AF – Gabarito: Criptografia simétrica e assimétrica

```
import java.io.UnsupportedEncodingException;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

public class AES
{
    public static byte[] cifra(String texto, String chave) throws InvalidKeyException, IllegalBlockSizeException, BadPaddingException, NoSuchAlgorithmException, NoSuchPaddingException, UnsupportedEncodingException
    {
        return cifra(texto.getBytes(), chave);
    }

    public static byte[] cifra(byte[] texto, String chave)
            throws IllegalBlockSizeException, BadPaddingException, NoSuchAlgorithmException, NoSuchPaddingException, UnsupportedEncodingException, InvalidKeyException
    {
        Key key =
```

```
                    new SecretKeySpec(chave.getBytes(StandardCharsets.UTF_8), "AES");

                    Cipher cifrador = Cipher.getInstance("AES");
                    cifrador.init(Cipher.ENCRYPT_MODE, key);
                    byte[] textoCifrado = cifrador.doFinal(texto);
                    return textoCifrado;
        }

        public static String decifra(byte[] texto, String chave)
                    throws IllegalBlockSizeException, BadPaddingException, NoSuchAlgorithmException, NoSuchPaddingException, UnsupportedEncodingException, InvalidKeyException
        {
                    Key key =
                    new SecretKeySpec(chave.getBytes(StandardCharsets.UTF_8), "AES");
                    Cipher decifrador = Cipher.getInstance("AES");
                    decifrador.init(Cipher.DECRYPT_MODE, key);
         byte[] textoDecifrado = decifrador.doFinal(texto);
         return new String(textoDecifrado);
        }

        public static void main(String[] args) throws Exception {
                    try
                    {
                                String chave = "bolabolabolabola"; //tamanho: 16
                                String texto = "O Java SE possui um conjunto amplo de bibliotecas, ferramentas e implementações comumente utilizadas em algoritmos, mecanismos e protoco-
```

```
los de segurança.";

                byte[] textoCifrado = AES.cifra(texto, chave);
                String textoDecifrado = AES.decifra(textoCifrado, chave);
                System.out.println(new String(textoCifrado));
                System.out.println(textoDecifrado);
            }
            catch(Exception e)
            {
                System.out.println(e.getMessage());
            }
        }
}
```

```
import java.security.Key;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import javax.crypto.Cipher;

public class RSA {

          public static final String ALGORITHM = "RSA";

          public static byte[] encrypt(String text, Key key) {
                    byte[] cipherText = null;
                    try {
                              // get an RSA cipher object and print the
provider
                              Cipher cipher = Ci-
pher.getInstance(ALGORITHM);
                              // encrypt the plain text using the key
                              cipher.init(Cipher.ENCRYPT_MODE, key);
                              cipherText = ci-
pher.doFinal(text.getBytes());
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    return cipherText;
          }

          public static String decrypt(byte[] text, Key key) {
                        byte[] dectyptedText = null;
                        try {
                                  // get an RSA cipher object and print
the provider
                                  Cipher cipher = Ci-
pher.getInstance(ALGORITHM);
                                  // decrypt the text using the key
                                  cipher.init(Cipher.DECRYPT_MODE, key);
                                  dectyptedText = cipher.doFinal(text);

                        } catch (Exception ex) {
                                  ex.printStackTrace();
                        }

                        return new String(dectyptedText);
          }

          public static void main(String[] args)
          {
                    try
                    {
                        KeyPairGenerator keyGen = KeyPairGenera-
tor.getInstance(ALGORITHM);
                              keyGen.initialize(1024);
                              KeyPair key = keyGen.generateKeyPair();

                                  PublicKey publicKey = key.getPublic();
                                  PrivateKey privateKey =
key.getPrivate();
```

```
                                String originalText = "O Java SE possui
um conjunto amplo de bibliotecas, ferramentas e implementações de se-
gurança.";

                                byte[] cipherText = en-
crypt(originalText, publicKey);
                                String plainText = decrypt(cipherText,
privateKey);

                                System.out.println("Original: " + origi-
nalText);
                                System.out.println("Cifrado: " + new
String(cipherText));
                                System.out.println("Decifrado: " +
plainText);

                    }
                    catch (Exception e) {
                        e.printStackTrace();
                    }
          }
}
```