DES algorithm

package com.company;  
  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.OutputStream;  
import java.security.InvalidAlgorithmParameterException;  
import java.security.InvalidKeyException;  
import java.security.NoSuchAlgorithmException;  
import java.security.spec.AlgorithmParameterSpec;  
  
import javax.crypto.Cipher;  
import javax.crypto.CipherInputStream;  
import javax.crypto.CipherOutputStream;  
import javax.crypto.KeyGenerator;  
import javax.crypto.NoSuchPaddingException;  
import javax.crypto.SecretKey;  
import javax.crypto.spec.IvParameterSpec;  
  
import java.io.\*;  
import java.util.\*;  
  
public class des {  
 private static Cipher *encryptCipher*;  
 private static Cipher *decryptCipher*;  
 private static final byte[] *iv* = { 11, 22, 33, 44, 99, 88, 77, 66 };  
  
 public static void main(String[] args) {  
 String clearTextFile = "D:\\уник/source.txt";  
 String cipherTextFile = "D:\\уник/cipher.txt";  
 String clearTextNewFile = "D:\\уник/source-new.txt";  
  
 try {  
 // create SecretKey using KeyGenerator  
 SecretKey key = KeyGenerator.*getInstance*("DES").generateKey();  
 AlgorithmParameterSpec paramSpec = new IvParameterSpec(*iv*);  
  
 // get Cipher instance and initiate in encrypt mode  
 *encryptCipher* = Cipher.*getInstance*("DES/CBC/PKCS5Padding");  
 *encryptCipher*.init(Cipher.*ENCRYPT\_MODE*, key, paramSpec);  
  
 // get Cipher instance and initiate in decrypt mode  
 *decryptCipher* = Cipher.*getInstance*("DES/CBC/PKCS5Padding");  
 *decryptCipher*.init(Cipher.*DECRYPT\_MODE*, key, paramSpec);  
  
 // method to encrypt clear text file to encrypted file  
 *encrypt*(new FileInputStream(clearTextFile), new FileOutputStream(cipherTextFile));  
  
 // method to decrypt encrypted file to clear text file  
 *decrypt*(new FileInputStream(cipherTextFile), new FileOutputStream(clearTextNewFile));  
 BufferedReader br = new BufferedReader(new InputStreamReader(System.*in*));  
 FileReader fin = new FileReader("D:\\уник/cipher.txt");  
 String name;  
 int c;  
 System.*out*.println("DONE");  
 System.*out*.println("Print cipher files? y/n? ");  
 name = br.readLine();  
 if (name.equals("y"))  
 while ((c = fin.read()) != -1 ) System.*out*.print((char) c);  
  
 } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException  
 | InvalidAlgorithmParameterException | IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 private static void encrypt(InputStream is, OutputStream os) throws IOException {  
  
 // create CipherOutputStream to encrypt the data using encryptCipher  
 os = new CipherOutputStream(os, *encryptCipher*);  
 *writeData*(is, os);  
 }  
  
 private static void decrypt(InputStream is, OutputStream os) throws IOException {  
  
 // create CipherOutputStream to decrypt the data using decryptCipher  
 is = new CipherInputStream(is, *decryptCipher*);  
 *writeData*(is, os);  
 }  
  
 // utility method to read data from input stream and write to output stream  
 private static void writeData(InputStream is, OutputStream os) throws IOException {  
 byte[] buf = new byte[1024];  
 int numRead = 0;  
 // read and write operation  
 while ((numRead = is.read(buf)) >= 0) {  
 os.write(buf, 0, numRead);  
 }  
 os.close();  
 is.close();  
 }  
  
}

BullFish

package com.company;  
  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.InputStream;  
import java.io.OutputStream;  
import java.security.Key;  
import javax.crypto.Cipher;  
import javax.crypto.spec.SecretKeySpec;  
  
public class Blowfish {  
  
 private static final String *ALGORITHM* = "Blowfish";  
 private static String *key*= "knowledgefactory";  
 private static final String *SAMPLE\_FILE\_PATH* = "D:\\уник/source.txt";  
 private static final String *ENCRYPTED\_FILE\_PATH* = "D:\\уник/cipher.txt";  
 private static final String *DECRYPTED\_FILE\_PATH* = "D:\\уник/source-new.txt";  
  
 public static void main(String[] args) {  
  
 File sampleFile = new File(*SAMPLE\_FILE\_PATH*);  
 File encryptedFile = new File(*ENCRYPTED\_FILE\_PATH*);  
 File decryptedFile = new File(*DECRYPTED\_FILE\_PATH*);  
  
 try {  
 Blowfish.*encrypt*(sampleFile, encryptedFile);  
 Blowfish.*decrypt*(encryptedFile, decryptedFile);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void encrypt(File sampleFile, File outputFile)  
 throws Exception {  
 *doCrypto*(Cipher.*ENCRYPT\_MODE*, sampleFile, outputFile);  
  
 }  
  
 public static void decrypt(File sampleFile, File outputFile)  
 throws Exception {  
 *doCrypto*(Cipher.*DECRYPT\_MODE*, sampleFile, outputFile);  
  
 }  
  
 private static void doCrypto(int cipherMode, File sampleFile,  
 File outputFile) throws Exception {  
  
 Key secretKey = new SecretKeySpec(*key*.getBytes(), *ALGORITHM*);  
 Cipher = Cipher.*getInstance*(*ALGORITHM*);  
 cipher.init(cipherMode, secretKey);  
  
 InputStream = new FileInputStream(sampleFile);  
 byte[] inputBytes = new byte[(int) sampleFile.length()];  
 inputStream.read(inputBytes);  
  
 byte[] outputBytes = cipher.doFinal(inputBytes);  
  
 OutputStream = new FileOutputStream(outputFile);  
 outputStream.write(outputBytes);  
  
 inputStream.close();  
 outputStream.close();  
  
 }  
}