DES

```
import java.util.*;
import javax.crypto.*;
import javax.crypto.spec.*;

public class DES {
   public static void main(String args[]) throws Exception {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter message to encrypt: ");
      String msg = sc.nextLine();
      byte[] message = msg.getBytes();

      System.out.print("Enter custom key: ");
      String key = sc.nextLine();
      byte[] keyData = key.getBytes();

      DESKeySpec secretKey = new DESKeySpec(keyData);
      SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
      SecretKey keyN = keyFactory.generateSecret(secretKey);

      Cipher cipher = Cipher.getInstance("DES");
      cipher.init(Cipher.ENCRYPT_MODE, keyN);
      byte[] encrypted = cipher.doFinal(message);

      cipher.init(Cipher.DECRYPT_MODE, keyN);
      byte[] decrypted = cipher.doFinal(encrypted);
      String decryptedMsg = new String(decrypted);


          System.out.println("Message: " + msg);
      System.out.println("Encrypted: " + encrypted);
      System.out.println("Decrypted: " + decryptedMsg);
   }
}
```

RSA

```
import java.math.*;
import java.util.*;

public class Main {

   public static int getGCD(int mod, int num) {
      if (mod == 0)
         return num;
      else
         return getGCD(num % mod, mod);
   }

   public static void main(String args[]) {
      int d = 0, e;
      int message = 88;
      int prime1 = 11;
      int prime2 = 17;
      int primeMul = prime1 * prime2;
      int primeMul1 = (prime1 - 1) * (prime2 - 1);

      System.out.println("primeMul1 is equal to : " + primeMul1 + "\n");

      for (e = 2; e < primeMul1; e++) {
         if (getGCD(e, primeMul1) == 1) {
            break;
         }
      }

      System.out.println("Public key e is = " + e);

      for (int m = 0; m <= 9; m++) {
         int temp = 1 + (m * primeMul1);
         if (temp % e == 0) {
            d = temp / e;
            break;
         }
      }

      System.out.println("d is : " + d);

      double cipher;
      BigInteger d_message;

      cipher = (Math.pow(message, e)) % primeMul;
      System.out.println("Cipher text is : " + cipher);
```

```
        BigInteger bigN = BigInteger.valueOf(primeMul);
        BigInteger bigC = BigDecimal.valueOf(cipher).toBigInteger();

        d_message = (bigC.pow(d)).mod(bigN);

        System.out.println("Decrypted text is : " + d_message);
    }
}
```

Caeser Cipher

```
import java.util.*;

public class CaesarCipher {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter String to Encrypt: ");
        String word = sc.nextLine();

        System.out.print("Enter K: ");
        int k = sc.nextInt();

        String encryptedWord = "";

        for (int i = 0; i < word.length(); i++) {
            int r = (int)(word.charAt(i)) - 97;
            r = (r + k) % 26;
            encryptedWord += (char)(r + 97);
        }

        System.out.println("Encrypted Text : " + encryptedWord);

        String decryptedWord = "";

        for (int i = 0; i < encryptedWord.length(); i++) {
            int r = (int)(encryptedWord.charAt(i)) - 97;
            r = (r - k) >= 0 ? (r - k) : 26 + (r - k);
            decryptedWord += (char)(r + 97);
        }

        System.out.println("Decrypted Text : " + decryptedWord);
```

```
        sc.close();
    }
}
```

Substitution Cipher

```
import java.util.*;

public class SubstitutionCipher {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a word to encrypt (lowercase letters only): ");
        String word = sc.nextLine();

        String encryptedWord = "";
        String decryptedWord = "";

        for (int i = 0; i < word.length(); i++) {
            char ch = word.charAt(i);
            if (ch >= 'a' && ch <= 'z') {
                int pos = ch - 'a';
                int newPos = 25 - pos;
                char enc = (char) (newPos + 'a');
                encryptedWord += enc;
            } else {
                encryptedWord += ch;
            }
        }

        for (int i = 0; i < encryptedWord.length(); i++) {
            char ch = encryptedWord.charAt(i);
            if (ch >= 'a' && ch <= 'z') {
                int pos = ch - 'a';
                int newPos = 25 - pos;
                char dec = (char) (newPos + 'a');
                decryptedWord += dec;
            } else {
                decryptedWord += ch;
            }
```

```
        }

        System.out.println("Encrypted Word: " + encryptedWord);
        System.out.println("Decrypted Word: " + decryptedWord);

        sc.close();
    }
}
```