

```
00001:
00002: import java.io.*;
00003: import java.util.*;
00004:
00005: /**
00006:  * This program encrypts or decrypts files and prints content in different file.
00007:  *
00008:  * @author WahabEhsan
00009:  */
00010: public class Cipher {
00011:
00012:     /**
00013:      * Global variable for the number of letters in English alphabet.
00014:      */
00015:     public static final int Alphabet = 26;
00016:
00017:     /**
00018:      * Runs the program
00019:      *
00020:      * @param args not used
00021:      */
00022:     public static void main(String[] args) {
00023:         userInterface();
00024:     }
00025:
00026:     /**
00027:      * This method prompts user and asks if they want to encrypt, decrypt or
00028:      * quit.
00029:      */
00030:     public static void userInterface() {
00031:         Scanner input = new Scanner(System.in);
00032:         boolean loop = true;
00033:         boolean encrypte;
```

```

00034:     while (loop) {
00035:         System.out.print("Would you like to (E)ncrypte or (D)ecrypte? or (Q)uit? ");
00036:         String in = input.nextLine();
00037:         in = in.toLowerCase();
00038:         if (in.equals("q")) { //if "q" entered, program quits
00039:             loop = false;
00040:         } else if (in.equals("d")) { //if "d" entered, sets encrypt to false and runs the procedure
00041:             encrypte = false;
00042:             procedure(encrypte, input);
00043:         } else if (in.equals("e")) { //if "d" entered, sets encrypt to true and runs the procedure
00044:             encrypte = true;
00045:             procedure(encrypte, input);
00046:         } else { //else says "invalid input"
00047:             System.out.println("Invalid input");
00048:         }
00049:     }
00050: }
00051:
00052: }
00053:
00054: /**
00055:  * This method runs the methods depending on if encrypt or decrypt
00056:  *
00057:  * @param encrypte boolean value true for encrypt or false for decrypt
00058:  * @param input The input as scanner
00059:  */
00060: public static void procedure(boolean encrypte, Scanner input) {
00061:
00062:     String key = getKey(input); //Sets input to key
00063:     Scanner fileReader = getInputScanner(input); //makes filereader scanner
00064:     PrintStream fileWriter = getOutputPrintStream(input); //makes filewriter printstream
00065:     processFile(encrypte, key, fileReader, fileWriter); //processes file
00066:

```

```
00067:     }
00068:
00069:     /**
00070:      * Prompts the user for and returns a valid key for encryption/decryption.
00071:      *
00072:      * @param console Scanner for input
00073:      * @return Key is returned after is handled
00074:      */
00075:     public static String getKey(Scanner console) {
00076:         boolean perm = true;
00077:         String key = null;
00078:         while (perm) { //runs while loop as long as variable perm is true
00079:             System.out.print("Key? ");
00080:             key = console.nextLine(); //prompts user for key
00081:             if (key.length() == 0) { //if key has no length, prompts again
00082:                 System.out.println("Please Enter Lowercase letters.");
00083:                 continue;
00084:             }
00085:             perm = handlingKey(key, console); //runs handlingKey method
00086:             if (perm == false) { //if perm returned is false, breaks loop
00087:                 break;
00088:             }
00089:
00090:         }
00091:
00092:         return key;
00093:     }
00094: }
00095:
00096: /**
00097:  * Handles the key entered, prints error if the input is not lowercase
00098:  * letters
00099:  *
```

```

00100:      * @param key The key entered in getKey()
00101:      * @param console Scanner input
00102:      * @return
00103:      */
00104: public static boolean handlingKey(String key, Scanner console) {
00105:     boolean perm = false;
00106:     for (int i = 0; i < key.length(); i++) {
00107:         char c = key.charAt(i); //Scans every character to make sure not uppercase or unicode
00108:         if (!Character.isLowerCase(c) || (Character.UnicodeBlock.of(c)
00109:             != Character.UnicodeBlock.BASIC_LATIN)) {
00110:             //if uppercase or unicode, breaks loop and sets perm to true
00111:             System.out.println("Please Enter Lowercase letters.");
00112:             perm = true;
00113:             break;
00114:         }
00115:     }
00116:     return perm;
00117: }
00118:
00119: /**
00120:  * Returns Scanner for an input file Use a try/catch block to catch and
00121:  * handle any FileNotFoundException's that occur
00122:  *
00123:  * @param console Scanner input
00124:  * @return File is returned after valid file is found
00125:  */
00126: public static Scanner getInputScanner(Scanner console) {
00127:     Scanner file = null;
00128:     try {
00129:         File f; //makes file object
00130:         do {
00131:             System.out.print("Enter Input File: ");
00132:             String input = console.nextLine();

```

```

00133:         f = new File(input);
00134:         if (!f.exists()) { //if file not found, error message appears
00135:             System.out.println("File not Found");
00136:         }
00137:
00138:         } while (!f.exists()); //runs do-while loop until file found
00139:         file = new Scanner(f); //passes file into scanner
00140:
00141:     } catch (FileNotFoundException ex) { //catches FileNotFoundException
00142:         System.out.print("File not found");
00143:     }
00144:     return file; //returns scanner for file
00145: }
00146:
00147: /**
00148:  * Returns PrintStream for output file Use a try/catch block to catch and
00149:  * handle any FileNotFoundException's that occur
00150:  *
00151:  * @param console Scanner input
00152:  * @return File PrintStream returned after checked with user if file already
00153:  * exist and if they would like to overwrite
00154:  */
00155: public static PrintStream getOutputPrintStream(Scanner console) {
00156:     boolean perm = true;
00157:     File f;
00158:     PrintStream file = null;
00159:     while (file == null) { //runs while loop until file no more null
00160:         try {
00161:             System.out.print("Enter Output file: ");
00162:             String input = console.nextLine();
00163:             f = new File(input);
00164:             if (f.exists()) { //if file exists, informs user it exists
00165:                 System.out.println("File already exists.");

```

```

00166:         perm = outputHandling(perm); //then runs outputhandling method
00167:         if (perm == false) { //if perm returned is false, re starts the loop
00168:             continue;
00169:         }
00170:     }
00171:     file = new PrintStream(f); //sets printstream
00172: } catch (FileNotFoundException ex) { //catches FileNotFoundException
00173:     System.out.print("File not found");
00174: }
00175: }
00176:
00177: return file; //returns printstream
00178:
00179: }
00180:
00181: /**
00182:  * This method asks the user if they would like to overwrite the file.
00183:  *
00184:  * @param perm boolean value to continue or re-prompt for another file
00185:  * @return Boolean value to continue or re-prompt for another file or
00186:  *         overwrite
00187:  */
00188: public static boolean outputHandling(boolean perm) {
00189:     boolean loop = true;
00190:     Scanner input = new Scanner(System.in);
00191:     while (loop) { //runs while loop until valid answer given
00192:         System.out.println("Would you like to overwrite? (Y/N)");
00193:         String request = input.next();
00194:         request = request.toLowerCase(); //sets input to lowercase
00195:         if (request.contains("y")) { //if "y" entered, it breaks the loop and sets perm true
00196:             perm = true;
00197:             loop = false;
00198:         } else if (request.contains("n")) { //if "n" entered, it breaks the loop and sets perm false

```

```
00199:         perm = false;
00200:         loop = false;
00201:     } else { //if anything else entered, prints "invalid input".
00202:         System.out.println("Invalid input");
00203:     }
00204: }
00205: return perm; //returns the value of boolean if want to overwrite or not.
00206: }
00207:
00208: /**
00209:  * If encrypt is true, encrypts message in input and outputs encrypted
00210:  * message based on key, If encrypt is false, decrypts message in input and
00211:  * outputs decrypted message based on key.
00212:  *
00213:  * @param encrypt Boolean value for encrypt if true or decrypt if false,
00214:  * @param key String that is processed and checked if valid
00215:  * @param input Scanner input for file
00216:  * @param output Scanner output for PrintStream file
00217:  */
00218: public static void processFile(boolean encrypt, String key, Scanner input, PrintStream output) {
00219:     String content;
00220:     while (input.hasNextLine()) { //runs while loop until input has line
00221:         String line = input.nextLine(); //turns scanner to line
00222:         if (encrypt == true) { //if encrypt was true, runs encrypt method
00223:             content = encryptLine(key, line);
00224:         } else { //if decrypt was true, runs decrypt method
00225:             content = decryptLine(key, line);
00226:         }
00227:         output.println(content); //prints content on the printstream file
00228:     }
00229:     output.close(); //closes output
00230:     input.close(); //closes input
00231: }
```

```

00232:     }
00233:
00234:     /**
00235:      * Returns string containing line encrypted using key
00236:      *
00237:      * @param key The String value the shifts the characters for the line
00238:      * @param line The Line that processes through the input file and is shifted
00239:      * according to the key
00240:      * @return Sentence that are returned and added on
00241:      */
00242:     public static String encryptLine(String key, String line) {
00243:         String sen = "";
00244:         char cas = 0;
00245:
00246:         for (int i = 0, j = 0; i < line.length(); i++) { //runs for loop for lenght of line
00247:             char c = line.charAt(i);
00248:             if (c == '\n' || c == '\r') { // if char c is a newline, prints content and continues
00249:                 j = 0; //sets key letter to the first letter for the new line
00250:                 sen += c;
00251:                 continue;
00252:             }
00253:             if (c < 'a' || c > 'z') { //if char c is not lowercase goes to this loop
00254:                 if (Character.isUpperCase(c)) { //if char c is uppercase
00255:                     c = Character.toLowerCase(c); //sets char to c for temprarly
00256:                     cas = (char) ((c + key.charAt(j) - 2 * 'a') % Alphabet + 'a'); //shifts the char value depending on key
00257:                     cas = Character.toUpperCase(cas); //sets char back to uppercase
00258:                     sen += cas; //adds char to the rest of the line
00259:                     j = ++j % key.length(); //increases to the next key character
00260:                     continue;
00261:                 }
00262:
00263:                 sen += c; //if not a letter, just adds to the line
00264:                 continue;

```



```

00265:         }
00266:         sen += (char) ((c + key.charAt(j) - 2 * 'a') % Alphabet + 'a');
00267:         //if lowercase letter shifts and adds to line
00268:         j = ++j % key.length();
00269:     }
00270:     return sen; //returns sentence
00271: }
00272:
00273: /**
00274:  * Returns string containing line decrypted using key
00275:  *
00276:  * @param key The String value the shifts the characters for the line
00277:  * @param line The Line that processes through the input file and is shifted
00278:  * according to the key
00279:  * @return Sentence that are returned and added on
00280:  */
00281: public static String decryptLine(String key, String line) {
00282:     String sen = "";
00283:     char cas = 0;
00284:     for (int i = 0, j = 0; i < line.length(); i++) { //runs for loop for lenght of line
00285:         char c = line.charAt(i);
00286:         if (c == '\n' || c == '\r') { // if char c is a newline, prints content and continues
00287:             j = 0; //sets key letter to the first letter for the new line
00288:             sen += c;
00289:             continue;
00290:         }
00291:         if (c < 'a' || c > 'z') { //if char c is not lowercase goes to this loop
00292:             if (Character.isUpperCase(c)) { //if char c is uppercase
00293:                 c = Character.toLowerCase(c); //sets char to c for temprarly
00294:                 cas = (char) ((c - key.charAt(j) + Alphabet) % Alphabet + 'a');
00295:                 //shifts the char value depending on key
00296:                 cas = Character.toUpperCase(cas); //sets char back to uppercase
00297:                 sen += cas; //adds char to the rest of the line

```

```
00298:             j = ++j % key.length(); //increases to the next key character
00299:             continue;
00300:         }
00301:         sen += c; //if not a letter, just adds to the line
00302:         continue;
00303:     }
00304:     sen += (char) ((c - key.charAt(j) + Alphabet) % Alphabet + 'a');
00305:     //if lowercase letter shifts and adds to line
00306:     j = ++j % key.length();
00307: }
00308: return sen; //returns sentence
00309: }
00310:
00311: }
```