Universidad Nacional de San Agustín
Escuela Profesional de Ingeniería de Sistemas
Fundamentos de Programación II

Arreglos (Arrays): Búsquedas y Ordenamientos

Nombre: Jhonatan Benjamin Mamani Céspedes **CUI: 20232188**

Link del repositorio GitHub: https://github.com/JBenjamin01/fp2-24b.git

Ejercicio 1:

```java
import java.util.*;
public class Ejercicio1 {
    // Tabla de tiempos de vuelo entre ciudades
    private final static int[][] vuelos = {
        {0, 22, 30, 42, 57}, // AQP
        {23, 0, 15, 25, 58}, // JUL
        {31, 17, 0, 24, 30}, // CUS
        {45, 27, 25, 0, 95}, // TCQ
        {59, 58, 30, 97, 0}  // LIM
    };

    // Nombres de las ciudades en el orden: AQP, JUL, CUS, TCQ, LIM
    private final static String[] ciudades = {"AQP", "JUL", "CUS", "TCQ", "LIM"};

    /**
     * Método para mostrar la tabla de tiempos de vuelo con etiquetas de ciudades.
     */
    public static void mostrarTablaTiemposVuelo() {
        System.out.printf("%5s", ""); // Espacio para la esquina superior izquierda
        for (String ciudad : ciudades) {
            System.out.printf("%5s", ciudad); // Encabezados de las columnas
        }
        System.out.println();

        for (int i = 0; i < vuelos.length; i++) {
            System.out.printf("%5s", ciudades[i]); // Etiquetas de filas (ciudades)
            for (int j = 0; j < vuelos[i].length; j++) {
                System.out.printf("%5d", vuelos[i][j]); // Imprimir tiempos de vuelo
            }
            System.out.println();
        }
    }
```

```java
/**
 * Método para mostrar las opciones de ciudades al usuario.
 */
public static void mostrarCiudades() {
    System.out.println("Ciudades disponibles:");
    for (int i = 0; i < ciudades.length; i++) {
        System.out.printf("%d - %s%n", i + 1, ciudades[i]);
    }
}
```

```java
/**
 * Método para preguntar por una ruta de vuelo de varios tramos.
 * Permite al usuario ingresar múltiples tramos y acumula el tiempo total.
 */
public static void preguntarPorRutaDeVuelo() {
    Scanner sc = new Scanner(System.in);
    int ciudadActual = -1, ciudadDestino;
    int tiempoTotal = 0;
    boolean seguir = true;

    System.out.println("\nIngrese la ciudad de partida:");
    mostrarCiudades();
    System.out.print("Partida: ");
    ciudadActual = sc.nextInt() - 1;

    if (ciudadActual < 0 || ciudadActual >= ciudades.length) {
        System.out.println("Ciudad inválida. Saliendo...");
        return;
    }

    while (seguir) {
        System.out.println("\nIngrese la ciudad de destino (o 0 para finalizar):");
        mostrarCiudades();
        System.out.print("Destino: ");
        ciudadDestino = sc.nextInt() - 1;

        if (ciudadDestino == -1) { // Usuario ingresó 0
            System.out.printf("Ruta finalizada. Tiempo total de vuelo: %d minutos.%n", tiempoTotal);
            seguir = false;
        } else if (ciudadDestino < 0 || ciudadDestino >= ciudades.length) {
            System.out.println("Ciudad inválida. Intente nuevamente.");
        } else {
            int tiempoTramo = vuelos[ciudadActual][ciudadDestino];
            if (tiempoTramo == 0) {
                System.out.printf("No hay vuelo directo de %s a %s.%n", ciudades[ciudadActual], ciudades[ciudadDestino]);
            } else {
                tiempoTotal += tiempoTramo;
                System.out.printf("El tiempo de vuelo de %s a %s es: %d minutos.%n",
                        ciudades[ciudadActual], ciudades[ciudadDestino], tiempoTramo);
                // Actualizar la ciudad actual para el siguiente tramo
                ciudadActual = ciudadDestino;
            }
        }
```

```java
/**
 * Método para calcular el tiempo total de una ruta predefinida: JUL - AQP - LIM - CUS.
 */
public static void calcularRutaPredefinida() {
    String[] ruta = {"JUL", "AQP", "LIM", "CUS"};
    int tiempoTotal = 0;
    System.out.println("\nCalculando la ruta predefinida: JUL - AQP - LIM - CUS");

    for (int i = 0; i < ruta.length - 1; i++) {
        int origen = obtenerIndiceCiudad(ruta[i]);
        int destino = obtenerIndiceCiudad(ruta[i + 1]);

        if (origen == -1 || destino == -1) {
            System.out.printf("Error: Una de las ciudades en la ruta no es válida.%n");
            return;
        }

        int tiempoTramo = vuelos[origen][destino];
        if (tiempoTramo == 0) {
            System.out.printf("No hay vuelo directo de %s a %s.%n", ruta[i], ruta[i + 1]);
            return;
        } else {
            System.out.printf("Tiempo de vuelo de %s a %s: %d minutos.%n", ruta[i], ruta[i + 1], tiempoTramo);
            tiempoTotal += tiempoTramo;
        }
    }

    System.out.printf("Tiempo total de la ruta predefinida: %d minutos.%n", tiempoTotal);
}

```

```java
/**
 * Método auxiliar para obtener el índice de una ciudad en el arreglo 'ciudades'.
 * Retorna -1 si la ciudad no se encuentra.
 *
 * @param nombreCiudad Nombre de la ciudad.
 * @return Índice de la ciudad o -1 si no se encuentra.
 */
public static int obtenerIndiceCiudad(String nombreCiudad) {
    for (int i = 0; i < ciudades.length; i++) {
        if (ciudades[i].equalsIgnoreCase(nombreCiudad)) {
            return i;
        }
    }
    return -1;
}
```

```java
    /**
     * Método principal que ejecuta el programa.
     * Muestra la tabla de tiempos de vuelo, las opciones de ciudades y un menú interactivo.
     */
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        boolean continuar = true;

        // Mostrar la tabla de tiempos de vuelo
        mostrarTablaTiemposVuelo();

        // Mostrar las opciones de ciudades
        mostrarCiudades();

        while (continuar) {
            System.out.println("\nSeleccione una opción:");
            System.out.println("1 - Ingresar una ruta personalizada");
            System.out.println("2 - Calcular una ruta predefinida (JUL - AQP - LIM - CUS)");
            System.out.println("0 - Salir");
            System.out.print("Opción: ");
            int opcion = sc.nextInt();

            switch (opcion) {
                case 1:
                    preguntarPorRutaDeVuelo();
                    break;
                case 2:
                    calcularRutaPredefinida();
                    break;
                case 0:
                    System.out.println("Saliendo del programa...");
                    continuar = false;
                    break;
                default:
                    System.out.println("Opción inválida. Intente nuevamente.");
            }
        }

        sc.close();
    }
}
```