```
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
public class DES
{
public static void main(String[] argv) {
try{
System.out.println("Message Encryption Using DES Algorithm\n-------");
KeyGenerator keygenerator = KeyGenerator.getInstance("DES");
SecretKey myDesKey = keygenerator.generateKey();
Cipher desCipher;
desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
desCipher.init(Cipher.ENCRYPT_MODE, myDesKey);
byte[] text = "kusal shubham ".getBytes();
System.out.println("Message [Byte Format] : " + text);
System.out.println("Message : " + new String(text));
byte[] textEncrypted = desCipher.doFinal(text);
System.out.println("Encrypted Message: " + textEncrypted);
desCipher.init(Cipher.DECRYPT_MODE, myDesKey);
byte[] textDecrypted = desCipher.doFinal(textEncrypted);
System.out.println("Decrypted Message: " + new String(textDecrypted));
}catch(NoSuchAlgorithmException e){
e.printStackTrace();
}catch(NoSuchPaddingException e){
e.printStackTrace();
}catch(InvalidKeyException e){
```

```
e.printStackTrace();
}catch(IllegalBlockSizeException e){
e.printStackTrace();
}catch(BadPaddingException e){
e.printStackTrace();
}
}
}
```

Output: -

Message Encryption Using DES Algorithm

Message [Byte Format] : [B@1f32e575

Message: kusal shubham

Encrypted Message: [B@4aa298b7

Decrypted Message: kusal Shubham.

Output explanation: -

The "Message [Byte Format]" line shows the byte representation of the original message, which is "kusal shubham" in this case. The "Message" line shows the original message in string format.

The "Encrypted Message" line shows the byte representation of the encrypted message. The "Decrypted Message" line shows the decrypted message in string format, which should match the original message.

- **This Java program demonstrates how to use the Data Encryption Standard (DES) algorithm to encrypt and decrypt a message using a symmetric key.**

  In this program, a key generator is used to generate a random secret key, which is then used to initialize the DES cipher in encryption mode. The cipher is then used to encrypt the message "kusal shubham" using the ECB mode of operation and PKCS5 padding scheme.

  The encrypted message is then printed to the console. The same key is then used to initialize the DES cipher in decryption mode, and the encrypted message is decrypted back to its original form.

  The decrypted message is then printed to the console as proof that the encryption and decryption process was successful.

Note that this is just a simple example to demonstrate the use of DES encryption in Java. In practice, it is important to carefully manage the keys used for encryption to ensure the security of the encrypted data.

- here's an explanation of each line of the program:

1. **import java.security.InvalidKeyException;**
   import java.security.NoSuchAlgorithmException;
   import javax.crypto.BadPaddingException;
   import javax.crypto.Cipher;
   import javax.crypto.IllegalBlockSizeException;
   import javax.crypto.KeyGenerator;
   import javax.crypto.NoSuchPaddingException;
   import javax.crypto.SecretKey;

These are import statements that import the necessary classes for the program to use the DES encryption algorithm, as well as exception classes that may be thrown during runtime.

2. **public class DES {**

This declares a public class named "DES", which contains the main method of the program.

3. **public static void main (String [] argv) {**

This is the main method that will be executed when the program is run. It takes an array of strings as input arguments.

4. **try {**

This marks the start of a try-catch block, which is used to handle exceptions that may occur during the program's execution.

5. **System.out.println("Message Encryption Using DES Algorithm\n-------");**

This line prints a message to the console.

6. **KeyGenerator keygenerator = KeyGenerator.getInstance("DES");**
   SecretKey myDesKey = keygenerator.generateKey();

These lines create a new instance of the KeyGenerator class and use it to generate a new DES key. The key is stored in a SecretKey object named "myDesKey".

7. **Cipher desCipher;**
   desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

These lines create a new instance of the Cipher class and initialize it with the DES encryption algorithm, the ECB mode of operation, and the PKCS5 padding scheme.

8. **desCipher.init(Cipher.ENCRYPT_MODE, myDesKey);**
   byte [] text = "kusal shubham ".getBytes();

These lines initialize the cipher with the encryption mode and the DES key, and define a byte array named "text" that holds the message to be encrypted.

9. **System.out.println("Message [Byte Format] : " + text);**
   System.out.println("Message: " + new String(text));

These lines print the byte representation and the string representation of the message to be encrypted.

10. **byte [] textEncrypted = desCipher.doFinal(text);**
    System.out.println("Encrypted Message: " + textEncrypted);

These lines use the cipher to encrypt the message and store the result in a byte array named "textEncrypted". The encrypted message is printed to the console.

11. **desCipher.init(Cipher.DECRYPT_MODE, myDesKey);**
    byte [] textDecrypted = desCipher.doFinal(textEncrypted);
    System.out.println("Decrypted Message: " + new String(textDecrypted));

These lines initialize the cipher with the decryption mode and the same DES key used for encryption, and use the cipher to decrypt the encrypted message. The decrypted message is printed to the console.

12. **} catch (NoSuchAlgorithmException e) {**
    e.printStackTrace();
    } catch (NoSuchPaddingException e) {
    e.printStackTrace();
    } catch (InvalidKeyException e) {
    e.printStackTrace();
    } catch (IllegalBlockSizeException e) {
    e.printStackTrace();
    } catch (BadPaddingException e) {
    e.printStackTrace();
    }

These catch blocks handle exceptions that may be thrown during the program's execution. Each catch block prints a stack trace to the console.