REQUERIMIENTOS

Requerimientos Funcionales:

1. Mostrar un menú con todas las rutas de archivos de texto tanto cifrados como descifrados por el programa.
2. Ordenar las rutas de los archivos según el criterio seleccionado por el usuario (Nombre o fecha).
3. Eliminar rutas de archivos de texto como cifrados como descifrados guardadas por el programa.
4. Personalizar la visualización de los archivos tanto cifrados como descifrados (Tipo de fuente, negrita, color y cursiva).
5. Cifrar archivos de texto usando la técnica de cifrado Mono alfabético Cesar mediante un numero de posiciones (NumberKey) y una dirección (Right o Left).
6. Cifrar archivos de texto usando la técnica de cifrado mono alfabético Atbash.
7. Cifrar archivos de texto usando la técnica de cifrado poli alfabético Vigenere mediante una contraseña (WordKey).
8. Cifrar archivos de texto usando una técnica de cifrado simplificado inspirado en el cifrado simétrico AES haciendo uso de una contraseña de 16 bits.
9. Cifrar texto por consola usando la técnica de cifrado Mono alfabético Cesar mediante un numero de posiciones (NumberKey) y una dirección (Right o Left).
10. Cifrar texto por consola usando la técnica de cifrado mono alfabético Atbash.
11. Cifrar texto por consola usando la técnica de cifrado poli alfabético Vigenere mediante una contraseña (WordKey).
12. Cifrar texto por consola usando una técnica de cifrado simplificado inspirado en el cifrado simétrico AES haciendo uso de una contraseña de 16 bits.
13. Descifrar archivos de texto usando la técnica de cifrado Mono alfabético Cesar.
14. Descifrar archivos de texto usando la técnica de cifrado Mono alfabético Atbash.
15. Descifrar archivos de texto usando la técnica de cifrado Poli alfabético Vigenere.
16. Descifrar archivos de texto usando una técnica de cifrado simplificado inspirado en el cifrado simétrico AES haciendo uso de una contraseña de 16 bits.
17. Descifrar texto por consola usando la técnica de cifrado Mono alfabético Cesar.
18. Descifrar texto por consola usando la técnica de cifrado Mono alfabético Atbash.
19. Descifrar texto por consola usando la técnica de cifrado Poli alfabético Vigenere.
20. Descifrar texto por consola usando una técnica de cifrado simplificado inspirado en el cifrado simétrico AES haciendo uso de una contraseña de 16 bits.

Requerimientos No Funcionales:

1. Leer el archivo de texto a cifrar usando la clase BufferedReader.
2. Leer el archivo de texto a descifrar usando la clase BufferedReader.
3. Guardar el texto cifrado en un archivo txt usando la clase BufferedWritter.
4. Guardar el texto descifrado en un archivo txt usando la clase BufferedWritter.
5. Serializar la clase RouteManager ya que esta se encarga de almacenar los archivos cifrados y descifrados.
6. Almacenar las rutas de los textos cifrados por el programa en una lista doblemente enlazada.
7. Almacenar las rutas de los textos descifrados por el programa en una lista doblemente enlazada.
8. Usar métodos recursivos para buscar objetos en una matriz.
9. Usar métodos de búsqueda binaria para buscar datos char en un String.
10. Usar la interfaz Comparable y Comparator para ordenar la lista enlazadas de rutas usando distintos criterios de orden, como por ejemplo fecha y nombre del archivo.
11. Mostrar las rutas de los textos cifrados y descifrados haciendo uso de TableView y de hilos.
12. Implementar excepciones para los casos:
    1. Cuando el usuario habilita la opción de escribir por consola, pero no escriba nada en ella (ConsoleEmptyException).
    2. Cuando el usuario no configure ningún aspecto del cifrado que quiere realizar a un texto (EmptyFieldsException).
    3. Cuando el usuario especifique una contraseña más larga que el propio mensaje, en el cifrado Vigenere (WordKeyInvalidException).
    4. Cuando el usuario escribe una ruta que no pertenece a un archivo de texto (FileRouteExcepcion).
    5. Cuando el usuario escriba una contraseña que tiene menos de 16 bits o excede este límite (Invalid16bitsKeyException).
    6. Cuando el usuario escriba una palabra en la contraseña numérica de Cesar (NumberFormatException)

Nota: Los requerimientos implementados serán cifrar y descifrar con Cesar mediante Consola.

PRUEBAS

Pruebas Cesar

Configuración de los Escenarios

Nombre	Clase	Escenario
setup1	CesarTest	Un objeto de la clase Cesar con track= 1 y direction= “R”
setup2	CesarTest	Un objeto de la clase Cesar con track= 1 y direction= “L”
setup3	CesarTest	Un objeto de la clase Cesar con track= 54 y direction= “R”
setup4	CesarTest	Un objeto de la clase Cesar con track= 54 y direction= “L”
setup5	CesarTest	Un objeto de la clase Cesar con track= 100 y direction= “R”
Setup6	CesarTest	Un objeto de la clase Cesar con track= 100 y direction= “L”

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método encrypt de la clase Cesar funcione correctamente, encriptando un texto según el track y la direction dada.

Clase	Método	Escenario	Valores de Entrada	Resultado
Cesar	encrypt	setup1	txt = “a”	El mensaje encriptado es “b”
Cesar	encrypt	setup1	txt = “z”	El mensaje encriptado es “A”
Cesar	encrypt	setup1	txt = “Z”	El mensaje encriptado es “a”
Cesar	encrypt	setup1	txt = “hola como estas”	El mensaje encriptado es “ipmb dpnp ftubt”
Cesar	encrypt	setup1	txt = “Hola, ¿como estas?”	El mensaje encriptado es “ipmb, ¿dpnp ftubt?”
Cesar	encrypt	setup2	txt = “a”	El mensaje encriptado es “Z”
Cesar	encrypt	setup2	txt = “z”	El mensaje encriptado es “y”
Cesar	encrypt	setup2	txt = “Z”	El mensaje encriptado es “Y”
Cesar	encrypt	setup2	txt = “hola como estas”	El mensaje encriptado es “gñkZ bñlñ drsZr”
Cesar	encrypt	setup2	txt = “Hola, ¿como estas?”	El mensaje encriptado es “GñkZ, ¿bñlñ drsZr?”
Cesar	encrypt	setup3	txt = “a”	El mensaje encriptado es “a”
Cesar	encrypt	setup3	txt = “z”	El mensaje encriptado es “z”
Cesar	encrypt	setup3	txt = “Z”	El mensaje encriptado es “Z”
Cesar	encrypt	setup3	txt = “hola como estas”	El mensaje encriptado es “hola como estas”
Cesar	encrypt	setup3	txt = “Hola, ¿como estas?”	El mensaje encriptado es “Hola, ¿como estas?”
Cesar	encrypt	setup4	txt = “a”	El mensaje encriptado es “a”
Cesar	encrypt	setup4	txt = “z”	El mensaje encriptado es “z”
Cesar	encrypt	setup4	txt = “Z”	El mensaje encriptado es “Z”
Cesar	encrypt	setup4	txt = “hola como estas”	El mensaje encriptado es “hola como estas”

Cesar	encrypt	setup4	txt = “Hola, ¿como estas?”	El mensaje encriptado es “Hola, ¿como estas?”
Cesar	encrypt	setup5	txt = “a”	El mensaje encriptado es “S”
Cesar	encrypt	setup5	txt = “z”	El mensaje encriptado es “r”
Cesar	encrypt	setup5	txt = “Z”	El mensaje encriptado es “R”
Cesar	encrypt	setup5	txt = “hola como estas”	El mensaje encriptado es “ZhdS Uheh WlmSl”
Cesar	encrypt	setup5	txt = “Hola, ¿como estas?”	El mensaje encriptado es “zhdS, ¿Uheh WlmSl?”
Cesar	encrypt	setup6	txt = “a”	El mensaje encriptado es “i”
Cesar	encrypt	setup6	txt = “z”	El mensaje encriptado es “H”
Cesar	encrypt	setup6	txt = “Z”	El mensaje encriptado es “h”
Cesar	encrypt	setup6	txt = “hola como estas”	El mensaje encriptado es “owsi kwtw mABiA”
Cesar	encrypt	setup6	txt = “Hola, ¿como estas?”	El mensaje encriptado es “Owsi, ¿kwtw mABiA?”

Objetivo de la Prueba: Verificar que el método decrypt de la clase Cesar funcione correctamente, desencriptando un texto según el track y la direction dada.

Clase	Método	Escenario	Valores de Entrada	Resultado
Cesar	decrypt	setup1	txt = “b”	El mensaje desencriptado es “a”
Cesar	decrypt	setup1	txt = “A”	El mensaje desencriptado es “z”
Cesar	decrypt	setup1	txt = “a”	El mensaje desencriptado es “Z”
Cesar	decrypt	setup1	txt = “ipmb dpnp ftubt”	El mensaje desencriptado es “hola como estas”
Cesar	decrypt	setup1	txt = “ipmb, ¿dpnp ftubt?””	El mensaje desencriptado es “Hola, ¿como estas?”
Cesar	decrypt	setup2	txt = “Z”	El mensaje desencriptado es “a”
Cesar	decrypt	setup2	txt = “y”	El mensaje desencriptado es “z”
Cesar	decrypt	setup2	txt = “Y”	El mensaje desencriptado es “Z”
Cesar	decrypt	setup2	txt = “gñkZ bñlñ drsZr”	El mensaje desencriptado es “hola como estas”
Cesar	decrypt	setup2	txt = “GñkZ, ¿bñlñ drsZr?”	El mensaje desencriptado es “Hola, ¿como estas?””
Cesar	decrypt	setup3	txt = “a”	El mensaje desencriptado es “a”
Cesar	decrypt	setup3	txt = “z”	El mensaje desencriptado es “z”
Cesar	decrypt	setup3	txt = “Z”	El mensaje desencriptado es “Z”
Cesar	decrypt	setup3	txt = “hola como estas”	El mensaje desencriptado es “hola como estas”
Cesar	decrypt	setup3	txt = “Hola, ¿como estas?””	El mensaje desencriptado es “Hola, ¿como estas?””
Cesar	decrypt	setup4	txt = “a”	El mensaje desencriptado es “a”
Cesar	decrypt	setup4	txt = “z”	El mensaje desencriptado es “z”
Cesar	decrypt	setup4	txt = “Z”	El mensaje desencriptado es “Z”

Cesar	decrypt	setup4	txt = “hola como estas”	El mensaje desencriptado es “hola como estas”
Cesar	decrypt	setup4	txt = “Hola, ¿como estas?””	El mensaje desencriptado es “Hola, ¿como estas?””
Cesar	decrypt	setup5	txt = “S”	El mensaje desencriptado es “a”
Cesar	decrypt	setup5	txt = “r”	El mensaje desencriptado es “z”
Cesar	decrypt	setup5	txt = “R”	El mensaje desencriptado es “Z”
Cesar	decrypt	setup5	txt = “ZhdS Uheh WlmSl”	El mensaje desencriptado es “hola como estas”
Cesar	decrypt	setup5	txt = “zhdS, ¿Uheh WlmSl”	El mensaje desencriptado es “Hola, ¿como estas?””
Cesar	decrypt	setup6	txt = “i”	El mensaje desencriptado es “a”
Cesar	decrypt	setup6	txt = “H”	El mensaje desencriptado es “z”
Cesar	decrypt	setup6	txt = “h”	El mensaje desencriptado es “Z”
Cesar	txt = “i”	setup6	txt = “owsi kwtw mABiA”	El mensaje desencriptado es “hola como estas”
Cesar	decrypt	setup6	txt = “Owsi, ¿kwtw mABiA?”	El mensaje desencriptado es “Hola, ¿como estas?””

Pruebas Atbash

Configuración de los Escenarios

Nombre	Clase	Escenario
setup1	AtbashTest	Un objeto de la clase Atbash

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método encrypt de la clase Cesar funcione correctamente, encriptando un texto según el track y la direction dada.

Clase	Método	Escenario	Valores de Entrada	Resultado
Atbash	encrypt	setup1	txt = “a”	El mensaje encriptado es “z”
Atbash	encrypt	setup1	txt = “n”	El mensaje encriptado es “n”
Atbash	encrypt	setup1	txt = “A”	El mensaje encriptado es “Z”
Atbash	encrypt	setup1	txt = “algoritmos”	El mensaje encriptado es “zotlirgñlh”
Atbash	encrypt	setup1	txt = “¡Programacion!”	El mensaje encriptado es “¡Kiltizñzxrln!”

Objetivo de la Prueba: Verificar que el método decrypt de la clase Cesar funcione correctamente, desencriptando un texto según el track y la direction dada.

Clase	Método	Escenario	Valores de Entrada	Resultado
Atbash	decrypt	setup1	txt = “z”	El mensaje desencriptado es “a”
Atbash	decrypt	setup1	txt = “n”	El mensaje desencriptado es “n”
Atbash	decrypt	setup1	txt = “Z”	El mensaje desencriptado es “A”
Atbash	decrypt	setup1	txt = “zotlirgñlh”	El mensaje desencriptado es “algoritmos”
Atbash	decrypt	setup1	txt = “¡Kiltizñzxrln!”	El mensaje desencriptado es “¡Programacion!”

Pruebas Vigenere

Configuración de los Escenarios

Nombre	Clase	Escenario
setup1	VigenereTest	Un objeto de la clase Vigenere con key= “louis”
setup2	VigenereTest	Un objeto de la clase Vigenere con key= “fishclown”
setup3	VigenereTest	Un objeto de la clase Vigenere con key= “redhair”
setup4	VigenereTest	Un objeto de la clase Vigenere con key= “reggaeton”

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método encrypt de la clase Vigenere funcione correctamente, encriptando según la key dada.

Clase	Método	Escenario	Valores de Entrada	Resultado
Vigenere	encrypt	setup1	message = “niño”	El mensaje encriptado es “ywxi”
Vigenere	encrypt	setup2	message = “valentina”	El mensaje encriptado es “aidlpewjn”
Vigenere	encrypt	setup3	message = “alejandro”	El mensaje encriptado es “rphqavuis”
Vigenere	encrypt	setup4	message = “jose luis”	El mensaje encriptado es “asyk lybg”

Objetivo de la Prueba: Verificar que el método decrypt de la clase Vigenere funcione correctamente, desencriptando según la key dada.

Clase	Método	Escenario	Valores de Entrada	Resultado
Vigenere	decrypt	setup4	message = “asyk lybg”	El mensaje encriptado es “jose luis”
Vigenere	decrypt	setup3	message = “rphqavuis”	El mensaje encriptado es “alejandro”
Vigenere	decrypt	setup2	message = “aidlpewjn”	El mensaje encriptado es “valentina”

Objetivo de la Prueba: Verificar que el método recursiveSearchFileEncriptedLetter de la clase Vigenere funcione correctamente, buscando la posición de una letra según la fila de la key dada.

Clase	Método	Escenario	Valores de Entrada	Resultado
Vigenere	recursiveSearchFileEncriptedLetter	setup1	fileKey = 10 currentColumn = 1 currentChar = ‘n’	La posición de la letra es 5
Vigenere	recursiveSearchFileEncriptedLetter	setup1	fileKey = 23 currentColumn = 1 currentChar = ‘y’	La posición de la letra es 3
Vigenere	recursiveSearchFileEncriptedLetter	setup1	fileKey = 23 currentColumn = 1 currentChar = ‘a’	La posición de la letra es 5
Vigenere	recursiveSearchFileEncriptedLetter	setup1	fileKey = 26 currentColumn = 1 currentChar = ‘y’	La posición de la letra es 26

Vigenere	recursiveSearchFileEncriptedLetter	setup1	fileKey = 11 currentColumn = 1 currentChar = ‘w’	La posición de la letra es 13

Objetivo de la Prueba: Verificar que el método recursiveSearchPlaneLetter de la clase Vigenere funcione correctamente, buscando la posición de una letra del texto a encriptar en la primera fila de la matriz				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Vigenere	recursiveSearchPlaneLetter	setup1	currentColumn = 0 currentChar = ‘a’	La posición de la letra es 1
Vigenere	recursiveSearchPlaneLetter	setup1	currentColumn = 0 currentChar = ‘m’	La posición de la letra es 13
Vigenere	recursiveSearchPlaneLetter	setup1	currentColumn = 0 currentChar = ‘l’	La posición de la letra es 12
Vigenere	recursiveSearchPlaneLetter	setup1	currentColumn = 0 currentChar = ‘z’	La posición de la letra es 26

Objetivo de la Prueba: Verificar que el método recursiveSearchKeyLetter de la clase Vigenere funcione correctamente, buscando la posición de una letra de la Key en la primera columna de la matriz				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Vigenere	recursiveSearchKeyLetter	setup1	currentRow = 0 currentChar = ‘g’	La posición de la letra es 7
Vigenere	recursiveSearchKeyLetter	setup1	currentRow = 0 currentChar = ‘j’	La posición de la letra es 10
Vigenere	recursiveSearchKeyLetter	setup1	currentRow = 0 currentChar = ‘p’	La posición de la letra es 16
Vigenere	recursiveSearchKeyLetter	setup1	currentRow = 0 currentChar = ‘z’	La posición de la letra es 26

Objetivo de la Prueba: Verificar que el método getVigenereMatrix de la clase Vigenere funcione correctamente, buscando una letra según la posición dada en la matriz				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Vigenere	getVigenereMatrix	setup1	column = 1 file = 1	La letra en la posición es a
Vigenere	getVigenereMatrix	setup1	column = 7 file = 6	La letra en la posición es l
Vigenere	getVigenereMatrix	setup1	column = 15 file = 17	La letra en la posición es e

Objetivo de la Prueba: Verificar que el método filled de la clase Vigenere funcione correctamente, confirmando que se ha llenado bien la matriz dándole una posición				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Vigenere	filled	setup1	column = 4 file = 4	La letra en la posición es g
Vigenere	filled	setup1	column = 7 file = 4	La letra en la posición es j
Vigenere	filled	setup1	column = 6 file = 6	La letra en la posición es k

Pruebas AES

//Pendiente

Configuración de los Escenarios

Nombre	Clase	Escenario
setup1	AESTest	Un objeto de la clase AES con encriptionKey = “9B,FE,DE,BC,FE,DE,EF,86,EE,8A,B8,CC,DC,B9,81,F2” txt = “Píldora Thoth 30”

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método subBytes de la clase AES funcione correctamente, haciendo la debida sustitución en la matriz de estado				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
AES	subBytes	setup1	state = {{50,ED,6C,64},{6F,74,68,20} ,{54,68,6F,74},{68,20,33,30}}	La nueva matriz de state es {{53,55,50,43},{A8,92,45,B7},{20,45,A8,92}, {45,B7,C3,04}}

Objetivo de la Prueba: Verificar que el método shiftRows de la clase AES funcione correctamente, rotando cada fila de la matriz state según los bytes correspondientes.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
AES	shiftRows	setup1	state = {{53,55,50,43},{A8,92,45,B7}, {20,45,A8,92},{45,B7,C3,04}}	La nueva matriz state es {{53,55,50,43},{92,45,B7,A8},{A8,92,20,45}, {04,45,B7,C3}}

Objetivo de la Prueba: Verificar que el método mixColums de la clase AES funcione correctamente, multiplicando la matriz constant por la matriz state				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
AES	mixColums	setup1	state = {{53,55,50,43},{92,45,B7,A8}, {A8,92,20,45},{04,45,B7,C3}}	La nueva matriz state es {{,,,},{,,,},{,,,},{,,,}}

Objetivo de la Prueba: Verificar que el método addRoundKey de la clase AES funcione correctamente, multiplicando de forma XOR la matriz state por la key				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
AES	addRoundKey	setup1	state = {{,,,},{,,,},{,,,},{,,,}}	El mensaje encriptado es {{,,,},{,,,},{,,,},{,,,}}

Objetivo de la Prueba: Verificar que el método encrypt de la clase AES funcione correctamente, creando la matriz state y recibiéndole a addRoundKey un String con elmensaje encriptado				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
AES	shiftRows	setup1	txt = “Píldora Thoth 30”	El mensaje ecriptado es {}

Objetivo de la Prueba: Verificar que el método decrypt de la clase AES funcione correctamente,				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
AES	shiftRows	setup1	state = {}	El mensaje desencriptado es “”

Pruebas RouteManager

Configuración de los Escenarios

Nombre	Clase	Escenario
setup1	RouteMaganer	Crea un objeto de la clase RouteManager con un RouteList con listas enlazadas con elementos.
setUp2	RouteMaganer	Crea un objeto de la clase RouteManager con un objeto RouteList sin listas enlazadas (Vacias)
setUp3	RouteMaganer	Crea un objeto de la clase RouteList con una lista Enlazada EncryptedRoute
setUp4	RouteMaganer	Crea un objeto de la clase RouteList con una lista Enlazada DecryptedRoute

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método getEncryptedRoutes de la clase RouteMaganer funcione correctamente, según las listas enlazadas creadas en cada setUp;

Clase	Método	Escenario	Valores de Entrada	Resultado
RouteMaganer	getEncryptedRoutes	setUp2	Ninguno	El método no retorna nada ya que las listas están vacias
RouteMaganer	getEncryptedRoutes	setUp1	Ninguno	El método retorna un objeto de EncryptedRoute
RouteMaganer	getEncryptedRoutes	setUp4	Ninguno	El método no retorna nada ya que la lista EncryptedRoute se encuentra vacia
RouteMaganer	getEncryptedRoutes	setUp3	Ninguno	El método retorna un objeto de EncryptedRoute

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método getDecryptedRoutes de la clase RouteMaganer funcione correctamente, según las listas enlazadas creadas en cada setUp;

Clase	Método	Escenario	Valores de Entrada	Resultado
RouteMaganer	getDecryptedRoutes	setUp1	Ninguno	El método no retorna nada ya que las listas están vacias
RouteMaganer	getDecryptedRoutes	setUp1	Ninguno	El método retorna un objeto de EncryptedRoute
RouteMaganer	getDecryptedRoutes	setUp4	Ninguno	El método no retorna nada ya que la lista DecryptedRoute se encuentra vacia
RouteMaganer	getDecryptedRoutes	setUp4	Ninguno	El método retorna un objeto de DecryptedRoute

Objetivo de la Prueba: Verificar que el método addEncryptedRoute de la clase RouteMaganer funcione correctamente, según las listas enlazadas creadas en cada setUp;

Clase	Método	Escenario	Valores de Entrada	Resultado
RouteMaganer	addEncryptedRoute	setUp2	Route=" Crypter-/data/RouteListTest.txt"	El método envia exitosamente el parametro para que la clase RouteList pueda crear una lista enlazada de EncryptedRoute
RouteMaganer	addEncryptedRoute	setUp1	Route=" Crypter-/data/RouteListTest.txt"	El método añade una nueva lista enlazada.
RouteMaganer	addEncryptedRoute	setUp3	Route=" Crypter-/data/RandomEncryptRouteListTest.txt"	El metodo añade en orden un nodo de EncryptedRoute

Objetivo de la Prueba: Verificar que el método addDecryptedRoute de la clase RouteMaganer funcione correctamente, según las listas enlazadas creadas en cada setUp;

Clase	Método	Escenario	Valores de Entrada	Resultado
RouteMaganer	addDecryptedRoute	Setup2	Route=" Crypter-/data/RouteListTest.txt"	El método envia exitosamente el parametro para que la clase RouteList pueda crear una lista enlazada de DecryptedRoute
RouteMaganer	addDecryptedRoute	Setup1	Route=" Crypter-/data/RouteListTest.txt"	El método añade una nueva lista enlazada.
RouteMaganer	addDecryptedRoute	setUp4	Route=" Crypter-/data/RandomDecryptedRouteListTest.txt"	El metodo añade en orden un nodo de DecryptedRoute

Pruebas RouteList

Configuración de los Escenarios

Nombre	Clase	Escenario
setup1	RouteList	Crea un objeto de la clase RouteList con listas vacias.
setup2	RouteList	Crea un objeto de la clase RouteList con una lista enlazada EncryptedRoute
setup3	RouteList	Crea un objeto de la clase RouteList con una lista Enlazada DecryptedRoute

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método addEncryptedRoutes de la clase RouteList funcione correctamente, según las listas creadas en cada caso.

Clase	Método	Escenario	Valores de Entrada	Resultado
RouteList	addEncryptedRoutes	Setup1	Route=" Crypter-/data/RouteListTest.txt"	El método crea un nodo EncryptedRoute con la route pasada como parámetro.
RouteList	addEncryptedRoutes	Setup2	Route=" Crypter-/data/RouteListTest.txt"	El método añade en el orden correcto el nodo con la route pasada como parámetro
RouteList	addEncryptedRoutes	SetUp1	Route="121423asdas_+!@ "	El metodo no añade esa ruta a un nodo EncryptedRoute debido a que es una ruta invalidad

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método addDecryptedRoutes de la clase RouteList funcione correctamente, según las listas creadas en cada caso.

Clase	Método	Escenario	Valores de Entrada	Resultado
RouteList	addDecryptRoutes	setUp1	Route=" ";	El método no crea un nodo DecryptedRoute debido a que esa ruta es una ruta invalida
RouteList	addDecryptedRoutes	Setup1	Route=" Crypter-/data/RouteListTest.txt"	El método crea un nodo DecryptedRoute con la route pasada como parámetro.
RouteList	addDecryptedRoutes	Setup2	Route=" Crypter-/data/RouteListTest.txt"	El método añade en el orden correcto el nodo con la route pasada como parámetro

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método deleteDecryptedRoute de la clase RouteList funcione correctamente, eliminando un nodo según el parámetro.

Clase	Método	Escenario	Valores de Entrada	Resultado
RouteList	deleteDecryptedRoute	Setup1	Route=" Crypter-/data/RouteListTest.txt"	El método no hace nada ya que no tiene ningún nodo.
RouteList	deleteDecryptedRout	Setup3	Route=" Crypter-/data/RouteListTest.txt"	El método elimina un nodo exitosamente.
RouteList	deleteDecryptedRoute	SetUp3	Route=" "	El metodo no hace nada ya que no tiene ningun nodo y esa ruta es invalida

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método deleteDecryptedRoute de la clase RouteList funcione correctamente, eliminando un nodo según el parámetro.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
RouteList	deleteEncryptedRoute	Setup1	Route=” Crypter-/data/RouteListTest.txt”	El método no hace nada ya que no tiene ningún nodo.
RouteList	deleteEncryptedRout	Setup2	Route=” Crypter-/data/RouteListTest.txt”	El método elimina un nodo exitosamente.
RouteList	deleteEncryptedRoute	SetUp2	Route=” ”	El metodo no hace nada ya que no tiene ningun nodo y esa ruta es invalida

Pruebas RouteNodo

Configuración de los Escenarios

Nombre	Clase	Escenario
setup1	RouteNode	Un objeto de la clase RouteNode con un route= “Crypter-/data/FirstRouteListTest.txt” y un objeto LocalDateTime con la fecha “02/05/2020”

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar que el método setEncryptedRoute de la clase RouteNode funcione correctamente, detectando rutas validas e invalidas para así poder añadirlas al siguiente nodo.

Clase	Método	Escenario	Valores de Entrada	Resultado
RouteNode	setEncryptedRoute	setUp1	Route=” “;	El método no crea ningún nodo nuevo porque esa ruta no existe. Este es un caso extremo debido a que, este tipo de rutas no serán permitidas en el programa.
RouteNode	setEncryptedRoute	setUp1	Route=” Crypter-/data/RouteListTest.txt”	El método crea y asigna un nuevo nodo a la lista enlazada
RouteNode	setEncryptedRoute	setUp1	Route=” Crypter-/data/NoExistRouteListTest.txt”	El método verifica que el archivo no existe y no lo añade a la lista. Este es otro caso extremo debido a que la no existencia de una ruta se comprueba desde la GUI

Objetivo de la Prueba: Verificar que el método getNextEncryptedRoute de la clase RouteNode funcione correctamente, retornando las rutas que han sido añadidas al nodo.

Clase	Método	Escenario	Valores de Entrada	Resultado
RouteNode	getNextEncryptedRoute	setUp1	Ninguno	El metodo no retorna nada debido a que no hay ningún nodo creado.
RouteNode	getNextEncryptedRoute	setUp1	Ninguno	El método retorna el nodo next.
RouteNode	getEncryptedRoutes	Setup1	Ninguno	El método no retorna nada debido a que no agrega rutas que no tengan archivos existentes

src

ui

CrypterGUI

-borderPaneMain: BorderPane
-settingsMenu: Menu
-subMenu: Menu
-textInputToggle: ToggleGroup
-methodToggle: ToggleGroup
-encryptByFile: Button
-decryptByFile: Button
-borderSelected: AnchorPane
-aesImage:ImageView
-vigenereImage: ImageView
-cesarImage: ImageView
-atbashImage: ImageView
-searchFile: Button
-fileRoute: TextField
-goToConsole: Button
-decryptionTitle: ImageView
-encryptionTitle: ImageView
-encryptedListViewRM: ListView<String>
-decryptedListViewRM: ListView<String>
-R: RadioButton
-directionToggleCesar: ToggleGroup
-L: RadioButton
-numberKeyCesar: TextField
-wordKeyVigenere_AES: TextField
-textConsole: TextArea
-encryptByConsole: Button
-decryptByConsole: Button
-textFromText: TextArea
-fontChoice: ChoiceBox<String>
-heightChoice: ChoiceBox<String>

loadEncryptWindow():void
load(route: String):void
loadEncryptWindow():void
loadDecryptWindow():void
loadRouteManager():void
loadMenu():void
subLoad(route: String):void
subLoadCesar():void
subLoadAtbash():void
subLoadVigenereAES():void
setInputSettings():void
setTypeSettings():void
aesM(event: ActionEvent):void
atbashM(event: ActionEvent):void
cesarM(event: ActionEvent):void
consoleTO(event: ActionEvent):void
decryptWelcome(event: ActionEvent):void
encryptWelcome(event: ActionEvent):void
routeManagerWelcome(event: ActionEvent):void
textFileTO(event: ActionEvent):void
vigenereM(event: ActionEvent):void
back(event: ActionEvent):void
encryptByFile(event: ActionEvent):void
decryptByFile(event: ActionEvent):void
goToConsole(event: ActionEvent):void
backRM(event: ActionEvent):void
deleteFileRM(event: ActionEvent):void
showFileRM(event: ActionEvent):void
decryptMenu(event: ActionEvent):void
encryptMenu(event: ActionEvent):void
routeManagerMenu(event: ActionEvent):void
exit(event: ActionEvent):void
encryptByConsole(event: ActionEvent):void
decryptByConsole(event: ActionEvent):void
cursiveOption(event: ActionEvent):void
negritaOption(event: ActionEvent):void
subOption(event: ActionEvent):void
loadEncryptWindow():void
load(route: String):void
loadEncryptWindow():void
loadDecryptWindow():void
loadRouteManager():void
loadMenu():void
subLoad(route: String):void
subLoadCesar():void
subLoadAtbash():void
subLoadVigenereAES():void
setInputSettings():void
setTypeSettings():void

Main
+start() : void

1 crypter

AtbashWindow
CesarWindow.fxml
EncryptDecryptWindow.fxml
Console.fxml
Menu.fxml
ShowFile.fxl
Vigenere-AES-Window.fx
RouteManager.fxml
Welcome.fxml

exceptions

<<Exception>> EmptyFieldsException
-emptyFields : String
+EmptyFieldsException(emptyFields : String) : void

<<Exception>> ConsoleEmptyException
+final EXAMPLEMESSAGE : String = "The treasure's...
+ConsoleEmptyException() : void

<<Exception>> Invalid16BitsKeyException
-wrongKey : String
-final EXAMPLEKEY : String = "thisisagoodkey!!"
+Invalid16BitsKeyException(wrongKey : String) : void

<<Exception>> WordKeyInvalidException
-wrongKey : String
-ExamplesKey : String[][]
+WordKeyInvalidException(wrongKey : String) : void
+getExample() : String

<<Exception>> FileRouteException
-wrongRoute : String
+FileRouteException(wrongRoute : String) : void

0..1 crypter

1 routeManager

model

RoutesThread
-encryptedList : EncryptedRouteList
-decryptedList : DecryptedRouteList
-encryptedArrayList : ArrayList <EncryptedRoute>
-decryptedArrayList : ArrayList <DecryptedRoute>

<<Interface>> Serializable

RouteMaganer
+RouteManager() : void
+getEncryptedRouteList() : ArrayList <String>
+getDecryptedRouteList() : ArrayList <String>
+searchRoute(route : String) : String
+addEncryptedRoute(route : String) : void
+addDecryptedRoute(route : String) : void

RoutesList
+DecryptedRoutesList () : void
+addDecryptedRoute(decryptedRoute : String) : void
+deleteDecryptedRoute(decryptedRoute : String) : String
+addEncryptedRoute(encryptedRoute : String) : void
+deleteEncryptedRoute(encryptedRoute : String) : String

0..1 firstRoute
1 last
1 next

RoutNode
-route : String
-date : LocalDateTime
+RoutNode(route : String, date : LocalDateTime, next : RoutNode, last : RoutNode) : void
+getNext() : RoutNode
+getLast() : RoutNode

sortByNameAndDateComparator
+compare(re1 : EncrypteRoute, re2 : Encrypt...
+compare(re1 : RoutNode, re2 : RoutNode)

<<Interface>> Comparator <EncryptedRoute>
+compare(re1 : EncryptedRoute...

<<Interface>> Comparator <DecryptedRoute>
+compare(re1 : RouteNodeTest...

<<Abstract>> Crypter
-decrypt(txt : String) : String
-encrypt(txt : String) : String

<<Abstract>> Monoalphabetic
#ABC : String = abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ
+encrypt(txt : String) : String
+decrypt(txt : String) : String

<<Abstract>> Polyalphabetic
#ABC : String = abcdefghijklmnñopqrstuvwxyz
#keyWord : String
+encrypt(txt : String) : String
+decrypt(txt : String) : String
+getEncriptionKey() : String

Cesar
-direction : String
-track : int
+Cesar(direction : String, track : int) : void
+leftMove(txt : String) : String
+leftMove(txt : String, trackParameter : int) : String
+rightMove(txt : String) : String
+rigtMove(txt : String, trackParameter : int) : String
+calculateExtends() : String
+searchNumber(txt : String) : int
+decrypt(txt : String) : String
+encrypt(txt : String) : String

Atbash
+Atbash() : void
+searchLetter() : int
+encrypt(txt : String) : String
+decrypt(txt : String) : String

Vigenere
-vigenereMatrix : char[][]
+Vigenere(keyWord : String) : void
+recursiveSearchFileEncriptedLetter(fileKey : int, currentColumn : int, currentChar : int) : int
+recursiveSearchKeyLetter(currentRow : int, currentChar : char) : int
+recursiveSearchPlaneLetter(currentColumn : int, currentChar : int) : int
+getVigenereMatrix(column : int, file : int) : char
+encrypt(txt : String) : String
+decrypt(txt : String) : String
+filled() : void

AES
-final SBOX : String[][]
-keyArray : String[][]
-final CONSTANTARRAY : int[][]
-stateMatrix : String[][]
+AES(keyWord : String, txt : String) : void
+addRoundKey(String[][]) : String[][]
+subBytes(state : String[][]) : String[][]
+shiftRows(state : String[][]) : String[][]
+mixColumns(state : [][]) : String[][]

routeManager routeList node cesar atbash vigenere aes

Model2

RouteManagerTest
+setUp1() : void
+setUp2() : void
+setUp3() : void
+setUp4() : void
+addEncryptedRouteTest() : void
+addDecryptedRouteTest() : void
+getEncryptedRouteListTest() : void
+getDecryptedRouteListTest() : void

RoutesListTest
+setUp1() : void
+setUp2() : void
+setUp3() : void
+addEncryptedRoute() : void
+addDecryptedRoute() : void
+deleteDecryptedRoute() : void
+deleteEncryptedRoute() : void

RouteNodeTest
+setUp1() : void
+getNextDecryptedRoute() : void
+setNextDecryptedRoute() : void

CesarTest
+setUp1() : void
+setUp2() : void
+setUp3() : void
+setUp4() : void
+setUp6() : void
+setUp5() : void
+encryptTest() : void
+decryptTest() : void

AtbashTest
+setUp1() : void
+encryptTest() : void
+decryptTest() : void
+searchLetterTest() : void

VigenereTest
+setUp1() : void
+setUp2() : void
+setup3() : void
+setUp4() : void
+encryptTest() : void
+decryptTest() : void
+fillesTest() : void
+recursiveSearchFileEncriptedLetterTest() : void
+recursiveSearchKeyLetterTest() : void
+recursiveSearchPlaneLetterTest() : void
+getVigenereMatrixTest() : void

AESTest
+setUp1() : void
+setUp2() : void
+addRoundKeyTest() : void
+subBytesTest() : void
+shiftRowsTest() : void
+mixColumnsTest() : void