LAB #21
File I/O & CRYPTOGRAPHY

Ted Pawlicki

December 10 – December 6, 2004

This lab should be completed during the scheduled lab sessions. In order to complete this lab you must demonstrate programs to your lab TA. The technical knowledge on how to accomplish these tasks is found in the assigned reading or the lecture notes, but the lab TA will help as well.

1. Write a JAVA application that reads and writes files while encrypting and decrypting text.

BACKGROUND:

The simplest kind of encryption is the shift cipher, also known as the Caesar cipher. In Caesar's case to encrypt each letter in a message would be moved forward by a number (referred to as the key). So, if the key were 2, 'a' would become 'c', 'b' would become 'd', etc. The cipher is said to wrap around from the end – 'y' goes to 'a', 'z' goes to 'b'.

A better encryption algorithm is a *Random Monoalphabetic Cipher*. Instead of using a number for the key, use a word. Suppose the key word is FEATHER. First remove all the duplicate letters, yielding FEATHR, and append the other letters of the alphabet in reverse order. Then encrypt/decrypt according to the mapping:

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
F	E	A	T	H	R	Z	Y	X	W	V	U	S	Q	P	O	N	M	L	K	J	I	G	D	C	B

Write a JAVA method to implement a Random Monoalphabetic Cipher. Two strings are parameters – the message, and the key word. Using the same data techniques you used in the Caesar Cipher. Only now you need to have two data strings.

Your program should operate from the command line. The program takes the following command line arguments:

- An optional –d flag to indicate decryption instead of encryption
- A keyword key used to generate the mapping
- The input file name

· The output file name

For example

java Crypt –d –FEATHER encrypt.txt output.txt

decrypts a file using the keyword FEATHER. It is an error not to supply a keyword.

Start by writing an application that deals with the command line arguments. Remember to do the error checking.

Write a method that simply copies the input file to the output file, character by character

a. For input the java.io.FileReader class with the associated read() method will suffice.

b. For output use the java.io.FileWriter class and the associated write(char) methods.

Write a method that generates the encryption mapping using string manipulation.

Write a method that encrypts/decrypts a single character.

Insert a call to the character encrypter/decrypter into your "file copy" method.

Your hand in should include your code and a copy of your encrypted and decrypted file.

Cut and paste your source code and screen shots of running code into your lab document and hand it in. Be sure to demo your running code to your TA so that the TA has a record of "checking you off" as having done this lab.

NOTE: Electronic hand in should be working at this point. Submit your lab work through the homework drop box in WebCT. However, if you encounter any problems, revert to paper based hand in.