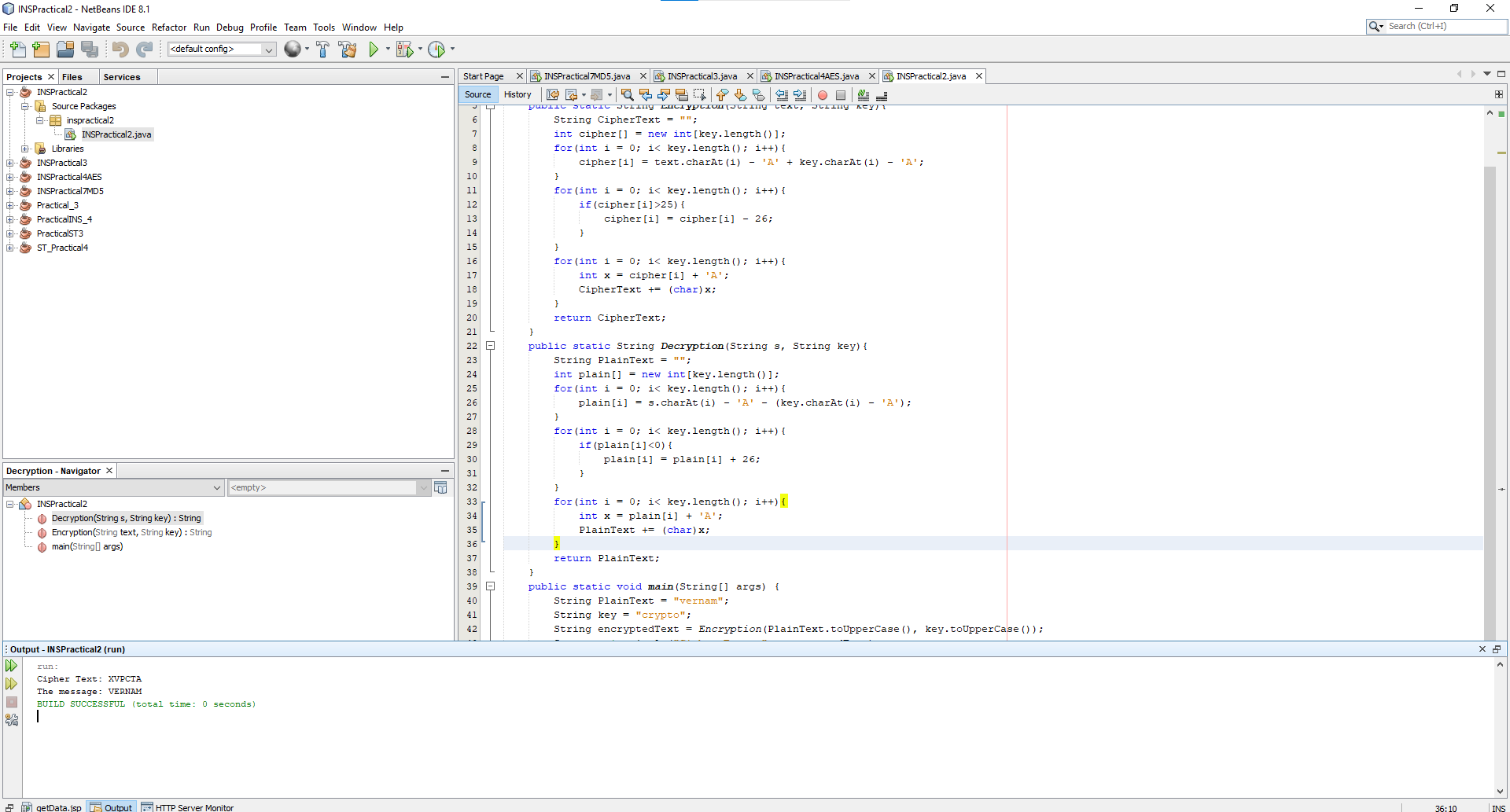
Practical 2

Name:Satishkumar Rajan Nadar

Vernamcipher



package inspractical2;

public class INSPractical2 {

public static String Encryption(String text, String key){

String CipherText = "";

int cipher[] = new int[key.length()];

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

cipher[i] = text.charAt(i) - 'A' + key.charAt(i) - 'A';

}

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

if(cipher[i]>25){

cipher[i] = cipher[i] - 26;

}

}

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

int x = cipher[i] + 'A';

CipherText += (char)x;

}

return CipherText;

}

public static String Decryption(String s, String key){

String PlainText = "";

int plain[] = new int[key.length()];

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

plain[i] = s.charAt(i) - 'A' - (key.charAt(i) - 'A');

}

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

if(plain[i]<0){

plain[i] = plain[i] + 26;

}

}

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

int x = plain[i] + 'A';

PlainText += (char)x;

}

return PlainText;

}

public static void main(String[] args) {

String PlainText = "vernam";

String key = "crypto";

String encryptedText = Encryption(PlainText.toUpperCase(), key.toUpperCase());

System.out.println("Cipher Text: " + encryptedText);

System.out.println("The message: "+ Decryption(encryptedText, key.toUpperCase()));

}

}

Playfair cipher:

package inspractical2part2;

import java.awt.Point;

import java.util.Scanner;

public class INSPractical2part2 {

private int length = 0;

private String [][] table;

public static void main(String args[])

{

INSPractical2part2 pf = new INSPractical2part2 ();

}

private INSPractical2part2 (){

System.out.print("Enter the key for playfair cipher: ");

Scanner sc = new Scanner(System.in);

String key = parseString(sc);

while(key.equals(""))

key = parseString(sc);

table = this.cipherTable(key);

System.out.print("Enter the plaintext to be encipher: ");

String input = parseString(sc);

while(input.equals(""))

input = parseString(sc);

String output = cipher(input);

String decodedOutput = decode(output);

this.keyTable(table);

this.printResults(output,decodedOutput);

}

private String parseString(Scanner sc) {

String parse = sc.nextLine();

parse = parse.toUpperCase();

parse = parse.replaceAll("[^A-Z]", "");

parse = parse.replace("J", "I");

return parse;

}

private String[][] cipherTable(String key) {

String[][] playfairTable = new String[5][5];

String keyString = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";

for(int i = 0; i < 5; i++)

for(int j = 0; j < 5; j++)

playfairTable[i][j] = "";

for(int k = 0; k < keyString.length(); k++) {

boolean repeat = false;

boolean used = false;

for(int i = 0; i < 5; i++) {

for(int j = 0; j < 5; j++) {

if(playfairTable[i][j].equals("" + keyString.charAt(k))) {

repeat = true;

}

else if(playfairTable[i][j].equals("") && !repeat && !used) {

playfairTable[i][j] = "" + keyString.charAt(k);

used = true;

}

}

}

}

return playfairTable;

}

private String cipher(String in)

{

length = (int) in.length() / 2 + in.length() % 2;

for(int i = 0; i < (length - 1); i++) {

if(in.charAt(2 \* i) == in.charAt(2 \* i + 1))

{

in = new StringBuffer(in).insert(2 \* i + 1, 'X').toString();

length = (int) in.length() / 2 + in.length() % 2;

}

}

String[] digraph = new String[length];

for(int j = 0; j < length ; j++){

if(j == (length - 1) && in.length() / 2 == (length - 1))

in = in + "X";

digraph[j] = in.charAt(2 \* j) +""+ in.charAt(2 \* j + 1);

}

String out = "";

String[] encDigraphs = new String[length];

encDigraphs = encodeDigraph(digraph);

for(int k = 0; k < length; k++)

out = out + encDigraphs[k];

return out;

}

private String[] encodeDigraph(String di[]){

String[] encipher = new String[length];

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

char a = di[i].charAt(0);

char b = di[i].charAt(1);

int r1 = (int) getPoint(a).getX();

int r2 = (int) getPoint(b).getX();

int c1 = (int) getPoint(a).getY();

int c2 = (int) getPoint(b).getY();

if(r1 == r2){

c1 = (c1 + 1) % 5;

c2 = (c2 + 1) % 5;

}

else if(c1 == c2){

r1 = (r1 + 1) % 5;

r2 = (r2 + 1) % 5;

}

else

{

int temp = c1;

c1 = c2;

c2 = temp;

}

encipher[i] = table[r1][c1] + "" + table[r2][c2];

}

return encipher;

}

private String decode(String out){

String decoded = "";

for(int i = 0; i < out.length() / 2; i++) {

char a = out.charAt(2\*i);

char b = out.charAt(2\*i+1);

int r1 = (int) getPoint(a).getX();

int r2 = (int) getPoint(b).getX();

int c1 = (int) getPoint(a).getY();

int c2 = (int) getPoint(b).getY();

if(r1 == r2){

c1 = (c1 + 4) % 5;

c2 = (c2 + 4) % 5;

}

else if(c1 == c2){

r1 = (r1 + 4) % 5;

r2 = (r2 + 4) % 5;

}

else{

int temp = c1;

c1 = c2;

c2 = temp;

}

decoded = decoded + table[r1][c1] + table[r2][c2];

}

return decoded;

}

private Point getPoint(char c){

Point pt = new Point(0,0);

for(int i = 0; i < 5; i++)

for(int j = 0; j < 5; j++)

if(c == table[i][j].charAt(0))

pt = new Point(i,j);

return pt;

}

private void keyTable(String[][] printTable){

System.out.println("Playfair Cipher Key Matrix: ");

System.out.println();

//loop iterates for rows

for(int i = 0; i < 5; i++)

{

for(int j = 0; j < 5; j++){

System.out.print(printTable[i][j]+" ");

}

System.out.println();

}

System.out.println();

}

private void printResults(String encipher, String dec){

System.out.print("Encrypted Message:");

System.out.println(encipher);

System.out.print("Decrypted Message: ");

System.out.println(dec);

}

}

