ATIVIDADE PARA SER ENTREGUE DIA 01/10 - Luana Sousa - Robson Calvetti - Gestão de TI

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.MessageDigest;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.util.Base64;
import java.util.Scanner;

public class CryptoExample {

    // Criptografia Simétrica (AES)
    public static String encryptAES(String data, SecretKey secretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedData = cipher.doFinal(data.getBytes());
        return Base64.getEncoder().encodeToString(encryptedData);
    }

    public static String decryptAES(String encryptedData, SecretKey secretKey) throws
Exception {
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decryptedData = cipher.doFinal(Base64.getDecoder().decode(encryptedData));
        return new String(decryptedData);
    }

    // Criptografia Assimétrica (RSA)
    public static KeyPair generateRSAKeyPair() throws Exception {
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
        keyPairGenerator.initialize(2048);
        return keyPairGenerator.generateKeyPair();
    }

    public static String encryptRSA(String data, PublicKey publicKey) throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        byte[] encryptedData = cipher.doFinal(data.getBytes());
        return Base64.getEncoder().encodeToString(encryptedData);
    }
    public static String decryptRSA(String encryptedData, PrivateKey privateKey) throws
Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        byte[] decryptedData = cipher.doFinal(Base64.getDecoder().decode(encryptedData));
        return new String(decryptedData);
    }

    // Função Hash (SHA-256)
    public static String hashSHA256(String data) throws Exception {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        byte[] hash = digest.digest(data.getBytes());
        return Base64.getEncoder().encodeToString(hash);
    }

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

        // Criptografia Simétrica
        try {
            KeyGenerator keyGen = KeyGenerator.getInstance("AES");
            keyGen.init(128);
            SecretKey secretKey = keyGen.generateKey();

            System.out.print("Digite a mensagem para criptografar (AES): ");
            String aesInput = scanner.nextLine();
            String encryptedAES = encryptAES(aesInput, secretKey);
            System.out.println("Mensagem criptografada (AES): " + encryptedAES);

            String decryptedAES = decryptAES(encryptedAES, secretKey);
            System.out.println("Mensagem descriptografada (AES): " + decryptedAES);
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Criptografia Assimétrica
        try {
            KeyPair keyPair = generateRSAKeyPair();

            System.out.print("Digite a mensagem para criptografar (RSA): ");
            String rsaInput = scanner.nextLine();
            String encryptedRSA = encryptRSA(rsaInput, keyPair.getPublic());
            System.out.println("Mensagem criptografada (RSA): " + encryptedRSA);

            String decryptedRSA = decryptRSA(encryptedRSA, keyPair.getPrivate());
            System.out.println("Mensagem descriptografada (RSA): " + decryptedRSA);
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Função Hash
        try {
            System.out.print("Digite a mensagem para hashear: ");
            String hashInput = scanner.nextLine();
            String hashedOutput = hashSHA256(hashInput);
            System.out.println("Hash SHA-256: " + hashedOutput);
        } catch (Exception e) {
            e.printStackTrace();
        }

        scanner.close();
    }
}
```