }** }  
  
 */\*\*  
 \* Розшифровує вхідний текст за допомогою ключа  
 \*  
 \** ***@param*** *inputText Текст, який потрібно розшифрувати  
 \** ***@param*** *inputKey Ключ для розшифрування  
 \** ***@return*** *Розшифрований текст  
 \*/* fun decryption(inputText: String, inputKey: String): String {  
 key = inputKey  
 val textNumbers = convertToNumbers(inputText.*lowercase*())  
 val keyNumbers = convertToNumbers(generateBigKey(inputText.length))  
  
 textNumbers.*indices*.*forEach* **{** i **->** textNumbers[i] -= keyNumbers[i] **}** return *buildString* **{** textNumbers.*forEach* **{** value **->** append(alf[if (value < 0) alf.length + value else value]) **}  
 }** }  
  
 */\*\*  
 \* Генерує алфавіт для шифрування та розшифрування  
 \** ***@return*** *Стрічка, яка містить алфавіт  
 \*/* private fun alphabet() = *buildString* **{** append(" ")  
 append(('a'..'z').*joinToString*(""))  
 **}** */\*\*  
 \* Генерує ключ необхідної довжини для шифрування/розшифрування  
 \** ***@param*** *length Довжина потрібного ключа  
 \** ***@return*** *Стрічка, яка містить ключ необхідної довжини  
 \*/* private fun generateBigKey(length: Int) = *buildString* **{** *repeat*((length + key.length - 1) / key.length) **{** append(key) **}  
 }**.*take*(length)  
  
  
 */\*\*  
 \* Конвертує вхідний текст у числове представлення на основі алфавіту  
 \** ***@param*** *input Вхідний текст для конвертації  
 \** ***@return*** *Масив цілих чисел, який представляє текст  
 \** ***@throws*** *IllegalArgumentException Якщо символ відсутній в алфавіті  
 \*/* private fun convertToNumbers(input: String) = IntArray(input.length) **{** val index = alf.*indexOf*(input[**it**])  
 if (index == -1) throw IllegalArgumentException("Символ '${input[**it**]}' відсутній в алфавіті!")  
 index  
 **}**}

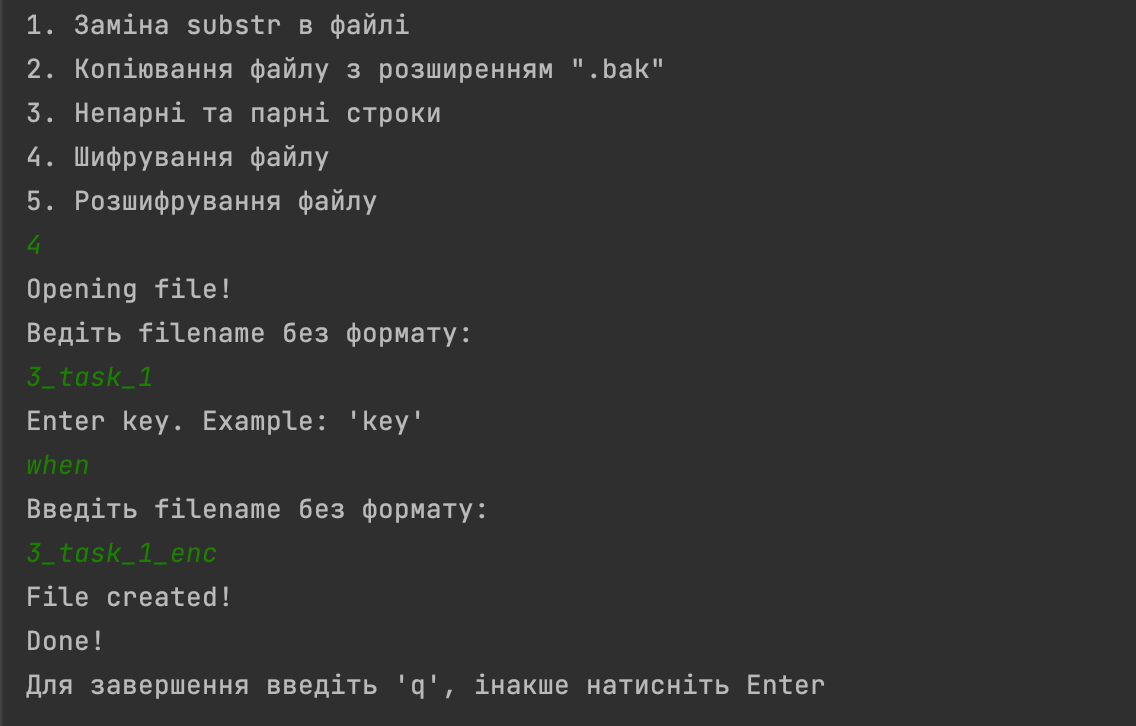
Пункти 4 та 5 виконуються через виклик методу cipher() з класу «Fileworker».

fun cipher() {  
 val file = openFile() ?: return *println*("File does not exist")  
 *println*("Enter key. Example: 'key'")  
 val key = *readLine*() ?: ""  
 val lines = file.*readLines*()  
 val processedLines = when (choose) {  
 1 -> lines.*filter* **{ it**.*isNotEmpty*() **}**.*map* **{** Cipher().encryption(**it**, key) **}** 2 -> lines.*filter* **{ it**.*isNotEmpty*() **}**.*map* **{** Cipher().decryption(**it**, key) **}** else -> lines  
 }  
 File(newFile()).*writeText*(processedLines.*joinToString*("\n"))  
 *println*("Done!")  
}

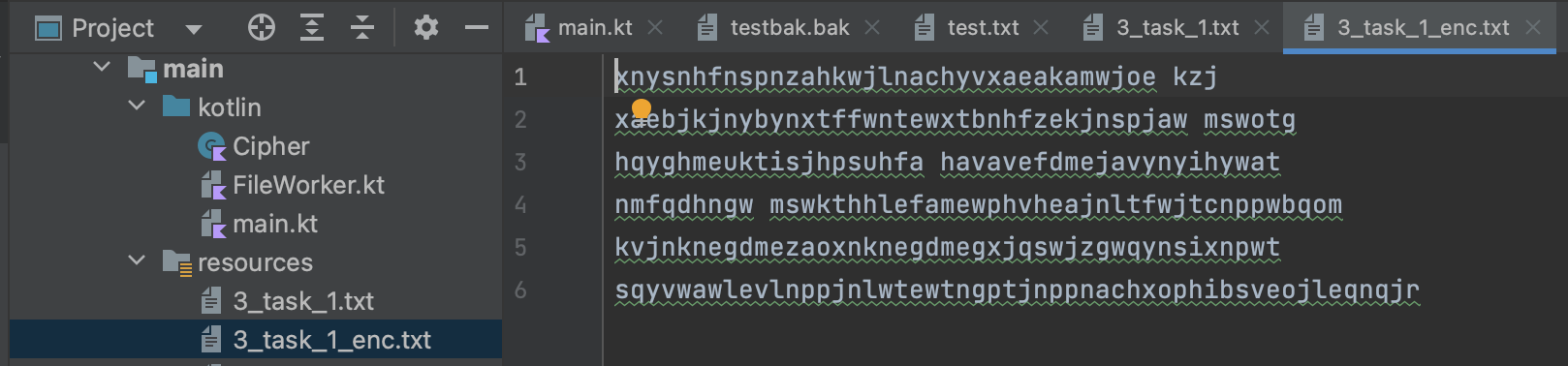
Виконання:

В папці «resourses» було створено текстовий файл із наявною в ньому інформацією:

hello my name is Gleb  
You am from Ukrain  
You learn java and kotlin every day  
You have parents and one sibling brother



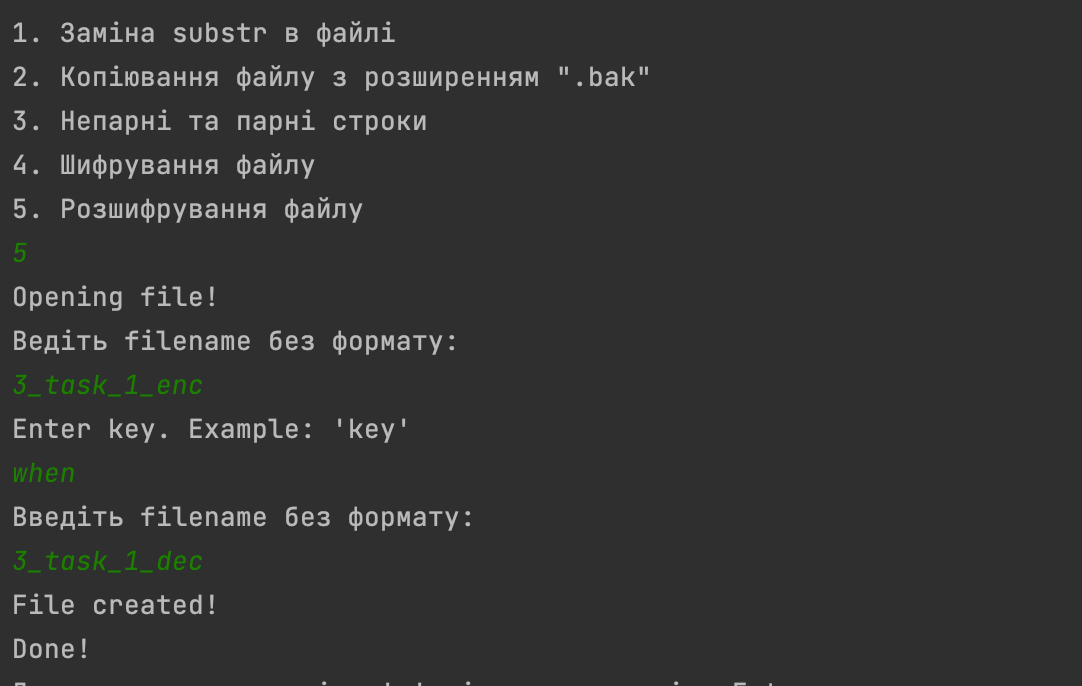
Результат шифрування:

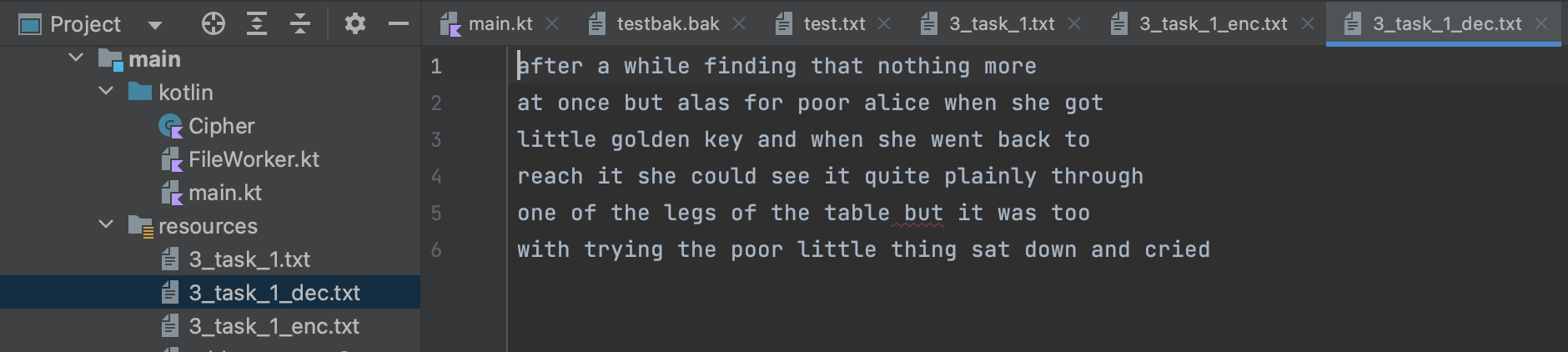


**Завдання 5:** Здійснити розшифрування файлу зашифрованого вище.

fun cipher() {  
 val file = openFile() ?: return *println*("File does not exist")  
 *println*("Enter key. Example: 'key'")  
 val key = *readLine*() ?: ""  
 val lines = file.*readLines*()  
 val processedLines = when (choose) {  
 1 -> lines.*filter* **{ it**.*isNotEmpty*() **}**.*map* **{** Cipher().encryption(**it**, key) **}** 2 -> lines.*filter* **{ it**.*isNotEmpty*() **}**.*map* **{** Cipher().decryption(**it**, key) **}** else -> lines  
 }  
 File(newFile()).*writeText*(processedLines.*joinToString*("\n"))  
 *println*("Done!")  
}

Виконання:





**Висновок**

В результаті виконання домашнього завдання №14 було освоєно основні принципи роботи с мовою Kotlin використано їх на практиці, особливості створення класів та їх об’єктів. Було вдосконалено зання з роботи зі строковими типами даних, файлами та як це використовувати одне з одним. Також було реалізовано шифр Віженера та використано на практиці.