DOCUMENTO PARA ENTREGA DE EJERCICIOS - II CORTE/2024

Michael Steven Giraldo Buitrón

Febrero 2024.

Ingeniería de Sistemas.

Análisis y Estructura de Datos

Fundación Universitaria de Popayán

No se puso el primer punto, porque en este ejercicio aparece el mismo y no es necesario ponerlo dos veces.

1. Ejercicios, Búsqueda y Ordenamiento.

CLASE

PruebaTemperaturas.java | Temperaturas.java

```java
package EjercicioBusqueda.ejercicio;

import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

public class Temperaturas {  2 usages

    double arrTemperaturas[] = new double[5];  13 usages
    Scanner in = new Scanner(System.in);  4 usages
    //Arreglo llenado con datos dados por el usuario
    void llenarConDatosUsuario() {  1 usage
        System.out.println("LLenando con Datos del Usuario...");
        for (int i = 0; i < arrTemperaturas.length; i++) {
            //1.pido el dato al usuario
            System.out.print("Digite la temperatura a guardar: ");
            //2. recibo la temperatura
            double datoEnt = in.nextDouble();
            //3. La guardo en el arreglo
            arrTemperaturas[i] = datoEnt;
        }
        System.out.println("Datos Registrados!!");
```

```java
        //Ordeno el arreglo
        Arrays.sort(arrTemperaturas);
        //Mostrando el arreglo ordenado
        System.out.println("Mostrando datos ordenados");
        for (double temp : arrTemperaturas) {
            System.out.print(temp + " / ");
        }
    }

    public void BusquedaLineal() {  1 usage
        System.out.println("\nIngrese un valor a buscar:");
        double busqueda = in.nextDouble();
        boolean encontrado = false;
        int posicion = -1;

        for (int i = 0; i < arrTemperaturas.length; i++) {
            if (arrTemperaturas[i] == busqueda) {
                encontrado = true;
                posicion = i;
                break;
            }
        }
```

```java
    public void BusquedaLineal() {  1 usage

        if (encontrado) {
            System.out.println("El valor " + busqueda + " fue encontrado en la posicion " + posicion);
        } else {
            System.out.println("El valor " + busqueda + " no fue encontrado en el arreglo");
        }
    }
```

```java
public void BusquedaBinaria() {  1 usage
    System.out.println("\nIngrese un valor a buscar:");
    double busqueda = in.nextDouble();
    int inicio = 0;
    int fin = arrTemperaturas.length - 1;
    int posicion = -1;
    boolean encontrado = false;

    while (inicio <= fin && !encontrado) {
        int medio = (inicio + fin) / 2;

        if (arrTemperaturas[medio] == busqueda) {
            encontrado = true;
            posicion = medio;
        } else if (arrTemperaturas[medio] < busqueda) {
            inicio = medio + 1;
        } else {
            fin = medio - 1;
        }
    }

    if (encontrado) {
        System.out.println("El valor " + busqueda + " fue encontrado en la posicion " + posicion);
    } else {
        System.out.println("El valor " + busqueda + " no fue encontrado en el arreglo");
    }
}
```

```java
//Arreglo llenado con Datos Aleatorios
void llenarConDatosAleatorios(){  1 usage
    System.out.println("LLenando arreglo con Datos Aleatorios");
    //Se usa la clase Random
    Random generador=new Random();
    for (int i = 0; i < arrTemperaturas.length; i++) {
        //uso el random para generar un numero aleatorio
         double numAleatorio = (double)generador.nextInt( bound: 50);
         //guardo el dato en el arreglo
         arrTemperaturas[i]=numAleatorio;
    }
    System.out.println("Arreglado llenado...");
    //ordenar el arreglo
    Arrays.sort(arrTemperaturas);
    System.out.println("Mostrando datos ordenados");
    for (double temp : arrTemperaturas) {
        System.out.print(temp + "-");
    }
```

```java
//Menu
void mostrarMenu() {  1 usage
    int opcion;
    do {
        System.out.println("\nMENU DE OPCIONES");
        System.out.println("1. Guardar Temperaturas dadas por Usuario");
        System.out.println("2. Busqueda Lineal");
        System.out.println("3. Busqueda Binaria");
        System.out.println("4. LLenar con numeros Aleatorios");
        System.out.println("5. Salir");

        System.out.print("\nDigite una opcion: ");
        opcion = in.nextInt();

        switch (opcion) {
            case 1:
               this.llenarConDatosUsuario();
                break;
            case 2:
                this.BusquedaLineal();
                break;
            case 3:
                this.BusquedaBinaria();
                break;
            case 4:
                this.llenarConDatosAleatorios();
                break;
```

PRUEBA

```java
public class PruebaTemperaturas {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //instancio la clase Temperaturas
        Temperaturas obj=new Temperaturas();
        obj.mostrarMenu();
    }

}
```

CÓDIGO EJECUTADO

```
MENU DE OPCIONES
1. Guardar Temperaturas dadas por Usuario
2. Busqueda Lineal
3. Busqueda Binaria
4. LLenar con numeros Aleatorios
5. Salir

Digite una opcion: 1
LLenando con Datos del Usuario...
Digite la temperatura a guardar: 12,4
Digite la temperatura a guardar: 3,1
Digite la temperatura a guardar: 5,6
Digite la temperatura a guardar: 15,4
Digite la temperatura a guardar: 22,1
Datos Registrados!!
Mostrando datos ordenados
3.1 / 5.6 / 12.4 / 15.4 / 22.1 /
```

```
Digite una opcion: 2

Ingrese un valor a buscar:
12,4
El valor 12.4 fue encontrado en la posicion 2

MENU DE OPCIONES
1. Guardar Temperaturas dadas por Usuario
2. Busqueda Lineal
3. Busqueda Binaria
4. LLenar con numeros Aleatorios
5. Salir

Digite una opcion: 3

Ingrese un valor a buscar:
15,4

El valor 15.4 fue encontrado en la posicion 3
```

```
Digite una opcion: 4
LLenando arreglo con Datos Aleatorios
Arreglado llenado...
Mostrando datos ordenados
24.0  /  31.0  /  32.0  /  38.0  /  43.0  /
```

Ejercicio 2:

1. Iteración 1:

- Comparar el primer par de elementos (1000 y 1).
- Como 1000 es mayor que 1, se intercambia estos dos elementos.
- Nuevo arreglo: [1, 1000, 35, 20, -4]

Iteración 2:

- Comparar el segundo par de elementos (1000 y 35).
- Como 1000 es mayor que 35, se intercambia estos dos elementos.
-Arreglo actual: [1, 35, 1000, 20, -4]

Iteración 3:

- Comparar el tercer par de elementos (1000 y 20).
- Como 1000 es mayor que 20, se intercambia estos dos elementos.
- Arreglo actual: [1, 35, 20, 1000, -4]

Iteración 4:

- Comparar el cuarto par de elementos (1000 y -4).
- Como 1000 es mayor que -4, se intercambia estos dos elementos.
- Arreglo actual: [1, 35, 20, -4, 1000]

Iteración 5:

- Comparar el primer par de elementos (1 y 35).
- Como 1 es menor que 35, no se intercambian los elementos.
- Arreglo actual: [1, 35, 20, -4, 1000]

Iteración 6:

- Comparar el segundo par de elementos (35 y 20).
- Como 35 es mayor que 20, se intercambia estos dos elementos.
- Arreglo actual: [1, 20, 35, -4, 1000]

Iteración 7:

- Comparar el tercer par de elementos (35 y -4).
- Como 35 es mayor que -4, se intercambia estos dos elementos.
- Arreglo actual: [1, 20, -4, 35, 1000]

Iteración 8:

- Comparar el cuarto par de elementos (35 y 1000).
- Como 35 es menor que 1000, no se intercambian los elementos.
- Arreglo actual: [1, 20, -4, 35, 1000]

Iteración 9:

- Comparar el primer par de elementos (1 y 20).
- Como 1 es menor que 20, no se intercambian los elementos.
- Arreglo actual: [1, 20, -4, 35, 1000]

Iteración 10:

- Comparar el segundo par de elementos (20 y -4).
- Como 20 es mayor que -4, se intercambia estos dos elementos.
- Arreglo actual: [1, -4, 20, 35, 1000]

Iteración 11:

- Comparar el tercer par de elementos (20 y 35).
- Como 20 es menor que 35, no se intercambian los elementos.
- Arreglo actual: [1, -4, 20, 35, 1000]

Iteración 12:

- Comparar el primer par de elementos (1 y -4).
- Como 1 es mayor que -4, se intercambia estos dos elementos.
- Arreglo actual: [-4, 1, 20, 35, 1000]

Iteración 13:

- Comparar el segundo par de elementos (1 y 20).
- Como 1 es menor que 20, no se intercambian los elementos.
- Arreglo actual: [-4, 1, 20, 35, 1000]

¡Ejercicio completado Exitosamente!

Ejercicio 3:

CLASE

```
public class Control {  2 usages                                    1  31
    double Calificaciones[] = new double[5];  11 usages
    Scanner in = new Scanner(System.in);  2 usages


    public void Menu() {  1 usage
        int option;
        do {
            System.out.println("\n**** MENU DE NOTAS****");
            System.out.println("1. Guardar Notas");
            System.out.println("2. Mostrar Notas");
            System.out.println("3. Ordenar Notas");
            System.out.println("4. Salir");
            System.out.print("Digite una opción: ");
            option = in.nextInt();

            switch (option) {
                case 1:
                    guardarNotas();
                    break;
                case 2:
                    mostrarNotas();
                    break;
                case 3:
                    ordenarNotas();
                    break;
```

```
                case 4:
                    System.out.println("¡Hasta luego!");
                    break;
                default:
                    System.out.println("Opción inválida, intente de nuevo.");
            }
        } while (option != 4);
    }

    public void guardarNotas() {  1 usage
        System.out.println("\nGuardar calificaciones");
        for (int i = 0; i < Calificaciones.length; i++) {
            System.out.print("Ingrese calificación " + (i + 1) + ": ");
            Calificaciones[i] = in.nextDouble();
        }
    }

    public void mostrarNotas() {  1 usage
        System.out.println("\nMostrar calificaciones");
        for (double calificacion : Calificaciones) {
            System.out.print(calificacion + "  /  ");
        }
    }
```

```
    public void ordenarNotas() {  1 usage
        System.out.println("\nOrdenar calificaciones");
        for (int i = 0; i < Calificaciones.length - 1; i++) {
            for (int j = 0; j < Calificaciones.length - 1 - i; j++) {
                if (Calificaciones[j] < Calificaciones[j + 1]) {
                    double ord = Calificaciones[j];
                    Calificaciones[j] = Calificaciones[j + 1];
                    Calificaciones[j + 1] = ord;
                }
            }
        }
    }
}
```

PRUEBA

```
package EjercicioBasico;

public class Prueba {
    public static void main(String[] args){
        Control objeto = new Control();
        objeto.Menu();
    }
}
```

CÓDIGO EJECUTADO

```
**** MENU DE NOTAS****
1. Guardar Notas
2. Mostrar Notas
3. Ordenar Notas
4. Salir
Digite una opción: 1

Guardar calificaciones
Ingrese calificación 1: 2,4
Ingrese calificación 2: 3,5
Ingrese calificación 3: 4,5
Ingrese calificación 4: 6,0
Ingrese calificación 5: 5,0
```

```
Digite una opción: 2

Mostrar calificaciones
2.4  /  3.5  /  4.5  /  6.0  /  5.0  /
**** MENU DE NOTAS****
1. Guardar Notas
2. Mostrar Notas
3. Ordenar Notas
4. Salir
Digite una opción: 3

Ordenar calificaciones

**** MENU DE NOTAS****
1. Guardar Notas
2. Mostrar Notas
3. Ordenar Notas
4. Salir
Digite una opción: 2

Mostrar calificaciones
6.0  /  5.0  /  4.5  /  3.5  /  2.4  /
```

2. Ejercicios Ordenamiento Burbuja.

1. Iteración 1:

- Comparar el primer par de elementos (100 y 3).

- Como 100 es mayor que 3, se intercambia estos dos elementos.

- Nuevo arreglo: [3, 100, 1, 200, 500, 5]

Iteración 2:

- Comparar el segundo par de elementos (100 y 1).

- Como 100 es mayor que 1, se intercambia estos dos elementos.

-Arreglo actual: [3, 1, 100, 200, 500, 5]

Iteración 3:

- Comparar el tercer par de elementos (100 y 200).
- Como 100 es menor que 200, no se intercambian los elementos.
- Arreglo actual: [3, 1, 100, 200, 500, 5]

Iteración 4:

- Comparar el primer par de elementos (3 y 1).
- Como 3 es mayor que 1, se intercambia estos dos elementos.
-Arreglo actual: [1, 3, 100, 200, 500, 5]

Iteración 5:

- Comparar el segundo par de elementos (3 y 100).
- Como 3 es menor que 100, no se intercambian los elementos.
- Arreglo actual: [1, 3, 100, 200, 500, 5]

Iteración 6:

- Comparar el cuarto par de elementos (200 y 500).
- Como 200 es menor que 500, no se intercambia los elementos.
- Arreglo actual: [1, 3, 100, 200, 500, 5]

Iteración 7:

- Comparar el quinto par de elementos (500 y 5).
- Como 500 es mayor que 5, se intercambia estos dos elementos.
- Arreglo actual: [1, 3, 100, 200, 5, 500]

Iteración 8:

- Comparar el cuarto par de elementos (200 y 5).
- Como 200 es mayor que 5, se intercambia estos dos elementos.
- Arreglo actual: [1, 3, 100, 5, 200, 500]

Iteración 9:

- Comparar el tercer par de elementos (100 y 5).
- Como 100 es mayor que 5, se intercambia estos dos elementos.
- Arreglo actual: [1, 3, 5, 100, 200, 500]

Iteración 10:

- Comparar el segundo par de elementos (3 y 5).
- Como 3 es menor que 5, no se intercambia los elementos.
- Arreglo actual: [1, 3, 5, 100, 200, 500]

¡Ejercicio completado Exitosamente!

Ejercicio 2:

CLASE

```java
package Ejercicio2;

import java.util.Scanner;

public class Control {  2 usages
       double promedios[] = new double[4];  15 usages
    Scanner in = new Scanner(System.in);  3 usages

    public void Menux() {  1 usage
            int option;
            do {
                System.out.println("\n**** MENU DE PROMEDIOS ****");
                System.out.println("1. Registrar promedios");
                System.out.println("2. Mostrar promedios");
                System.out.println("3. Ordenar promedios");
                System.out.println("4. Buscar promedio");
                System.out.println("5. Salir");
                System.out.print("Digite una opción: ");
                option = in.nextInt();

                switch (option) {
                    case 1:
                        this.registrarPromedios();
                        break;
                    case 2:
                        this.mostrarPromedios();
                        break;
                    case 3:
                        this.ordenarPromedios();
                        break;
                    case 4:
                        this.BusquedaBinaria();
                        break;
```

```
                case 5:
                    System.out.println("¡Hasta luego!");
                    break;
                default:
                    System.out.println("Opción inválida, intente de nuevo.");
            }
        } while (option != 5);
    }

    public void registrarPromedios() {  1 usage
        System.out.println("\nRegistrar promedios");
        for (int i = 0; i < promedios.length; i++) {
            System.out.print("Ingrese promedio " + (i + 1) + ": ");
            promedios[i] = in.nextDouble();
        }
    }

    public void mostrarPromedios() {  1 usage
        System.out.println("\nMostrar promedios");
        for (double promedio : promedios) {
            System.out.print(promedio + "  /  ");
        }
    }
```

```
    public void ordenarPromedios() {  1 usage
        System.out.println("\nOrdenar promedios");
        for (int i = 0; i < promedios.length - 1; i++) {
            for (int j = 0; j < promedios.length - 1 - i; j++) {
                if (promedios[j] < promedios[j + 1]) {
                    double ord = promedios[j];
                    promedios[j] = promedios[j + 1];
                    promedios[j + 1] = ord;
                }
            }
        }

        System.out.println("\nPromedios Actualizados");
        for (double promedio : promedios) {
            System.out.print(promedio + "  /  ");
        }
    }
```

```java
public void BusquedaBinaria () {  1 usage
    System.out.println("\nIngrese un promedio a buscar:");
    double busqueda = in.nextDouble();
    int inicio = 0;
    int fin = promedios.length - 1;
    int posicion = -1;
    boolean encontrado = false;

    while (inicio <= fin && !encontrado) {
        int medio = (inicio + fin) / 2;

        if (promedios[medio] == busqueda) {
            encontrado = true;
            posicion = medio;
        } else if (promedios[medio] < busqueda) {
            inicio = medio + 1;
        } else {
            fin = medio - 1;
        }
    }

    if (encontrado) {
        System.out.println("El Promedio " + busqueda + " fue encontrado en la posicion " + posicion);
    } else {
        System.out.println("El Promedio " + busqueda + " no fue encontrado en el arreglo");
    }
}
```

PRUEBA

```java
public class Prueba {
    public static void main(String[] args){
        Ejercicio2.Control objeto = new Control();
        objeto.Menux();
    }
}
```

CÓDIGO EJECUTADO

```
**** MENU DE PROMEDIOS ****
1. Registrar promedios
2. Mostrar promedios
3. Ordenar promedios
4. Buscar promedio
5. Salir
Digite una opción: 1

Registrar promedios
Ingrese promedio 1: 2,1
Ingrese promedio 2: 3,1
Ingrese promedio 3: 5,0
Ingrese promedio 4: 2,4
```

```
Digite una opción: 2

Mostrar promedios
2.1  /  3.1  /  5.0  /  2.4  /
**** MENU DE PROMEDIOS ****
1. Registrar promedios
2. Mostrar promedios
3. Ordenar promedios
4. Buscar promedio
5. Salir
Digite una opción: 3

ordenar promedios

Promedios Actualizados
5.0  /  3.1  /  2.4  /  2.1  /
```

```
Digite una opción: 4

Ingrese un promedio a buscar:
3,1
El Promedio 3.1 fue encontrado en la posicion 1
```

EJERCICIO PILAS BÁSICO

- Se Agregó un menú de opciones al ejercicio de Pila Básico
- Se usó otro método agregar para que se apilen 5 datos pero dados por teclado
- ¿Qué pasa si al eliminar la pila está vacía? Solucione él problema

EJERCICIO 1