```
package m9_proves;

//Tots els imports del programa
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;


public class ExerciciClau1 {
	public static void main (String[] args) {

		//Generació de claus (Clau simètrica i la basada en contrasenyes)
		SecretKey Key, Key2;
		Key = keygenKeyGeneration(256);
		Key2 = passwordKeyGeneration("casa", 256);

		String Clau = Base64.getEncoder().encodeToString(Key.getEncoded());
		String Clau2 = Base64.getEncoder().encodeToString(Key.getEncoded());

		System.out.println("La clau generada en format String Base64 és: " + Clau);
		System.out.println("La clau generada en format String Base64 és: " + Clau2);

		//Xifrar un text
		String textaXifrar = "Hola, avui fa una bona tarda";

		byte[] textXifrat = encryptData(Key, textaXifrar.getBytes());

		System.out.println("El text xiftat és: " + new String(textXifrat));

		//Desxifrar el text xifrat
		byte[] textDesxifrat = decryptData(Key, textXifrat);

		System.out.println("El text desxifrat és: " + new String(textDesxifrat));

	}

	//Generació de claus simètriques
	public static SecretKey keygenKeyGeneration(int keySize) {
		SecretKey sKey = null;
		if ((keySize == 128)||(keySize == 192)||(keySize == 256)) {
		try {
			KeyGenerator kgen = KeyGenerator.getInstance("AES");
			kgen.init(keySize);
```

```
                sKey = kgen.generateKey();

            } catch (NoSuchAlgorithmException ex) {
                System.err.println("Generador no disponible.");
             }
             }
             return sKey;
}

//Generació de claus simètriques basades en contrasenya
public static SecretKey passwordKeyGeneration(String text, int keySize) {
        SecretKey sKey = null;
        if ((keySize == 128)||(keySize == 192)||(keySize == 256)) {
        try {
                byte[] data = text.getBytes("UTF-8");
                MessageDigest md = MessageDigest.getInstance("SHA-256");
                byte[] hash = md.digest(data);
                byte[] key = Arrays.copyOf(hash, keySize/8);
                sKey = new SecretKeySpec(key, "AES");
             } catch (Exception ex) {
                 System.err.println("Error generant la clau:" + ex);
             }
             }
             return sKey;
}

//Xifrar un text (Xifratge en mode AES)
public static byte[] encryptData(SecretKey sKey, byte[] data) {
        byte[] encryptedData = null;
        try {
                Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
                cipher.init(Cipher.ENCRYPT_MODE, sKey);
                encryptedData = cipher.doFinal(data);
        } catch (Exception ex) {
                System.err.println("Error xifrant les dades: " + ex);
        }
             return encryptedData;
}

//Desxifrar el mateix text xifrat (Desxifratge en mode AES)
public static byte[] decryptData(SecretKey sKey, byte[] data) {
        byte[] decryptedData = null;
        try {
                Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
                cipher.init(Cipher.DECRYPT_MODE, sKey);
                decryptedData = cipher.doFinal(data);
        } catch (Exception ex) {
                System.err.println("Error xifrant les dades: " + ex);
```

```
                }
                 return decryptedData;
        }
}
```