UC- SISTEMAS COMPUTACIONAIS E SEGURANÇA
Maria Eduarda Medeiro Porto 824144948
ORIENTADOR: Robson Calvetti

PRÁTICA 05- Proteção de Dados e Informação III

ATIVIDADE 1- Pesquisar, Implementar, Codificar e Verificar Algoritmos

1. Criptografia com Chaves Simétricas (JAVA)- usando: avax.crypto

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import java.util.Base64;

public class AESCrypto {
    public static void main(String[] args) throws Exception {
        // Geração de chave AES
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(256); // AES-256
        SecretKey key = keyGenerator.generateKey();
        IvParameterSpec iv = new IvParameterSpec(new byte[16]);

        // Criptografia
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, key, iv);
        byte[] encrypted = cipher.doFinal("Exemplo de texto claro".getBytes());
        System.out.println("Criptografado: " + Base64.getEncoder().encodeToString(encrypted));

        // Descriptografia
        cipher.init(Cipher.DECRYPT_MODE, key, iv);
        byte[] decrypted = cipher.doFinal(encrypted);
        System.out.println("Descriptografado: " + new String(decrypted));
    }
}
```

2. Criptografia com chaves Assimétricas (JAVA)

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import javax.crypto.Cipher;
import java.util.Base64;

public class RSACrypto {
    public static void main(String[] args) throws Exception {
        // Geração de chaves RSA
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("RSA");
        keyPairGen.initialize(2048);
        KeyPair pair = keyPairGen.generateKeyPair();
        PublicKey publicKey = pair.getPublic();
        PrivateKey privateKey = pair.getPrivate();

        // Criptografia
        Cipher cipher = Cipher.getInstance("RSA/ECB/OAEPWithSHA-256AndMGF1Padding");
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        byte[] encrypted = cipher.doFinal("Mensagem secreta".getBytes());
        System.out.println("Criptografado: " + Base64.getEncoder().encodeToString(encrypted));

        // Descriptografia
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        byte[] decrypted = cipher.doFinal(encrypted);
        System.out.println("Descriptografado: " + new String(decrypted));
    }
}
```

3. Função HASH (JAVA)

```
import java.security.MessageDigest;

public class SHA256Hash {
    public static void main(String[] args) throws Exception {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        byte[] hash = digest.digest("Exemplo de texto".getBytes("UTF-8"));

        // Convertendo o hash para uma string hexadecimal
        StringBuilder hexString = new StringBuilder();
        for (byte b : hash) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) hexString.append('0');
            hexString.append(hex);
        }
        System.out.println("Hash SHA-256: " + hexString.toString());
    }
}
```

ATIVIDADE 2- HASHING

1.

```
import java.util.Scanner;

public class HashingAberto {
    private static final int TAMANHO_TABELA = 10;

    private static class No {
        String nome;
        No proximo;

        public No(String nome) {
            this.nome = nome;
            this.proximo = null;
        }
    }

    private static No[] tabela;

    public static void main(String[] args) {
        tabela = new No[TAMANHO_TABELA];

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.print("Digite um nome (ou 'sair' para sair): ");
            String nome = scanner.nextLine();
            if (nome.equalsIgnoreCase("sair")) {
                break;
            }

            int indice = calcularIndice(nome);
            inserir(nome, indice);
        }

        scanner.close();

        imprimirTabela();
    }
```

```
    private static int
calcularIndice(String nome) {
        return nome.length() %
TAMANHO_TABELA;
    }

    private static void
inserir(String nome, int indice) {
        No novoNo = new
No(nome);

        if (tabela[indice] == null) {
            tabela[indice] =
novoNo;
        } else {
            No atual =
tabela[indice];

            while (atual.proximo !=
null) {
                atual = atual.proximo;
            }

            atual.proximo =
novoNo;
        }
    }

    private static void
imprimirTabela() {
        for (int i = 0; i <
TAMANHO_TABELA; i++) {
            System.out.print("Índice
" + i + ": ");

            No atual = tabela[i];

            while (atual != null) {

System.out.print(atual.nome + "
-> ");
                atual = atual.proximo;
            }


System.out.println("null");
        }
    }
}
```

2.

```
import java.util.Scanner;

public class HashingAberto {
    private static final int
TAMANHO_TABELA = 5;

    private static class No {
        int numero;
        No proximo;

        public No(int numero) {
            this.numero = numero;
            this.proximo = null;
        }
    }

    private static No[] tabela;

    public static void
main(String[] args) {
        tabela = new
No[TAMANHO_TABELA];
        Scanner scanner = new
Scanner(System.in);

        while (true) {
            System.out.print("Digite
um número (ou 'sair' para sair):
");
            String input =
scanner.nextLine();

            if
(input.equalsIgnoreCase("sair")
) {
                break;
            }

            int numero =
Integer.parseInt(input);
            int indice =
calcularIndice(numero);
            inserir(numero, indice);
        }
        scanner.close();

        imprimirTabela();
    }

    private static int
calcularIndice(int numero) {
        int digitos =
contarDigitos(numero);
        return digitos %
TAMANHO_TABELA;
    }

    private static int
contarDigitos(int numero) {
        int digitos = 0;
        while (numero > 0) {
            numero /= 10;
            digitos++;
        }
        return digitos;
    }
```

```
    private static void inserir(int
numero, int indice) {
        No novoNo = new
No(numero);

        if (tabela[indice] == null) {
            tabela[indice] =
novoNo;
        } else {
            No atual =
tabela[indice];

            while (atual.proximo !=
null) {
                atual = atual.proximo;
            }

            atual.proximo =
novoNo;
        }
    }

    private static void
imprimirTabela() {
        for (int i = 0; i <
TAMANHO_TABELA; i++) {
            System.out.print("Índice
" + i + ": ");

            No atual = tabela[i];

            while (atual != null) {

System.out.print(atual.numero
+ " -> ");
                atual = atual.proximo;
            }

System.out.println("null");
        }
    }
}
```

3.

```
import java.util.Scanner;

public class HashingAberto {
    private static class No {
        int numero;
        No proximo;

        public No(int numero) {
            this.numero = numero;
            this.proximo = null;
        }
    }

    private static class Multilista {
        private No[] tabela;

        public Multilista() {
            tabela = new No[0];
        }

        public void inserir(int
numero) {
            int indice =
calcularIndice(numero);

            No novoNo = new
No(numero);

            if (tabela.length <=
indice) {
                No[] novaTabela =
new No[indice + 1];

System.arraycopy(tabela, 0,
novaTabela, 0, tabela.length);
                tabela = novaTabela;
            }

            if (tabela[indice] == null)
{
                tabela[indice] =
novoNo;
            } else {
                No atual =
tabela[indice];

                while (atual.proximo
!= null) {
                    atual =
atual.proximo;
                }

                atual.proximo =
novoNo;
            }
        }

        public void
imprimirTabela() {
            for (int i = 0; i <
tabela.length; i++) {

System.out.print("Índice " + i +
": ");

                No atual = tabela[i];

                while (atual != null) {

System.out.print(atual.numero
+ " -> ");
                    atual =
atual.proximo;
                }
```

```
System.out.println("null");
            }
        }

        private int
calcularIndice(int numero) {
            int digitos =
contarDigitos(numero);
            return digitos;
        }

        private int contarDigitos(int
numero) {
            int digitos = 0;
            while (numero > 0) {
                numero /= 10;
                digitos++;
            }
            return digitos;
        }
    }

    public static void
main(String[] args) {
        Multilista multilista = new
Multilista();

        Scanner scanner = new
Scanner(System.in);

        while (true) {
            System.out.print("Digite
um número (ou 'sair' para sair):
");
            String input =
scanner.nextLine();
            if
(input.equalsIgnoreCase("sair")
) {
                break;
            }

            int numero =
Integer.parseInt(input);

multilista.inserir(numero);
        }

        scanner.close();

        multilista.imprimirTabela();
    }
}
```

4.

```
import java.util.Scanner;

public class HashingFechado {
    private static final int
TAMANHO_TABELA = 10;
    private static final int
MAX_COLISOES = 5;

    private static class
VetorColisoes {
        int[] vetor;
        int tamanho;

        public VetorColisoes() {
            vetor = new
int[MAX_COLISOES];
            tamanho = 0;
        }

        public void adicionar(int
numero) {
            if (tamanho <
MAX_COLISOES) {
                vetor[tamanho++] =
numero;
            } else {

System.out.println("Estouro de
colisões na posição!");
            }
        }

        public void imprimir() {
            for (int i = 0; i <
tamanho; i++) {

System.out.print(vetor[i] + " ");
            }
            System.out.println();
        }
    }

    private static VetorColisoes[]
tabela;

    public static void
main(String[] args) {
        tabela = new
VetorColisoes[TAMANHO_TAB
ELA];

        for (int i = 0; i <
TAMANHO_TABELA; i++) {
            tabela[i] = new
VetorColisoes();
        }

        Scanner scanner = new
Scanner(System.in);

        while (true) {
```

```
        System.out.print("Digite
um número (ou 'sair' para sair):
");
        String input =
scanner.nextLine();

        if
(input.equalsIgnoreCase("sair")
) {
            break;
        }

        int numero =
Integer.parseInt(input);
        int indice =
calcularIndice(numero);

tabela[indice].adicionar(numero
);
        }

        scanner.close();

        imprimirTabela();
    }

    private static int
calcularIndice(int numero) {
        int digitos =
contarDigitos(numero);
        return digitos %
TAMANHO_TABELA;
    }

    private static int
contarDigitos(int numero) {
        int digitos = 0;
        while (numero > 0) {
            numero /= 10;
            digitos++;
        }
        return digitos;
    }

    private static void
imprimirTabela() {
        for (int i = 0; i <
TAMANHO_TABELA; i++) {
            System.out.print("Índice
" + i + ": ");
            tabela[i].imprimir();
        }
    }
}
```