# PROBLEMA 1

## Clase Operaciones

import java.util.Arrays;

public class Operaciones {

public static boolean numerodeamstrong(int numero) {

int numeroOriginal = numero;

int numeroDigitos = numerodedigitos(numero);

int suma = 0;

while (numero > 0) {

int digito = numero % 10;

suma += Math.pow(digito, numeroDigitos);

numero /= 10;

}

return suma == numeroOriginal;

}

public static int numerodedigitos(int numero) {

int contador = 0;

while (numero != 0) {

numero /= 10;

contador++;

}

return contador;

}

public static double sumaserie(int n) {

double suma = 0.0;

for (int i = 1; i <= n; i++) {

suma += 1.0 / i;

}

return suma;

}

public static String fibonacci(int n) {

long[] fibonacciserie = new long[n];

if (n >= 1) {

fibonacciserie[0] = 0;

}

if (n >= 2) {

fibonacciserie[1] = 1;

}

for (int i = 2; i < n; i++) {

fibonacciserie[i] = fibonacciserie[i - 1] + fibonacciserie[i - 2];

}

return Arrays.toString(fibonacciserie);

}

}

## Prueba de Amstrong

public class PruebaAmstrong {

public static void main(String[] args) {

System.out.println("PRUEBA DE AMSTRONG");

System.out.println("371: " + Operaciones.numerodeamstrong(371));

System.out.println("6578: " + Operaciones.numerodeamstrong(6789));

System.out.println("1: " + Operaciones.numerodeamstrong(1));

}

}

PRUEBA DE AMSTRONG

371: true

6578: false

1: true

## Prueba de Fibonacci

public class PruebaFibonacci {

public static void main(String[] args) {

System.out.println("\ti\tfinonacci");

for(int i=1; i<=5;i++){

System.out.println("\t" + i + "\t" + Operaciones.fibonacci(i));

}

}

}

i finonacci

1 [0]

2 [0, 1]

3 [0, 1, 1]

4 [0, 1, 1, 2]

5 [0, 1, 1, 2, 3]

## Prueba de la Serie

public class PruebaSerie {

public static void main(String[] args) {

System.out.println("\ti\tserie");

for (int i = 1; i <= 5; i++) {

System.out.println("\t" + i + "\t" + Operaciones.sumaserie(i));

}

}

}

i serie

1 1.0

2 1.5

3 1.8333333333333333

4 2.083333333333333

5 2.283333333333333

# PROBLEMA 2

## Clase ArregloService

import java.util.Arrays;

import java.util.Random;

public class Service {

private int[] arreglo1;

private int[] arreglo2;

public Service(int n) {

// Variables: Inicialización de los arreglos

arreglo1 = generarArregloAleatorio(n);

arreglo2 = generarArregloAleatorio(n);

}

// Método privado para generar un arreglo aleatorio de tamaño n

private int[] generarArregloAleatorio(int n) {

// Variables: Declaración e inicialización de variables locales

int[] arreglo = new int[n];

Random random = new Random();

// Proceso: Llenar el arreglo con números aleatorios en el rango [21, 30]

for (int i = 0; i < n; i++) {

arreglo[i] = random.nextInt(10) + 21;

}

// Reporte: Retornar el arreglo generado

return arreglo;

}

public int[] getArreglo1() {

// Reporte: Retornar el primer arreglo

return arreglo1;

}

public int[] getArreglo2() {

// Reporte: Retornar el segundo arreglo

return arreglo2;

}

// Método para calcular la diferencia entre los arreglos

public int[] arregloDiferencia() {

// Variables: Declaración e inicialización de variables locales

int[] diferencia = new int[arreglo1.length];

int index = 0;

// Proceso: Calcular la diferencia entre los arreglos

for (int num : arreglo1) {

if (!contiene(arreglo2, num) && !contiene(diferencia, num)) {

diferencia[index] = num;

index++;

}

}

// Reporte: Retornar el arreglo de diferencia

return Arrays.copyOf(diferencia, index);

}

// Método para calcular la intersección entre los arreglos

public int[] arregloInterseccion() {

// Variables: Declaración e inicialización de variables locales

int[] interseccion = new int[arreglo1.length];

int index = 0;

// Proceso: Calcular la intersección entre los arreglos

for (int num : arreglo1) {

if (contiene(arreglo2, num) && !contiene(interseccion, num)) {

interseccion[index] = num;

index++;

}

}

// Reporte: Retornar el arreglo de intersección

return Arrays.copyOf(interseccion, index);

}

// Método para verificar si un elemento existe en un arreglo

private boolean contiene(int[] arreglo, int elemento) {

// Proceso: Verificar si el elemento existe en el arreglo

for (int num : arreglo) {

if (num == elemento) {

return true;

}

}

// Reporte: Retornar verdadero si el elemento se encuentra en el arreglo, de lo contrario, falso

return false;

}

}

## Clase de Prueba

import java.util.Arrays;

public class Prueba {

public static void main(String[] args) {

int n = 5;

Service servicio;

servicio = new Service(n);

// Variables

int[] arreglo1 = servicio.getArreglo1();

int[] arreglo2 = servicio.getArreglo2();

int[] diferencia = servicio.arregloDiferencia();

int[] interseccion = servicio.arregloInterseccion();

// Reporte

System.out.println("Arreglo 1: " + Arrays.toString(arreglo1));

System.out.println("Arreglo 2: " + Arrays.toString(arreglo2));

System.out.println("Arreglo Diferencia: " + Arrays.toString(diferencia));

System.out.println("Arreglo Intersección: " + Arrays.toString(interseccion));

}

}

Arreglo 1: [23, 27, 27, 26, 22]

Arreglo 2: [25, 28, 27, 27, 23]

Arreglo Diferencia: [26, 22]

Arreglo Intersección: [23, 27]