UC- SISTEMAS COMPUTACIONAIS E SEGURANÇA

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));

}

}

# 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));

}

}

# 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 !=

}

atual.proximo = novoNo;

}

}

private static void imprimirTabela() {

for (int i = 0; i < TAMANHO\_TABELA; i++) {

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

while (atual != null) { System.out.print(atual.numero

+ " -> ");

atual = atual.proximo;

}

System.out.println("null");

}

}

}

null) {

atual = atual.proximo;

No atual = tabela[i];

# 3.

import java.util.Scanner;

public class HashingAberto { private static class No {

int numero; No proximo;

No novoNo = new No(numero);

if (tabela.length <= indice) {

No[] novaTabela = new No[indice + 1];

novoNo;

}

}

}

atual.proximo =

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

}

}

System.arraycopy(tabela, 0,

novaTabela, 0, tabela.length);

tabela = novaTabela;

}

public void imprimirTabela() {

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

System.out.print("Índice " + i +

private static class Multilista { private No[] tabela;

if (tabela[indice] == null)

{

tabela[indice] =

": ");

No atual = tabela[i];

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

}

public void inserir(int numero) {

int indice = calcularIndice(numero);

novoNo;

} else {

No atual = tabela[indice];

while (atual.proximo

!= null) {

atual = atual.proximo;

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;

}

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) {

public void adicionar(int } numero) {

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();

}

}

}