

Facultad de Ciencias de la Administración

Escuela de Ciencias de la Computación **Programación I**

**Tema:**

Colecciones y clases predefinidas para la resolución de ejercicios en java.

**Autores:**

Pablo Vélez y Rodrigo Ortiz

**Profesor:**

SALGADO ARTEAGA JUAN CARLOS

**28/05/2023**

**Índice:**

[**Introducción: 3**](#_heading=h.2571ayj8rubm)

[**Contenidos: 5**](#_heading=h.3h32fwew9a4s)

[**Ejercicio 1: 5**](#_heading=h.gh5x934cv8uy)

[Investigación 5](#_heading=h.49lk9ggcqbz6)

[Código 6](#_heading=h.dfojh3xjm7nh)

[Prueba de escritorio 10](#_heading=h.z45ehbhze3wf)

[**Ejercicio 2 10**](#_heading=h.5ralt31a7mbe)

[Investigación: 10](#_heading=h.b5j6hdoe3c29)

[Código: 11](#_heading=h.ul6y7c1dikbi)

[Pruebas de computador: 16](#_heading=h.z451onrba8ty)

[**Ejercicio 3 17**](#_heading=h.eehpcyddi8t3)

[Investigación 17](#_heading=h.tjfiapgngvg1)

[Código 17](#_heading=h.op2wrhoq4v76)

[Prueba de escritorio 21](#_heading=h.cgt7wwvwv1o4)

[**Ejercicio 4 21**](#_heading=h.xb27fwhvuykz)

[Investigación: 21](#_heading=h.ogd8i6ezudn5)

[Código: 21](#_heading=h.3oodc5l5c2qa)

[Pruebas de computador: 26](#_heading=h.pjym2xou6mz8)

[**Ejercicio 5 27**](#_heading=h.37qptgetkxtg)

[Investigación 27](#_heading=h.dj1iqc89tu85)

[Código 27](#_heading=h.ba3k4sptatrr)

[Pruebas de computador 32](#_heading=h.i7q981s3o5o3)

[**Ejercicio 6 32**](#_heading=h.3dyu54qad4f5)

[Código:  
/\* 32](#_heading=h.lq739p4kdsgk)

[Prueba de escritorio: 40](#_heading=h.tvwoc7850w24)

[**Ejercicio 7 40**](#_heading=h.w4vunmn6jvat)

[Investigación 40](#_heading=h.ky07zw5dau56)

[Pruebas de computador 47](#_heading=h.de1313p6xhe1)

[**CONCLUSIONES: 49**](#_heading=h.medcbnnsxxdl)

[**REFERENCIAS BIBLIOGRÁFICAS: 50**](#_heading=h.7cepfs2xgttz)

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Introducción:

La programación se basa en el uso de estructuras de datos para almacenar y manipular información de manera eficiente. Entre las herramientas más versátiles y fundamentales en este ámbito se encuentran las colecciones y las clases predefinidas. Estos conceptos desempeñan un papel esencial en el desarrollo de aplicaciones y sistemas informáticos, ya que permiten organizar y gestionar conjuntos de datos de manera efectiva.

Las colecciones son estructuras de datos que agrupan elementos relacionados entre sí, brindando diversas funcionalidades para su manipulación. Pueden ser utilizadas para almacenar y recuperar datos de manera ordenada, realizar búsquedas eficientes, realizar operaciones de agregación y muchas otras tareas. Algunos ejemplos comunes de colecciones incluyen listas, conjuntos y diccionarios. Estas estructuras proporcionan flexibilidad y adaptabilidad a la hora de abordar diferentes tipos de problemas de programación.

Por otro lado, las clases predefinidas son un conjunto de estructuras y funciones proporcionadas por el lenguaje de programación que permiten crear objetos y definir su comportamiento. Estas clases predefinidas ofrecen una amplia gama de funcionalidades listas para usar, lo que facilita el desarrollo de aplicaciones sin necesidad de implementar todo desde cero. Algunos ejemplos de clases predefinidas son las clases de manejo de fechas y horas, manipulación de cadenas de texto y operaciones matemáticas.

La utilización de colecciones y clases predefinidas en la programación permite optimizar el tiempo y los recursos, ya que proporcionan soluciones probadas y eficientes para una variedad de tareas comunes. Además, brindan un enfoque estructurado y modular para organizar y manipular datos, lo que facilita el mantenimiento y la escalabilidad del código.

OBJETIVOS:

* Comprender el concepto y la importancia de las colecciones en la programación: El objetivo principal es adquirir conocimientos sólidos sobre las colecciones y su papel fundamental en el almacenamiento y manipulación eficiente de datos. Esto implica comprender los diferentes tipos de colecciones disponibles y sus características, así como aprender a seleccionar la colección más adecuada para cada situación.
* Dominar el uso de las clases predefinidas en el desarrollo de aplicaciones: El objetivo es familiarizarse con las clases predefinidas proporcionadas por el lenguaje de programación en uso y aprender a aprovechar sus funcionalidades. Esto implica comprender cómo instanciar y utilizar objetos de estas clases, así como explorar las diferentes operaciones y métodos disponibles para manipular datos de manera eficiente.
* Aplicar las colecciones y clases predefinidas en la solución de problemas: El objetivo es practicar la aplicación de las colecciones y clases predefinidas en la resolución de problemas reales. Esto implica desarrollar habilidades para identificar el mejor enfoque utilizando las colecciones y clases disponibles, así como aplicar técnicas de optimización y buenas prácticas de programación para obtener soluciones eficientes y robustas.
* Mejorar la capacidad de diseño y estructuración del código: El objetivo es desarrollar habilidades en la organización y estructuración del código al utilizar colecciones y clases predefinidas. Esto implica comprender cómo modularizar y reutilizar el código de manera efectiva, utilizando las colecciones y clases de forma coherente y legible, lo que facilita el mantenimiento y escalabilidad del software.
* Ampliar el conocimiento sobre las posibilidades de las colecciones y clases predefinidas: El objetivo es explorar y descubrir las funcionalidades más avanzadas y menos conocidas de las colecciones y clases predefinidas. Esto implica investigar y experimentar con características más avanzadas, como técnicas de filtrado, mapeo, ordenamiento y manipulación de datos complejos, para aprovechar al máximo las capacidades de estas herramientas.

# 

# Contenidos:

# Ejercicio 1:

## Investigación

Para resolver este ejercicio se tuvo que implementar una validación mediante expresiones regulares y tras eso lo que se hizo fue dividir la línea digitada en un arreglo de enteros para poder aumentar los elementos del arreglo de contadores según la cantidad de repeticiones dentro de un rango.

## Código

import java.util.Scanner;

/\*\*

\*

\* @author Rodrigo Ortiz y Pablo Velez

\*/

public class Histograma {

public static void main(String[] args) {

String op = "";

Scanner sc = new Scanner(System.in);

String nums;

do {

System.out.println("Ingrese números reales positivos (0 a 30) separados por un espacio en blanco:");

nums = sc.nextLine();

while(!nums.matches("(([0-2]?[0-9]{1}(\\.\\d+)?)|(30(\\.0+)?))(( [0-2]?[0-9]{1}(\\.\\d+)?)|( 30(\\.0+)?))\*")){

System.out.println("ERROR! Ingrese solo numeros del 0 al 30 separados por un espacio");

System.out.print("Digite nuevamente: ");

nums = sc.nextLine();

}

System.out.println("\nHISTOGRAMA");

imprimirHistograma(nums);

System.out.print("Desea continuar S/N: ");

op = sc.nextLine();

while(!op.matches("[nNsS]")){

System.out.print("ERROR! Ingrese S/N: ");

op = sc.nextLine();

}

}while(!op.matches("N|n"));

}

public static void imprimirHistograma(String s) {

String[] arregloCad = s.split(" ");

double[] arregloNum = new double[arregloCad.length];

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

arregloNum[i] = Double.parseDouble(arregloCad[i]);

}

int contRangos[] = new int[9];

String rangos[] = {"0-5","5-10","10-14","14-17","17-20","20-24","24-27","27-30","30"};

for (double elem : arregloNum) {

if(elem==30){

contRangos[8]++;

}else if(elem >= 27){

contRangos[7]++;

}else if(elem >= 24){

contRangos[6]++;

}else if(elem >= 20){

contRangos[5]++;

}else if(elem >= 17){

contRangos[4]++;

}else if(elem >= 14){

contRangos[3]++;

}else if(elem >= 10){

contRangos[2]++;

}else if(elem >= 5){

contRangos[1]++;

}else if(elem >= 0){

contRangos[0]++;

}

}

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

System.out.print(rangos[i]+"\t");

for (int j = 0; j < contRangos[i]; j++) {

System.out.print("+ ");

}

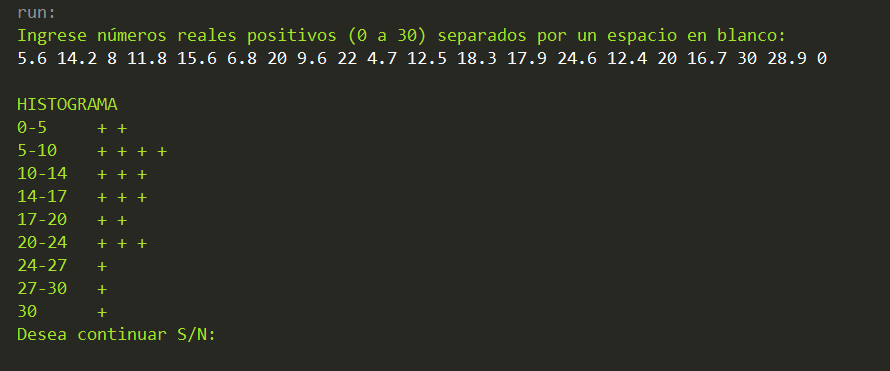
System.out.println("");

}

}

}

## Prueba de escritorio



# Ejercicio 2

## Investigación:

El programa comienza solicitando al usuario que ingrese un número decimal. Luego, se verifica si la entrada cumple con el formato de un número decimal válido utilizando expresiones regulares. Si la entrada no es válida, se muestra un mensaje de error y se solicita al usuario que ingrese nuevamente el número decimal. Una vez que se ha validado la entrada, el programa procede a convertir el número decimal ingresado a su representación binaria.

El número decimal se divide en su parte entera y fraccionaria para realizar las conversiones por separado.

Debemos conocer que la parte entera se transforma mediante divisiones sucesivas mientras que la fraccionaria por multiplicaciones sucesivas hasta obtener los dígitos binarios necesarios.

Finalmente, el programa muestra en pantalla el número binario obtenido, tanto la parte entera como la parte fraccionaria.

## Código:

import java.util.Scanner;

public class decimalBinario {

public static void convertirEnteros(double enteros[], String enterosBin[]) {

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

enterosBin[i] = "";

while (enteros[i] >= 2) {

enterosBin[i] = String.valueOf((int) enteros[i] % 2) + enterosBin[i];

enteros[i] /= 2;

}

enterosBin[i] = String.valueOf((int) enteros[i]) + enterosBin[i];

}

}

public static void convertirDecimales(double decimales[], String decimalesBin[]) {

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

decimalesBin[i] = "";

while (decimales[i] != 0 && decimalesBin[i].length() < 32) {

decimales[i] \*= 2;

if (decimales[i] >= 1) {

decimalesBin[i] += "1";

decimales[i] -= 1;

} else {

decimalesBin[i] += "0";

}

}

decimalesBin[i] = "0." + decimalesBin[i];

}

}

public static void escribirArray(String array1[], String array2[], String array3[]) {

System.out.println("Decimal\tBinario");

for (int i = 0; i < array1.length; i++) {

System.out.println(array3[i] + "\t" + array1[i] + "." + array2[i]);

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String op;

do {

System.out.print("Ingrese los números en decimal: ");

String cadena = scanner.nextLine();

while (!cadena.matches("((((\\+)?[0-9]+\\.[0-9]+)|((\\+)?[0-9]))( )?)+")) {

System.out.print("ERROR! Ingrese números reales positivos en sistema decimal separados por un espacio en blanco: ");

cadena = scanner.nextLine();

}

String[] separar = cadena.split(" ");

double[] enteros = new double[separar.length];

double[] decimales = new double[enteros.length];

String[] enterosBin = new String[enteros.length];

String[] decimalesBin = new String[decimales.length];

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

double numero = Double.parseDouble(separar[i]);

enteros[i] = Math.floor(numero);

decimales[i] = numero - enteros[i];

}

convertirEnteros(enteros, enterosBin);

convertirDecimales(decimales, decimalesBin);

escribirArray(enterosBin, decimalesBin, separar);

System.out.print("Desea continuar S/N: ");

op = scanner.nextLine();

while (!op.matches("[nNsS]")) {

System.out.print("ERROR! Ingrese S/N: ");

op = scanner.nextLine();

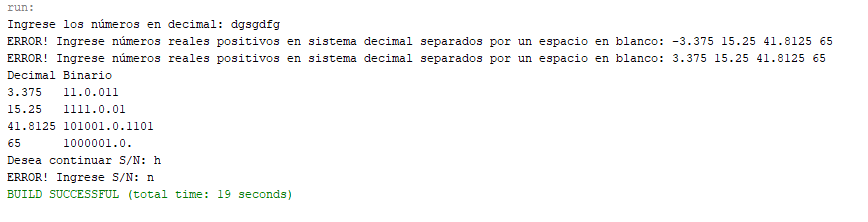
}

} while (!op.matches("[nN]"));

}

}

## Pruebas de computador:

****

# Ejercicio 3

## Investigación

Para realizar este programa tuvimos que investigar la condición para que un triángulo exista, el cual es que la suma de sus dos lados menores debe ser mayor al lado mayor. Una vez teniendo eso en cuenta validamos la entrada de datos mediante expresiones regulares, permitiendo que el usuario únicamente ingrese valores reales separados por un espacio. Tras eso almacenamos el valor de los lados en un arreglo y lo ordenamos, de tal manera que podíamos conocer los valores menores de cada lado mayor tan solo con saber en qué posición del arreglo se encuentra. Tras eso realizamos tres bucles “for”, recorriendo el lado mayor y consecutivamente sus dos lados menores, de esa manera pudimos conocer todas las combinaciones posibles, pero aun así faltaba validar la condición en la que un triángulo posible, entonces agregamos esa condición e imprimimos los valores que la cumplian.

## Código

import java.util.Scanner;

/\*\*

\*

\* @author Rodrigo Ortiz y Pablo Velez

\*/

public class CombinacionesTriangulos {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s,op;

do{

System.out.println("Ingrese números reales positivos separados por un espacio en blanco:");

s = sc.nextLine();

while(!s.matches("([0-9]+(\\.[0-9]+)?)(\\s[0-9]+(\\.[0-9]+)?){2,}")){

System.out.println("Error! Debe ingresar numeros reales positivos separados por un espacio: ");

s = sc.nextLine();

}

calcularPosibilidades(s);

System.out.print("Desea continuar?(s/n): ");

op = sc.nextLine();

while(!op.matches("[sSnN]")){

System.out.print("Error! Debe ingresar (s/n): ");

op = sc.nextLine();

}

System.out.print("\n");

}while(!op.matches("[nN]"));

}

public static void calcularPosibilidades(String s){

String cadNum[] = s.split(" ");

double nums[] = new double [cadNum.length];

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

nums[i] = Double.parseDouble(cadNum[i]);

}

double aux;

boolean intercambio;

for (int i = 0; i < nums.length-1; i++) {

intercambio = false;

for (int j = 0; j < nums.length-i-1; j++) {

if(nums[j]>nums[j+1]){

aux = nums[j];

nums[j] = nums[j+1];

nums[j+1] = aux;

intercambio = true;

}

}

if(!intercambio)

break;

}

System.out.println("\n---COMBINACIONES POSIBLES---");

for (int i = 2; i < nums.length; i++) {

for (int j = 0; j < i-1; j++) {

for (int k = j+1; k < i; k++) {

if((nums[j]+nums[k])>nums[i]){

System.out.println(nums[j]+"\t"+nums[k]+"\t"+nums[i]);

}

}

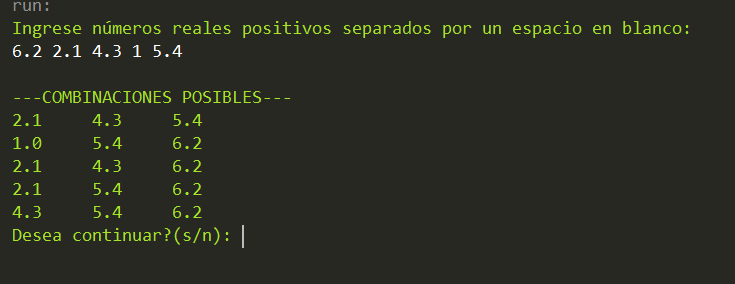
}

}

}

}

## Prueba de escritorio

****

# Ejercicio 4

## Investigación:

Un palíndromo es una palabra o frase que se lee igual en un sentido que en otro, para codificar esto debemos utilizar las funciones de la clase String, ignorando mayúsculas, minúsculas, signos de puntuación, tildes y espacios.  
Una vez con la cadena lista para comparar se utiliza un bucle for para ver si es o no igual a su inversa, devolviendo un valor booleano para a partir de eso mostrar un mensaje al usuario.

## Código:

/\*

4. Un palíndromo es una palabra o frase que se lee igual de izquierda a derecha, que de derecha a

izquierda. Escriba un programa que verifique si una cadena ingresada es un palíndromo. El

programa debe ignorar mayúsculas, tildes, espacios en blanco y signos de puntuación

\* @author Pablo Vélez y Rodrigo Ortiz MARZO - JULIO 2023

\*/

import java.util.Scanner;

public class palindromo {

public static boolean palindromo(String cadena) {

cadena = cadena.toLowerCase();

cadena = cadena.replaceAll("\\.", "");

cadena = cadena.replaceAll(",", "");

cadena = cadena.replaceAll("á", "a");

cadena = cadena.replaceAll("é", "e");

cadena = cadena.replaceAll("í", "i");

cadena = cadena.replaceAll("ó", "o");

cadena = cadena.replaceAll("ú", "u");

cadena = cadena.replaceAll(" ", "");

System.out.println(cadena);

char array[] = cadena.toCharArray();

int tamanio;

if (array.length % 2 == 0) {

tamanio = array.length / 2;

} else {

tamanio = (array.length / 2) + 1;

}

boolean palindromo = true;

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

if (array[i] != array[array.length - 1 - i]) {

palindromo = false;

break;

}

}

return palindromo;

}

public static void main(String[] args) {

String op = "";

Scanner s = new Scanner(System.in, "ISO-8859-1");

do {

System.out.print("Ingrese la palabra a verificar: ");

String cadena = s.nextLine();

while (!cadena.matches("([A-Za-z\u00E1\u00E9\u00ED\u00F3\u00FA\u00C1\u00C9\u00CD\u00D3\u00DA\u00F1\u00D1,\\.]+ ?)+")) {

System.out.print("ERROR! La cadena solo puede contener letras y signos de puntuacion: ");

cadena = s.nextLine();

}

if (palindromo(cadena)) {

System.out.println("La cadena es un palindromo");

} else {

System.out.println("La cadena no es un palindromo");

}

System.out.print("Desea continuar S/N: ");

op = s.nextLine();

while (!op.matches("[nNsS]")) {

System.out.print("ERROR! Ingrese S/N: ");

op = s.nextLine();

}

}while(!op.matches("[nN]"));

}

}

## Pruebas de computador:

# Ejercicio 5

## Investigación

Lo primero que se investigó fue la manera en la que se puede pasar un número entero a un número romano, tras eso llegamos a la conclusión de que podíamos utilizar arreglos paralelos para tener una cadena asignada a cada valor y de esa manera ir restando hasta que el número llegue a cero. Implementamos un menú que permite la repetición del programa, dentro de este menú se lee y valida el formato DD/MM/AAAA con expresiones regulares la fecha a convertir, tras eso se realiza otra validación la cual nos permite controlar que la fecha ingresada no sea mayor a la actual, nos ayudamos de la clase Calendar y de sus métodos que nos permiten comparar dos fechas para eso. Tras tener todo validado procedimos a implementar un método el cual convierte números a números romanos bajo la lógica explicada al inicio, después tras dividir los días, meses y años de la fecha ingresada enviamos cada uno de ellos al método que realiza el cambio a números romanos y después concatenamos la cadena devuelta con un guión y la imprimimos.

## Código

import java.util.Calendar;

import java.util.Scanner;

/\*\*

\*

\* @author Rodrigo Ortiz y Pablo Vélez

\*/

public class FechaRomanos {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String cad,op;

do{

System.out.print("Ingrese fecha en el formato DD/MM/AAAA: ");

cad = sc.nextLine();

while(!validarFormato(cad) || !validarFecha(cad)){

System.out.print("Error! Ingrese fecha valida hasta hoy en el formato DD/MM/AAAA: ");

cad = sc.nextLine();

};

System.out.println("Fecha en numeros romanos: "+fechaRomana(cad));

System.out.print("Desea continuar S/N: ");

op = sc.nextLine();

while(!op.matches("[nNsS]")){

System.out.print("ERROR! Ingrese S/N: ");

op = sc.nextLine();

}

System.out.println("");

}while(!op.matches("N|n"));

}

public static boolean validarFormato(String s){

return s.matches("(([1-2][0-9])|(3[0-1])|(0?[1-9]))/((1[0-2])|(0?[1-9]))/[1-9][0-9]?[0-9]?[0-9]?");

}

public static boolean validarFecha(String s){

String fecha[] = s.split("/");

int mes = Integer.parseInt(fecha[1]);

int dias = 30;

if(((mes<=7)&&(mes%2!=0))||((mes>=8)&&(mes%2==0))){

dias = 31;

}else if(mes==2){

dias = 28;

}

if( !(Integer.parseInt(fecha[0]) <= dias)){

return false;

}

Calendar fechaActual = Calendar.getInstance();

Calendar fechaIngresada = Calendar.getInstance();

fechaIngresada.set(Integer.parseInt(fecha[2]), Integer.parseInt(fecha[1]), Integer.parseInt(fecha[0]));

if(fechaIngresada.compareTo(fechaActual)>0){

return false;

}

return true;

}

public static String fechaRomana(String s){

String fecha[] = s.split("/");

StringBuilder fechaRomana = new StringBuilder();

for (String string : fecha) {

fechaRomana.append(convertirANumerosRomanos(Integer.parseInt(string)));

fechaRomana.append("-");

}

fechaRomana.deleteCharAt(fechaRomana.length()-1);

return fechaRomana.toString();

}

public static String convertirANumerosRomanos(int numero) {

int[] valoresNumericos = { 1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1 };

String[] simbolosRomanos = { "M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I" };

StringBuilder resultado = new StringBuilder();

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

while (numero >= valoresNumericos[i]) {

resultado.append(simbolosRomanos[i]);

numero -= valoresNumericos[i];

}

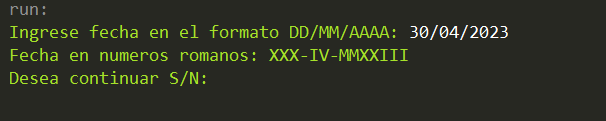
}

return resultado.toString();

}

}

## Pruebas de computador



# Ejercicio 6

## 

## **Código:** /\*

6. Escribir un programa que permita realizar operaciones con fechas

@autor Pablo Vélez y Rodrigo Ortiz MARZO - JULIO 2023

\*/

import java.util.Scanner;

import java.util.Calendar;

public class operacionesFechas {

public static void main(String[] args) {

Scanner s = new Scanner(System.in, "ISO-8859-1");

String op;

do {

System.out.print("Ingrese la fecha y la operación: ");

String entrada = s.nextLine();

boolean flag = verificarFormato(entrada);

while (!flag) {

System.out.print("ERROR! el formato debe ser DD-Nombre de mes-AAAA con las operaciones (+/-/cmp) separadas por espacios antes y despues: ");

entrada = s.nextLine();

flag = verificarFormato(entrada);

}

String[] partes = entrada.split(" ");

String fecha = partes[0];

String operacion = partes[1];

int dia = Integer.parseInt(fecha.substring(0, 2));

String mes = fecha.substring(3, fecha.indexOf("-", 3));

int anio = Integer.parseInt(fecha.substring(fecha.lastIndexOf("-") + 1));

Calendar calendario = Calendar.getInstance();

calendario.clear();

calendario.set(Calendar.DAY\_OF\_MONTH, dia);

calendario.set(Calendar.MONTH, obtenerIndiceMes(mes));

calendario.set(Calendar.YEAR, anio);

if (operacion.equals("+")) {

int diasASumar = Integer.parseInt(partes[2]);

calendario.add(Calendar.DAY\_OF\_MONTH, diasASumar);

} else if (operacion.equals("-")) {

int diasARestar = Integer.parseInt(partes[2]);

calendario.add(Calendar.DAY\_OF\_MONTH, -diasARestar);

} else if (operacion.equals("cmp")) {

String fecha2 = partes[2];

int dia2 = Integer.parseInt(fecha2.substring(0, 2));

String mes2 = fecha2.substring(3, fecha2.indexOf("-", 3));

int anio2 = Integer.parseInt(fecha2.substring(fecha2.lastIndexOf("-") + 1));

Calendar calendario2 = Calendar.getInstance();

calendario2.clear();

calendario2.set(Calendar.DAY\_OF\_MONTH, dia2);

calendario2.set(Calendar.MONTH, obtenerIndiceMes(mes2));

calendario2.set(Calendar.YEAR, anio2);

int comparacion = compararFechas(calendario, calendario2);

if (comparacion < 0) {

System.out.println(fecha + " es menor que " + fecha2);

} else if (comparacion > 0) {

System.out.println(fecha + " es mayor que " + fecha2);

} else {

System.out.println(fecha + " es igual a " + fecha2);

}

//return;

}

int resultadoDia = calendario.get(Calendar.DAY\_OF\_MONTH);

int resultadoMes = calendario.get(Calendar.MONTH);

int resultadoAnio = calendario.get(Calendar.YEAR);

String nombreMesResultado = obtenerNombreMes(resultadoMes);

String resultado = String.format("%02d-%s-%04d", resultadoDia, nombreMesResultado, resultadoAnio);

System.out.println("El resultado es: " + resultado);

System.out.print("Desea continuar S/N: ");

op = s.nextLine();

while (!op.matches("[nNsS]")) {

System.out.print("ERROR! Ingrese S/N: ");

op = s.nextLine();

}

} while (!op.matches("[nN]"));

}

public static int obtenerIndiceMes(String mes) {

String[] nombresMeses = {"Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"};

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

if (mes.equalsIgnoreCase(nombresMeses[i])) {

return i;

}

}

return -1;

}

public static String obtenerNombreMes(int indiceMes) {

String[] nombresMeses = {"Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"};

if (indiceMes >= 0 && indiceMes < nombresMeses.length) {

return nombresMeses[indiceMes];

}

return "";

}

public static int compararFechas(Calendar fecha1, Calendar fecha2) {

return fecha1.compareTo(fecha2);

}

public static boolean verificarFormato(String cadena) {

if (!cadena.matches("((0?[1-9]|([1-2][0-9])|30|31)-([A-Z][a-z]+)-(([1][0-9]+)|((20)[0-1][0-9])|((20)[0-2][0-3])))( )(//D)( )(\\d+|(0?[1-9]|([1-2][0-9])|30|31)-([A-Z][a-z]+)-(([1][0-9]+)|((20)[0-2][0-3])))")) {

return false;

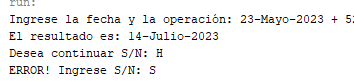
}

return true;

}

}

## Prueba de escritorio:

****

# Ejercicio 7

## Investigación

Para poder hacer el conteo de los caracteres de las palabras y las mismas palabras primero tuvimos que validar el ingreso de datos mediante expresiones regulares, así permitiendo únicamente el ingreso de palabras separadas por un espacio. Tras eso implementamos dos métodos con un lógica similar, lo que se hace es que primero se divide la cadena en una sola palabra y tras eso, dividimos la palabra en los caracteres que la conforman. Una vez dividida la palabra en solo caracteres, mediante un bucle vamos llenando un ArrayList de caracteres que contenga los valores no repetidos ordenados según su orden inicial. Tras eso realizamos dos bucles que vayan comparando el ArrayList con valores no repetidos con los caracteres de la palabra original e ir aumentando en 1 el contador en cada coincidencia, en cada iteración se imprimen los datos. Una vez completado eso, implementamos una lógica similar pero utilizando un ArrayList de String para poder comparar cadenas completas y no únicamente caracteres, con eso podemos obtener la cantidad de repeticiones de cada una de las palabras.

**Código**

import java.util.ArrayList;

import java.util.Scanner;

/\*\*

\*

\* @author Rodrigo Ortiz y Pablo Velez

\*/

public class ConteoPalabras {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

String cadena,op ;

do {

System.out.print("Ingrese la palabra: ");

cadena = s.nextLine();

while (!cadena.matches("([A-Za-z,\\.]+ ?)+")) {

System.out.print("ERROR! La cadena solo puede contener letras y signos de puntuacion: ");

cadena = s.nextLine();

}

contarPalabras(cadena);

System.out.print("Desea continuar S/N: ");

op = s.nextLine();

while (!op.matches("[nNsS]")) {

System.out.print("ERROR! Ingrese S/N: ");

op = s.nextLine();

}

} while (!op.matches("[nN]"));

}

public static void contarPalabras(String s) {

String palabras = s.toLowerCase();

palabras = palabras.replaceAll("[,//.]", "");

String arregloPalabras[] = palabras.split(" ");

for (String arregloPalabra : arregloPalabras) {

contarChar(arregloPalabra);

}

ArrayList<String> datosUnicos = new ArrayList<>();

boolean existente;

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

existente = false;

for (int j = i - 1; j >= 0; j--) {

if (arregloPalabras[i].equals(arregloPalabras[j])) {

existente = true;

break;

}

}

if (!existente) {

datosUnicos.add(arregloPalabras[i]);

}

}

if (datosUnicos.size() != arregloPalabras.length) {

System.out.print("\nExisten las siguientes palabras que se repiten");

System.out.println("\nPalabra\t\tVeces que aparece");

int cont;

for (String datosUnico : datosUnicos) {

cont = 0;

for (String arregloPalabra : arregloPalabras) {

if (arregloPalabra.equals(datosUnico)) {

cont++;

}

}

if (cont >= 2) {

System.out.printf("%s%13d%n", datosUnico, cont);

}

}

} else {

System.out.println("\nNo existen palabras que se repitan");

}

}

public static void contarChar(String s) {

char[] caracteres = s.toCharArray();

ArrayList<Character> datosUnicos = new ArrayList<>();

boolean existente;

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

existente = false;

for (int j = i - 1; j >= 0; j--) {

if (caracteres[i] == caracteres[j]) {

existente = true;

break;

}

}

if (!existente) {

datosUnicos.add(caracteres[i]);

}

}

System.out.println("\nPalabra: " + s);

System.out.println("Caracter\tVeces que aparece");

int cont;

int total = 0;

for (Character datosUnico : datosUnicos) {

cont = 0;

for (char caractere : caracteres) {

if (datosUnico.charValue() == caractere) {

cont++;

}

}

total += cont;

System.out.println(datosUnico.charValue() + "\t\t" + cont);

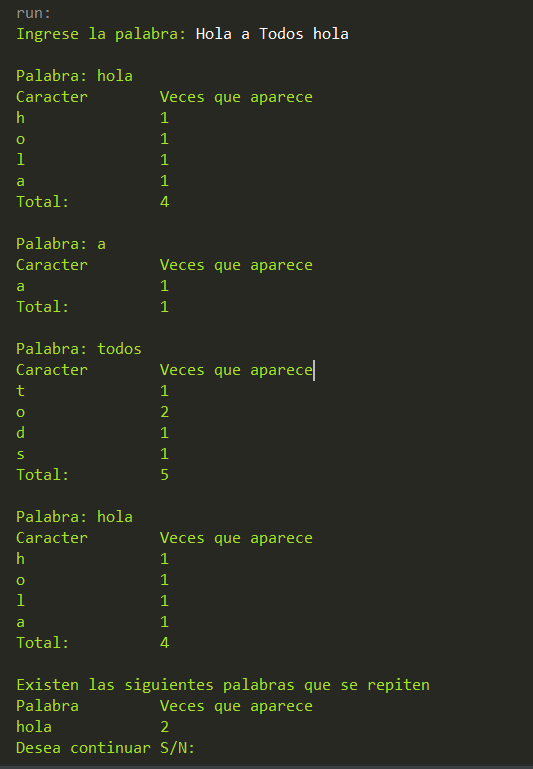
}

System.out.println("Total: \t\t" + total);

}

}

## Pruebas de computador



# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# CONCLUSIONES:

Las colecciones y clases predefinidas desempeñan un papel esencial en el desarrollo de aplicaciones y sistemas informáticos. Su utilización adecuada proporciona soluciones eficientes y estructuradas para el manejo de datos, facilitando el diseño, la implementación y el mantenimiento del código. Dominar estas herramientas permite a los programadores optimizar su tiempo y recursos, ofreciendo soluciones probadas y eficientes para una amplia variedad de problemas.

Al comprender el concepto y la importancia de las colecciones, así como dominar el uso de las clases predefinidas, los desarrolladores adquieren la capacidad de resolver problemas de manera más efectiva y desarrollar aplicaciones más robustas. Además, la aplicación de estas herramientas fomenta la modularidad y reutilización del código, lo que contribuye a un desarrollo más eficiente y escalable.

En conclusión, las colecciones y clases predefinidas son elementos fundamentales en la caja de herramientas de todo programador. Su conocimiento y dominio permiten aprovechar al máximo las capacidades del lenguaje de programación, agilizando el desarrollo y mejorando la calidad de las soluciones. Al seguir aprendiendo y explorando nuevas funcionalidades, los desarrolladores pueden expandir su habilidad para resolver problemas de manera más eficiente y creativa.

# 

# 

# 

# 

# REFERENCIAS BIBLIOGRÁFICAS:

*System (Java Platform SE 7 )*.

<https://docs.oracle.com/javase/7/docs/api/java/lang/System.html>

*Scanner (Java Platform SE 7 )*.

<https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html>

*Math (Java Platform SE 7 )*.

<http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>

*Charset (Java Platform SE 7).*

<https://docs.oracle.com/javase/8/docs/api/java/nio/charset/Charset.html>

*Arrays (Java Platform SE 8 )*.

<https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html>

*ArrayList (Java Platform SE 8 )*. <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>

*Character (Java Platform SE 8 )*. <https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html>

*Cómo puedo convertir números en números romanos - UPAEP - Google Apps*. (s. f.). <http://gapps.upaep.mx/inicio/googledocs/tips-de-google-docs/cmo-puedo-convertir-nmero-en-nmero-romanos#:~:text=Por%20lo%20general%2C%20los%20n%C3%BAmeros,%3D%20-1%20%2B%205>