DOCUMENTO PARA ENTREGA DE EJERCICIOS – I COHORTE/2024

Andrés Felipe Garcés Campo

Fundación Universitaria de Popayán
Ingeniería de Sistemas
Mg. Inge: Franco Urbano

Popayán, Colombia
15/03/2024

Tabla de Contenido

BÚSQUEDAS

Trabajo 1

Introducción Búsqueda Binaria (Taller Busqueda Binaria.pdf)

Punto 1

Escriba los pasos para realizar la búsqueda binaria en los siguientes ejercicios:

a) Clave de Búsqueda: -4

-4	-1	10	11	20	43	50	65	70	75	100	150

b) Clave de Búsqueda: 25

5	4	7	10	15	1	-4	2	10

- Desarrollo
  a) Clave de Búsqueda: -4

-4	-1	10	11	20	43	50	65	70	75	100	150
i=0						m=6					j=11

$$m = \frac{(i+j)}{2} => \frac{(0+11)}{2} = 5.5$$

Comparo:

Clave a Buscar	con	Valor Medio	
-4	>	50	NO
-4	<	50	SI

Desplazo j,

$j = m - 1 => 6 - 1 = 5$

-4	-1	10	11	20	43
i=0			m=3		j=5

$$m = \frac{(0+5)}{2} = 2.5$$

Comparo:

Clave a Buscar	con	Valor Medio	
-4	>	11	NO
-4	<	11	SI

Desplazo j,

$j = 3 - 1 = 2$

-4	-1	10
i=0	m=1	j=2

$$m = \frac{(0+5)}{2} = 2.5$$

Comparo:

Clave a Buscar	con	Valor Medio	
-4	>	-1	NO
-4	<	-1	SI

Desplazo j,
$j = 1 - 1 = 0$

-4

i
j → 0
m

$$m = \frac{(0+0)}{2} = 1$$

Comparo:

Clave a Buscar	con	Valor Medio	
-4	>	-4	NO
-4	<	-4	NO
-4	=	-4	SI

---

b) Clave de Búsqueda: 25

5	4	7	10	15	1	-4	2	10
i=0				m=4				j=8

$$m = \frac{(0+8)}{2} = 4$$

Comparo:

Clave a Buscar	con	Valor Medio	
25	>	15	SI
25	<	15	NO

Desplazo i,
$i = 4 + 1 = 5$

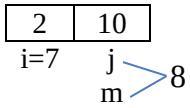
1	-4	2	10
i=5	m=6		j=8

$$m = \frac{(5+8)}{2} = 6.5$$

Comparo:

Clave a Buscar	con	Valor Medio	
25	>	-4	SI
25	<	-4	NO

Desplazo i,
$i = 6 + 1 = 7$



$$m = \frac{(7+8)}{2} = 7.5$$

Comparo:

Clave a Buscar	con	Valor Medio	
25	>	10	SI
25	<	10	NO

Desplazo i,
$i = 8 + 1 = 9$

No hay posición 9, por lo tanto, el Valor a Buscar no se encuentra en el Arreglo.

Trabajo 2

Introducción Búsqueda Binaria (Taller Busqueda Binaria.pdf)

Punto 2

Realice un programa que contenga un Menú de Opciones para:

a) Llenar un Arreglo con 5 Temperaturas de una Ciudad, mostrando el Arreglo con los datos ordenados
b) Realizar una Búsqueda Lineal para Buscar una Temperatura
c) Realizar una Búsqueda eficiente para Buscar una Temperatura (Búsqueda Binaria)
d) Llenar un Arreglo con 5 Temperaturas aleatorias.

- Desarrollo

Clase Temperatura:

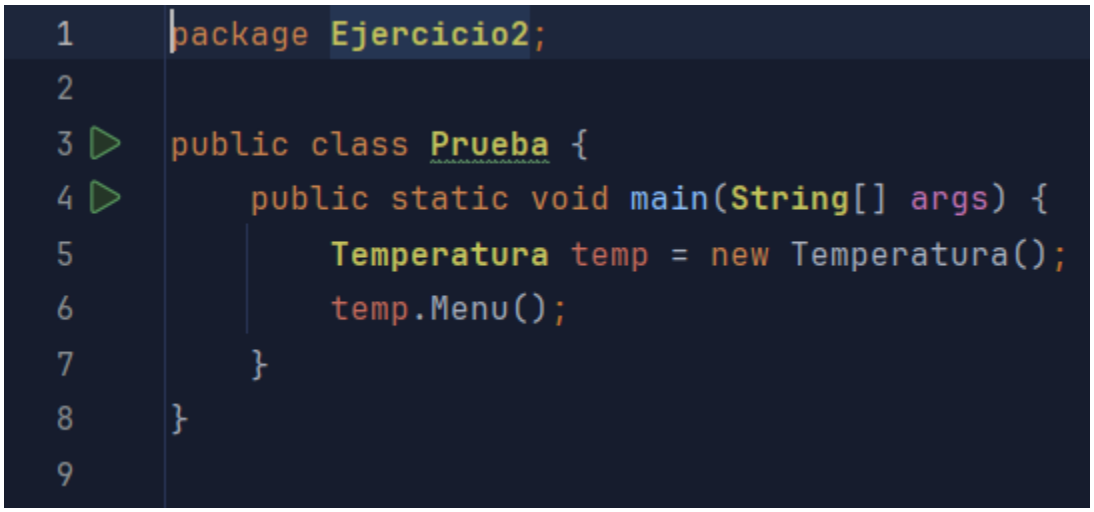
```java
package Ejercicio2;

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

public class Temperatura {  2 usages
    double Temperaturas[] = new double[5];  13 usages

    Scanner scan = new Scanner(System.in);  4 usages

    //Método para registrar las temperaturas
    void llenarConDatosUsuario() {  1 usage
        System.out.println("Temperaturas de la Ciudad Popayán");
        for (int i = 0; i < Temperaturas.length; i++) {

            //1. Pido la temperatura al usuario
            System.out.print("Digite la temperatura " + (i + 1) + " a guardar: ");

            //2. Recibo la temperatura
            double datoEnt = scan.nextDouble();

            //3. La guardo en el arreglo
            Temperaturas[i] = datoEnt;
        }
        System.out.println("Temperaturas Registradas!!");

        //Ordeno el arreglo
        Arrays.sort(Temperaturas);

        //Mostrando el arreglo ordenado
        System.out.println("");
```

```
33          System.out.println("Mostrando temperaturas ordenadas");
34          for (double temp : Temperaturas) {
35              System.out.print(temp + " / ");
36          }
37      }
38
39      //Método Llenado con temperaturas Aleatorios
40      void LlenarConTempAleatorios() {  1 usage
41          System.out.println("LLenando Arreglo con temperaturas Aleatorios");
42
43          //Se usa la clase Random
44          Random generador = new Random();
45          for (int i = 0; i < Temperaturas.length; i++) {
46
47              //Uso el random para generar un número aleatorio
48              double numAleatorio = (double)generador.nextDouble( bound: 50);
49
50              //Guardo el dato en el arreglo
51              Temperaturas[i] = numAleatorio;
52          }
53
54          //Ordenar el arreglo
55          Arrays.sort(Temperaturas);
56          System.out.println("");
57          System.out.println("Mostrando temperaturas ordenadas y aleatorizadas");
58          for (double temp : Temperaturas) {
59              System.out.print(temp + " / ");
60          }
61      }
62
```

```
63      //Método para Buscar Temperatura - Búsqueda Binaria
64      void buscarBinaria(){  1 usage
65          System.out.print("Digite la temperatura a Buscar: ");
66          double TempBuscado = scan.nextDouble();
67
68          //Llamó al Método del algortimo de Búsqueda Binaria
69          double datoRet = this.busquedaBinaria(TempBuscado);
70
71          if(datoRet == -1)
72              System.out.println("Temperatura no Encontrado!!");
73          else
74              System.out.println("Encontrada - Posicion: " + datoRet);
75
76      }
77
78      //Algortimo de Búsqueda Binaria
79      public int busquedaBinaria(double elementoBusqueda) {  1 usage
80          int inferior = 0; //Extremo inferior del área de búsqueda
81          int superior = Temperaturas.length - 1; //Extremo superior del área de búsqueda
82          int medio = (inferior + superior + 1) / 2; //Elemento medio
83          int ubicacion = -1; //Devuelve el valor; -1 si no lo encontró
84
85          do {
86              if (elementoBusqueda == Temperaturas[medio]) {
87                  ubicacion = medio; //La ubicación es el elemento medio actual
88              } //El elemento medio es demasiado alto
89              else if (elementoBusqueda < Temperaturas[medio]) {
90                  superior = medio - 1; //Elimina la mitad superior
91              } else //El elemento medio es demasiado bajo
92              {
93                  inferior = medio + 1; //Elimina la mitad inferior
```

```
            }
            medio = (inferior + superior + 1) / 2; //Recalcula el elemento medio
        } while ((inferior <= superior) && (ubicacion == -1));

        return ubicacion; //Devuelve
    }

    //Método para Buscar la temperatura - Búsqueda Lineal
    void BuscarLineal(){  1 usage
        System.out.print("Digite la temperatura a Buscar: ");
        double datoBuscar = scan.nextDouble();

        boolean bandera = false; //Variable Auxiliar
        for (int i = 0; i < Temperaturas.length; i++) {

            //Comparo
            if (datoBuscar == Temperaturas[i]) {
                bandera = true;
            }
        }

        //Revisó la bandera
        if (bandera == true)
            System.out.println("Temperatura encontrada: " + datoBuscar);
        else
            System.out.println("Temperatura no encontrado");
    }

    //Menú de Opciones
    void Menu() {  1 usage
        int opcion;
        do {
            System.out.println("\nMENU DE OPCIONES");
            System.out.println("1. Guardar una Temperatura");
            System.out.println("2. Buscar por Búsqueda Binaria");
            System.out.println("3. Buscar por Búsqueda Lineal");
            System.out.println("4. Llenar con Temperaturas Aleatorias");
            System.out.println("5. Salir");
            System.out.print("\nDigite una opción: ");
            opcion = scan.nextInt();

            switch (opcion) {
                case 1:
                    this.llenarConDatosUsuario();
                    break;
                case 2:
                    this.buscarBinaria();
                    break;
                case 3:
                    this.BuscarLineal();
                    break;
                case 4:
                    this.LlenarConTempAleatorios();
                    break;
            }
        } while (opcion !=5);
    }
}
```

Clase Prueba:



```java
package Ejercicio2;

public class Prueba {
    public static void main(String[] args) {
        Temperatura temp = new Temperatura();
        temp.Menu();
    }
}

```

Código Ejecutado:

a) Registrar las Temperaturas

```
MENU DE OPCIONES
1. Guardar una Temperatura
2. Buscar por Búsqueda Binaria
3. Buscar por Búsqueda Lineal
4. Llenar con Temperaturas Aleatorias
5. Salir

Digite una opción: 1
Temperaturas de la Ciudad Popayán
Digite la temperatura 1 a guardar: 9,5
Digite la temperatura 2 a guardar: 2,3
Digite la temperatura 3 a guardar: 54,3
Digite la temperatura 4 a guardar: 1,0
Digite la temperatura 5 a guardar: 7,3
Temperaturas Registradas!!

Mostrando temperaturas ordenados
1.0 / 2.3 / 7.3 / 9.5 / 54.3 /
```

b) Buscar una Temperatura (Búsqueda Binaria)

- Cuando si está registrada

```
MENU DE OPCIONES
1. Guardar una Temperatura
2. Buscar por Búsqueda Binaria
3. Buscar por Búsqueda Lineal
4. Llenar con Temperaturas Aleatorias
5. Salir

Digite una opción: 2
Digite la temperatura a Buscar: 9,5
Encontrada - Posicion: 3.0
```

- Cuando no está registrada

```
MENU DE OPCIONES
1. Guardar una Temperatura
2. Buscar por Búsqueda Binaria
3. Buscar por Búsqueda Lineal
4. Llenar con Temperaturas Aleatorias
5. Salir

Digite una opción: 2
Digite la temperatura a Buscar: 10
Temperatura no Encontrado!!
```

c) Buscar una Temperatura (Búsqueda Lineal)

- Cuando si está registrada

```
MENU DE OPCIONES
1. Guardar una Temperatura
2. Buscar por Búsqueda Binaria
3. Buscar por Búsqueda Lineal
4. Llenar con Temperaturas Aleatorias
5. Salir

Digite una opción: 3
Digite la temperatura a Buscar: 1,0
Temperatura encontrada: 1.0
```

- Cuando no está registrada

```
MENU DE OPCIONES
1. Guardar una Temperatura
2. Buscar por Búsqueda Binaria
3. Buscar por Búsqueda Lineal
4. Llenar con Temperaturas Aleatorias
5. Salir

Digite una opción: 3
Digite la temperatura a Buscar: 99
Temperatura no encontrado
```

d) Temperaturas Aleatorizadas

```
MENU DE OPCIONES
1. Guardar una Temperatura
2. Buscar por Búsqueda Binaria
3. Buscar por Búsqueda Lineal
4. Llenar con Temperaturas Aleatorias
5. Salir

Digite una opción: 4
LLenando Arreglo con temperaturas Aleatorios

Mostrando temperaturas ordenadas y aleatorizadas
0.0 / 9.0 / 30.0 / 30.0 / 48.0 /
```

e) Buscar una Temperatura (Búsqueda Binaria)

- Cuando si está en el Arreglo

```
MENU DE OPCIONES
1. Guardar una Temperatura
2. Buscar por Búsqueda Binaria
3. Buscar por Búsqueda Lineal
4. Llenar con Temperaturas Aleatorias
5. Salir

Digite una opción: 2
Digite la temperatura a Buscar: 0
Encontrada - Posicion: 0.0
```

- Cuando no está en el Arreglo

```
MENU DE OPCIONES
1. Guardar una Temperatura
2. Buscar por Búsqueda Binaria
3. Buscar por Búsqueda Lineal
4. Llenar con Temperaturas Aleatorias
5. Salir

Digite una opción: 2
Digite la temperatura a Buscar: 1000

Temperatura no Encontrado!!
```

f) Buscar una Temperatura (Búsqueda Lineal)

- Cuando si está en el Arreglo

```
MENU DE OPCIONES
1. Guardar una Temperatura
2. Buscar por Búsqueda Binaria
3. Buscar por Búsqueda Lineal
4. Llenar con Temperaturas Aleatorias
5. Salir

Digite una opción: 3
Digite la temperatura a Buscar: 48
Temperatura encontrada: 48.0
```

- Cuando no está en el Arreglo

```
MENU DE OPCIONES
1. Guardar una Temperatura
2. Buscar por Búsqueda Binaria
3. Buscar por Búsqueda Lineal
4. Llenar con Temperaturas Aleatorias
5. Salir

Digite una opción: 3
Digite la temperatura a Buscar: 12

Temperatura no encontrado
```

Trabajo 3

Búsquedas (Ejercicios Busqueda-Ordenamiento.pdf)

Punto 1

Realice un programa que contenga un Menú de Opciones para:

a) Llenar un Arreglo con 5 Temperaturas de una Ciudad, mostrando el Arreglo con los datos ordenados
b) Realizar una Búsqueda Lineal para Buscar una Temperatura
c) Realizar una Búsqueda eficiente para Buscar una Temperatura (Búsqueda Binaria)
d) Llenar un Arreglo con 5 Temperaturas aleatorias.

- Desarrollo

El Trabajo 3 es el mismo que el Trabajo 2

ORDENAMIENTOS

Trabajo 4

Ordenamiento por Burbuja (Ejercicios Busqueda-Ordenamiento.pdf)

Punto 2

En un documento de texto escriba los pasos que seguiría para ordenar el siguiente arreglo utilizando el método de Burbuja

a)

1000	1	35	20	-4

- Desarrollo

1000	1	35	20	-4

Fórmula: Si NúmeroActual > NúmeroSiguiente – Intercambio

El Primer Recorrido:

1000, **1**, 35, 20, -4 → Comparar 1000 y 1 – (1° y 2°) No están en orden – Intercambio
1, **1000**, 35, 20, -4 → Seguimos
1, **1000**, **35**, 20, -4 → Comparar 1000 y 35 – (2° y 3°) No están en orden – Intercambio
1, **35**, **1000**, 20, -4 → Seguimos
1, 35, **1000**, **20**, -4 → Comparar 1000 y 20 – (3° y 4°) No están en orden – Intercambio
1, 35, **20**, **1000**, -4 → Seguimos
1, 35, 20, **1000**, **-4** → Comparar 1000 y -4 – (4° y 5°) No están en orden – Intercambio
1, 35, 20, **-4**, **1000** → Ya hemos terminado está pasada
1, 35, 20, -4, **1000** → El 5° elemento ya está en su sitio

La Segundo Recorrida:

1, **35**, 20, -4, 1000 → Comparar 1 y 35 – (1° y 2°) Están en orden – Seguimos
1, **35**, **20**, -4, 1000 → Comparar 35 y 20 – (2° y 3°) No están en orden – Intercambio
1, **20**, **35**, -4, 1000 → Seguimos
1, 20, **35**, **-4**, 1000 → Comparar 35 y -4 – (3° y 4°) No están en orden – Intercambio
1, 20, **-4**, **35**, 1000 → Seguimos
1, 20, -4, **35**, **1000** → Comparar 35 y 1000 – (4° y 5°) Están en orden – Pasada terminada
1, 20, -4, **35**, **1000** → El 4° elemento ya está en su sitio

El Tercer Recorrido:
1, **20**, -4, 35, 1000 → Comparar 1 y 20 – (1° y 2°) Están en orden – Seguimos
1, **20**, **-4**, 35, 1000 → Comparar 20 y -4 – (2° y 3°) No están en orden – Intercambio
1, **-4**, **20**, 35, 1000 → Seguimos
1, -4, **20**, **35**, 1000 → Comparar 20 y 35 – (3° y 4°) Si están en orden – Pasada terminada
1, -4, **20**, **35**, **1000** → El 3° elemento ya está en su sitio

El Cuarto Recorrido:
1, **-4**, 20, 35, 1000 → Comparar 1 y -4 – (1° y 2°) No están en orden – Intercambio
-4, **1**, **20**, **35**, **1000** → El 1° y 2° elementos ya están en su sitio

Trabajo 5

Ordenamiento por Burbuja (Ejercicios Busqueda-Ordenamiento.pdf)

Punto 1

Realice un programa que permita guardar 5 Calificaciones y luego las ordene de mayor a menor. Debe usar un Menú que contenga las opciones de:

a) Guardar Calificaciones
b) Mostrar Calificaciones
c) Ordenar Calificaciones

- Desarrollo

Clase Calificacion:

```java
package ejercicioEscuela;

import java.util.Scanner;

public class Califacion {  2 usages

    double calificaciones[] = new double[5];  5 usages

    Scanner scan = new Scanner(System.in);  2 usages

    void registrarCalificaciones() {  1 usage
        double calificacion;
        for (int i = 0; i < calificaciones.length; i++) {
            System.out.print("Registre la Calificaión " + (i + 1) + ": ");
            calificacion = scan.nextDouble();
            calificaciones[i] = calificacion;
        }
        System.out.println("Calificaciones registradas exitosamente!");
    }

    void listarCalificaciones() {  1 usage
        System.out.println("Listado de Calificaciones");
        for (int i = 0; i < calificaciones.length; i++) {
            System.out.print(calificaciones[i] + " / ");
        }
    }

    //Algoritmo de Ordenamiento por Burbuja
    void ordenarporBurbuja(double arreglo[]) {  1 usage
        double n = arreglo.length;
        //Variable usada pa
        double aux = 0.0;
```

```java
        for (int i = 0; i < n - 1; i++) {
            /*Este ciclo se encraga de llevar el mayor al final del arreglo
             * Va desde 0 hasta se haya ido ornando el mayor
             * */
            for (int j = 0; j < n - i - 1; j++) {
                //En caso de que se cumpla la condición se intercambia la posición
                if(arreglo[j] > arreglo[j + 1]){
                    aux = arreglo[j];
                    arreglo[j] = arreglo[j+  1];
                    arreglo[j + 1] = aux;
                }
            }
        }

        //Listar el Arreglo ordenado
        System.out.println("\n****Calificaciones ordenadas por Método Burbuja****");
        for (double i : arreglo) {
            System.out.print(i + " / ");
        }
    }

    //Menú de Opciones
    void menu(){  1 usage
        int opcion;
        do {
            System.out.println("""
                    \n****MENÚ DE OPCIONES****

                    1. Registrar Calificaciones
                    2. Listar Calificaciones
                    3. Ordenar Calificaciones por Burbuja
                    4. Salir del Programa""");
            System.out.println("");
            System.out.print("Digite una Opción: ");
            opcion = scan.nextInt();
            switch(opcion) {
                case 1:
                    this.registrarCalificaciones();
                    break;
                case 2:
                    this.listarCalificaciones();
                    break;
                case 3:
                    this.ordenarporBurbuja(calificaciones);
                    System.out.println("");
                    System.out.println("Calificaciones ordenadas");
                    break;
            }
        } while(opcion != 4);
    }
}
```

Clase Prueba:

```java
package ejercicioEscuela;

public class Prueba {
    public static void main(String[] args) {
        Califacion obj = new Califacion();
        obj.menu();
    }
}
```

Código Ejecutado:

a) Registrar las Calificaciones

```
****MENÚ DE OPCIONES****

1. Registrar Calificaciones
2. Listar Calificaciones
3. Ordenar Notas por Burbuja
4. Salir del Programa

Digite una Opción: 1
Registre la Calificaión 1: 4,7
Registre la Calificaión 2: 5,0
Registre la Calificaión 3: 1,9
Registre la Calificaión 4: 3,5
Registre la Calificaión 5: 2,2
Calificaciones registradas exitosamente!
```

b) Listar las Calificaciones

```
****MENÚ DE OPCIONES****

1. Registrar Calificaciones
2. Listar Calificaciones
3. Ordenar Notas por Burbuja
4. Salir del Programa

Digite una Opción: 2
Listado de Calificaciones
4.7 / 5.0 / 1.9 / 3.5 / 2.2 /
```

c) Ordenar las Calificaciones por Burbuja

```
****MENÚ DE OPCIONES****

1. Registrar Calificaciones
2. Listar Calificaciones
3. Ordenar Notas por Burbuja
4. Salir del Programa

Digite una Opción: 3

****Calificaciones ordenadas por Método Burbuja****
1.9 / 2.2 / 3.5 / 4.7 / 5.0 /
Calificaciones ordenadas
```

Trabajo 6

Ordenamiento por Burbuja (5-Ejercicio Propuesto Burbuja.pdf)

Punto 1

En un documento de texto escriba los pasos que seguiría para ordenar el siguiente arreglo utilizando el método de Burbuja

a)

100	3	1	200	500	5

- Desarrollo

100	3	1	200	500	5

Fórmula: Si NúmeroActual > NúmeroSiguiente – Intercambio

El Primer Recorrido:

100, **3**, 1, 200, 500, 5 → Comparar 100 y 3 – (1° y 2°) No están en orden – Intercambio
3, **100**, 1, 200, 500, 5 → Seguimos
3, **100**, **1**, 200, 500, 5 → Comparar 100 y 1 – (2° y 3°) No están en orden – Intercambio
3, **1**, **100**, 200, 500, 5 → Seguimos
3, 1, **100**, **200**, 500, 5 → Comparar 100 y 200 – (3° y 4°) Están en orden – Seguimos
3, 1, 100, **200**, **500**, 5 → Comparar 200 y -500 – (4° y 5°) Están en orden – Seguimos
3, 1, 100, 200, **500**, **5** → Compara 500 y 5 – (5° y 6°) No están en orden – Intercambio
3, 1, 100, 200, 5, **500** → El 6° elemento ya está en su sitio

La Segundo Recorrida:

3, **1**, 100, 200, 5, 500 → Comparar 3 y 1 – (1° y 2°) No están en orden – Intercambio
1, **3**, 100, 200, 5, 500 → Seguimos
1, **3**, **100**, 200, 5, 500 → Comparar 3 y 100 – (2° y 3°) Están en orden – Seguimos
1, 3, **100**, **200**, 5, 500 → Comparar 100 y 200 – (3° y 4°) Están en orden – Seguimos
1, 3, 100, **200**, **5**, 500 → Comparar 200 y 5 – (4° y 5°) No están en orden – Intercambio
1, 3, 100, **5**, **200**, 500 → Seguimos
1, 3, 100, 5, **200**, **500** → Comparar 200 y 500 – (5° y 6°) Están en orden – Pasada terminada
1, 3, 100, 5, **200**, **500** → El 5° elemento ya está en su sitio

El Tercer Recorrido:
1, **3**, 100, 5, 200, 500 → Comparar 1 y 3 – (1° y 2°) Están en orden – Seguimos
1, **3**, **100**, 5, 200, 500 → Comparar 3 y 100 – (2° y 3°) Están en orden – Seguimos
1, 3, **100**, **5**, 200, 500 → Comparar 100 y 5 – (3° y 4°) No están en orden – Intercambio
1, 3, **5**, **100**, 200, 500 → Seguimos
1, 3, 5, **100**, **200**, 500 → Comparar 100 y 200 – (4° y 5°) Están en orden – Pasada terminada
1, 3, 5, **100**, **200**, **500** → El 4° elemento ya está en su sitio

El Tercer Recorrido:
1, **3**, 5, 100, 200, 500 → Comparar 1 y 3 – (1° y 2°) Están en orden – Seguimos
1, **3**, **5**, 100, 200, 500 → Comparar 3 y 5 – (2° y 3°) Están en orden – Pasada terminada
1, 3, **5**, **100**, **200**, **500** → El 3° elemento ya está en su sitio

El Cuarto Recorrido:
1, **3**, 5, 100, 200, 500 → Comparar 1 y 3 – (1° y 2°) Están en orden – Pasada terminada
1, **3**, **5**, **100**, **200**, **500** → El 1° y 2° elementos ya están en su sitio

La Tercer Recorrida:
1, **20**, -4, 35, 1000 → Comparar 1 y 20 – (1° y 2°) Están en orden – Seguimos
1, **20**, **-4**, 35, 1000 → Comparar 20 y -4 – (2° y 3°) No están en orden – Intercambio
1, **-4**, **20**, 35, 1000 → Seguimos
1, -4, **20**, **35**, 1000 → Comparar 20 y 35 – (3° y 4°) Si están en orden – Pasada terminada
1, -4, **20**, **35**, **1000** → El 3° elemento ya está en su sitio

La Cuarta Recorrida:
1, **-4**, 20, 35, 1000 → Comparar 1 y -4 – (1° y 2°) No están en orden – Intercambio
-4, **1**, **20**, **35**, **1000** → El 1° y 2° elementos ya están en su sitio

Trabajo 7

Ordenamiento por Burbuja (5-Ejercicio Propuesto Burbuja.pdf)

Punto 2

Realice un programa que permita ordenar los 4 promedios de un Estudiante, los cuáles son digitados por teclado. Utilice un Menú de Opciones para registrar los promedios, Mostrarlos y Ordenarlos. Ofrezca una opción para Buscar un promedio (Use Búsqueda Lineal y traté de hacerla lo más eficiente)

- Desarrollo

Clase Estudiante:

```java
package ejercicioEstudiante;

import java.util.Scanner;

public class Estudiante {  2 usages

    double promedios[] = new double[4];  7 usages

    Scanner scan = new Scanner(System.in);  3 usages

    void registrarPromedios() {  1 usage
        double promedio;
        for (int i = 0; i < promedios.length; i++) {
            System.out.print("Registre el Promedio " + (i + 1) + ": ");
            promedio = scan.nextDouble();
            promedios[i] = promedio;
        }
        System.out.println("Promedios registrados exitosamente!");
    }

    void listarPromedios() {  1 usage
        System.out.println("Listado de Promedios");
        for (int i = 0; i < promedios.length; i++) {
            System.out.print(promedios[i] + " / ");
        }
    }

    //Algoritmo de Ordenamiento por Burbuja
    void ordenarporBurbuja(double arreglo[]) {  1 usage
        double n = arreglo.length;
        //Variable usada pa
        double aux = 0.0;
```

```java
        for (int i = 0; i < n - 1; i++) {
            /*Este ciclo se encraga de llevar el mayor al final del arreglo
             * Va desde 0 hasta se haya ido ornando el mayor
             * */
            for (int j = 0; j < n - i - 1; j++) {
                //En caso de que se cumpla la condición se intercambia la posición
                if (arreglo[j] > arreglo[j + 1]) {
                    aux = arreglo[j];
                    arreglo[j] = arreglo[j + 1];
                    arreglo[j + 1] = aux;
                }
            }
        }

        //Listar el Arreglo ordenado
        System.out.println("\n****Promedios ordenados por Método Burbuja****");
        for (double i : arreglo) {
            System.out.print(i + " / ");
        }
    }

    //Método para Buscar el Promedio - Búsqueda Lineal
    void BuscarLineal() {  1 usage
        System.out.print("Digite el Promedio a Buscar: ");
        double datoBuscar = scan.nextDouble();

        boolean bandera = false; //Variable Auxiliar
        for (int i = 0; i < promedios.length; i++) {

            //Comparo
            if (datoBuscar == promedios[i]) {
                bandera = true;
            }
        }

        //Revisó la bandera
        if (bandera == true)
            System.out.println("Promedio encontrado: " + datoBuscar);
        else
            System.out.println("Promedio no encontrado");
    }

    //Menú de Opciones
    void menu() {  1 usage
        int opcion;
        do {
            System.out.println("""
                    \n****MENÚ DE OPCIONES****

                    1. Registrar Promedios
                    2. Listar Promedios
                    3. Ordenar Promedios por Burbuja
                    4. Buscar Promedio - Búsqueda Lineal
                    5. Salir del Programa""");
            System.out.println("");
            System.out.print("Digite una Opción: ");
            opcion = scan.nextInt();
            switch (opcion) {
                case 1:
```

```
                        this.registrarPromedios();
                        break;
                    case 2:
                        this.listarPromedios();
                        break;
                    case 3:
                        this.ordenarporBurbuja(promedios);
                        System.out.println("");
                        System.out.println("Promedios ordenadas");
                        break;
                    case 4:
                        this.BuscarLineal();
                        break;
                }
            } while (opcion == 5);
    }
}
```

Clase Prueba:

```java
package ejercicioEstudiante;

public class Prueba {
    public static void main(String[] args) {
        Estudiante obj = new Estudiante();
        obj.menu();
    }
}
```

Código Ejecutado:

a) Registrar las Calificaciones

```
****MENÚ DE OPCIONES****

1. Registrar Promedios
2. Listar Promedios
3. Ordenar Promedios por Burbuja
4. Buscar Promedio - Búsqueda Lineal
5. Salir del Programa

Digite una Opción: 1
Registre el Promedio 1: 5,0
Registre el Promedio 2: 1,6
Registre el Promedio 3: 2,7
Registre el Promedio 4: 4,2
Promedios registrados exitosamente!
```

b) Listar Promedios

```
****MENÚ DE OPCIONES****

1. Registrar Promedios
2. Listar Promedios
3. Ordenar Promedios por Burbuja
4. Buscar Promedio - Búsqueda Lineal
5. Salir del Programa

Digite una Opción: 2
Listado de Promedios
5.0 / 1.6 / 2.7 / 4.2 /
```

c) Ordenar Promedios por Burbuja

```
****MENÚ DE OPCIONES****

1. Registrar Promedios
2. Listar Promedios
3. Ordenar Promedios por Burbuja
4. Buscar Promedio - Búsqueda Lineal
5. Salir del Programa

Digite una Opción: 3

****Promedios ordenados por Método Burbuja****
1.6 / 2.7 / 4.2 / 5.0 /
Promedios ordenadas
```

d) Buscar Promedio (Búsqueda Lineal)

- Cuando si está registrado

```
****MENÚ DE OPCIONES****

1. Registrar Promedios
2. Listar Promedios
3. Ordenar Promedios por Burbuja
4. Buscar Promedio - Búsqueda Lineal
5. Salir del Programa

Digite una Opción: 4
Digite el Promedio a Buscar: 2,7
Promedio encontrado: 2.7
```

- Cuando no está registrado

```
****MENÚ DE OPCIONES****

1. Registrar Promedios
2. Listar Promedios
3. Ordenar Promedios por Burbuja
4. Buscar Promedio - Búsqueda Lineal
5. Salir del Programa

Digite una Opción: 4
Digite el Promedio a Buscar: 3,0
Promedio no encontrado
```

PILAS&OBJETOS

Trabajo 8

Ejercicio Pilas Básico

Punto 1

- **a)** Agréguele un menú de opciones al ejercicio de Pila Básico
- **b)** Use otro método agregar para que se apilen 5 datos, pero dados por teclado
- **c)** ¿Qué pasa si al eliminar la pila está vacía? Solucione el problema

- Desarrollo

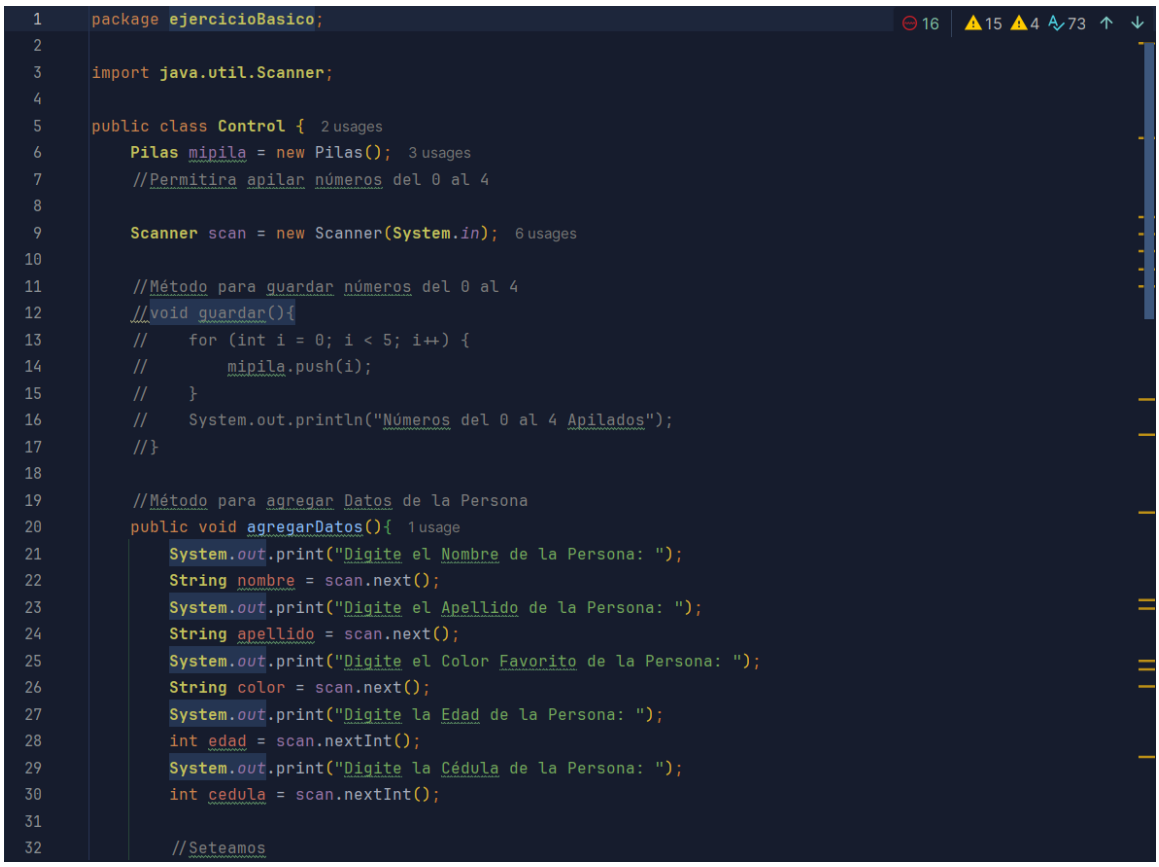
Clase Elemento:

```java
package ejercicioBasico;

6 usages
public class Elemento {

    2 usages
    String nombre, apellido, color;
    2 usages
    int edad, cedula;

    1 usage
    public String getNombre() { return nombre; }

    1 usage
    public void setNombre(String nombre) { this.nombre = nombre; }

    1 usage
    public String getApellido() { return apellido; }

    1 usage
    public void setApellido(String apellido) { this.apellido = apellido; }

    1 usage
    public String getColor() { return color; }

    1 usage
    public void setColor(String color) { this.color = color; }

    1 usage
    public int getEdad() { return edad; }

    1 usage
    public void setEdad(int edad) { this.edad = edad; }

    2 usages
    public int getCedula() { return cedula; }

    1 usage
    public void setCedula(int cedula) { this.cedula = cedula; }
}
```

Clase Pilas:

```java
package ejercicioBasico;

import java.util.LinkedList;

2 usages
public class Pilas {

    6 usages
    public LinkedList lista = new LinkedList();

    1 usage
    public void push(Object o) { lista.addFirst(o); }

    no usages
    public Object top() {  //Obtener el elemento que esta en la parte

        //Superior de la pila
        return lista.getFirst();
    }

    //public Object pop() { //Desapilar. Quitar un elemento de la pila.
    //    return lista.removeFirst();
    //}

    1 usage
    public Object pop() {
        if (lista.isEmpty()) {
            System.out.println("");
            return "****Lista Vacía****";
        } else {
            Elemento el = (Elemento) lista.removeFirst();
            //Casteo, de que un Object se convierta en algo

            return "Persona Desapilada - Cedula: " + el.getCedula();
        }
    }

    no usages
    public int size() { return lista.size(); }

    //public Object listar(int i) {
    //    return lista.get(i);
    //}

    1 usage
    void ListarTodo() {
        System.out.println("****Listando  Personas de la Pila****");
        System.out.println("( Cabecera )");
        for (Object o : lista) {
            Elemento elRecuperado = (Elemento) o;
            System.out.println("Nombre: " + elRecuperado.getNombre() +
                    "  /  " + "Apellido: " + elRecuperado.getApellido() +
                    "  /  " + "Edad: " + elRecuperado.getEdad() +
                    "  /  " + "Cédula: " + elRecuperado.getCedula() +
                    "  /  " + "Color Favorito: " + elRecuperado.getColor());
        }
    }
}
```

Clase Control:



```java
package ejercicioBasico;

import java.util.Scanner;

public class Control {  2 usages
    Pilas mipila = new Pilas();  3 usages
    //Permitira apilar números del 0 al 4

    Scanner scan = new Scanner(System.in);  6 usages

    //Método para guardar números del 0 al 4
    //void guardar(){
    //    for (int i = 0; i < 5; i++) {
    //        mipila.push(i);
    //    }
    //    System.out.println("Números del 0 al 4 Apilados");
    //}

    //Método para agregar Datos de la Persona
    public void agregarDatos(){  1 usage
        System.out.print("Digite el Nombre de la Persona: ");
        String nombre = scan.next();
        System.out.print("Digite el Apellido de la Persona: ");
        String apellido = scan.next();
        System.out.print("Digite el Color Favorito de la Persona: ");
        String color = scan.next();
        System.out.print("Digite la Edad de la Persona: ");
        int edad = scan.nextInt();
        System.out.print("Digite la Cédula de la Persona: ");
        int cedula = scan.nextInt();

        //Seteamos
```

```
        Elemento elm = new Elemento();
        elm.setNombre(nombre);
        elm.setApellido(apellido);
        elm.setEdad(edad);
        elm.setCedula(cedula);
        elm.setColor(color);

        //Llamamos al Método para apilar este objeto
        mipila.push(elm);
        System.out.println("Persona guardada en la Pila!!!");
    }

    //Método para Listar
    //void listarPila(){
    //    System.out.println("Listando la pila...");
    //    System.out.print("( Cabeza )");
    //    for (Object o : mipila.lista) {
    //        System.out.print(o + " / ");
    //    }
    //}

    //Método para Eliminar de la Pila
    //void eliminar(){
    //    System.out.println("Eliminando dato de la pila");
    //    System.out.println("Dato eliminado: " + mipila.pop());
    //    System.out.println("Tamaño de la pila: " + mipila.size());
    //}

    //Menú de Opciones
    void menu(){  1 usage
```

```
        int opcion;
        do {
            System.out.println("""
                    \n****MENÚ DE OPCIONES****

                    1. Apilar una Persona a la lista
                    2. Listar todas las Personas
                    3. Desapilar una Persona de la Lista
                    4. Salir del Programa""");
            System.out.println("");
            System.out.print("Digite una opción: ");
            opcion = scan.nextInt();
            switch(opcion) {
                case 1:
                    this.agregarDatos();
                    break;
                case 2:
                    mipila.ListarTodo();
                    break;
                case 3:
                    System.out.println(mipila.pop());
                    break;
            }
        } while(opcion != 4);
    }
}
```

Clase Prueba:

```java
package ejercicioBasico;

public class Prueba {
    public static void main(String[] args) {
        Control con = new Control();
        //con.guardar();
        //con.listarPila();
        //con.eliminar();
        //con.listarPila();
        con.menu();
    }
}
```

Código Ejecutado:

a) Apilar una Persona a la Lista

```
****MENÚ DE OPCIONES****

1. Apilar una Persona a la lista
2. Listar todas las Personas
3. Desapilar una Persona de la Lista
4. Salir del Programa

Digite una opción: 1
Digite el Nombre de la Persona: Andres
Digite el Apellido de la Persona: Garces
Digite el Color Favorito de la Persona: Verde
Digite la Edad de la Persona: 18
Digite la Cédula de la Persona: 12345678
Persona guardada en la Pila!!!
```

b) Listar todas las Personas

```
****MENÚ DE OPCIONES****

1. Apilar una Persona a la lista
2. Listar todas las Personas
3. Desapilar una Persona de la Lista
4. Salir del Programa

Digite una opción: 2
****Listando  Personas de la Pila****
( Cabecera )
Nombre: Felipe  /  Apellido: Bolaños  /  Edad: 30  /  Cédula: 24681357  /  Color Favorito: Amarillo
Nombre: Alexis  /  Apellido: Quenan  /  Edad: 19  /  Cédula: 1234567890  /  Color Favorito: Naranja
Nombre: Michael  /  Apellido: Giraldo  /  Edad: 24  /  Cédula: 987654321  /  Color Favorito: Rojo
Nombre: Salomon  /  Apellido: Jimenez  /  Edad: 20  /  Cédula: 87654321  /  Color Favorito: Azul
Nombre: Andres  /  Apellido: Garces  /  Edad: 18  /  Cédula: 12345678  /  Color Favorito: Verde
```

c) Desapilar una Persona de Lista

```
****MENÚ DE OPCIONES****

1. Apilar una Persona a la lista
2. Listar todas las Personas
3. Desapilar una Persona de la Lista
4. Salir del Programa

Digite una opción: 3
Persona Desapilada - Cedula: 24681357
```

- Listar todas las Personas (Cuando una se haya desapilado)

```
****MENÚ DE OPCIONES****

1. Apilar una Persona a la lista
2. Listar todas las Personas
3. Desapilar una Persona de la Lista
4. Salir del Programa

Digite una opción: 2
****Listando  Personas de la Pila****
( Cabecera )
Nombre: Alexis  /  Apellido: Quenan  /  Edad: 19  /  Cédula: 1234567890  /  Color Favorito: Naranja
Nombre: Michael  /  Apellido: Giraldo  /  Edad: 24  /  Cédula: 987654321  /  Color Favorito: Rojo
Nombre: Salomon  /  Apellido: Jimenez  /  Edad: 20  /  Cédula: 87654321  /  Color Favorito: Azul
Nombre: Andres  /  Apellido: Garces  /  Edad: 18  /  Cédula: 12345678  /  Color Favorito: Verde
```

Trabajo 9

Ejercicio Pilas Básico 2

Punto 1

Realice un programa para hacer una pila de cubos, sabiendo que cada cubo tiene un color diferente. A través de un menú se ofrecerán las siguientes opciones:

- **a)** Apilar 5 Cubos de colores, los cuales están marcados con el color. El programa deberá pedir al usuario de qué color es el cubo a agregar, para poder simular el proceso.
- **b)** Listar los Cubos que están en la pila, para ello mostrara de cada cubo el color.
- **c)** Eliminar un Cubo de la pila indicando cual fue eliminado.
- **d)** ¿Qué pasa si al tratar de eliminar un cubo de la pila, esta se encuentra vacía?

- Desarrollo

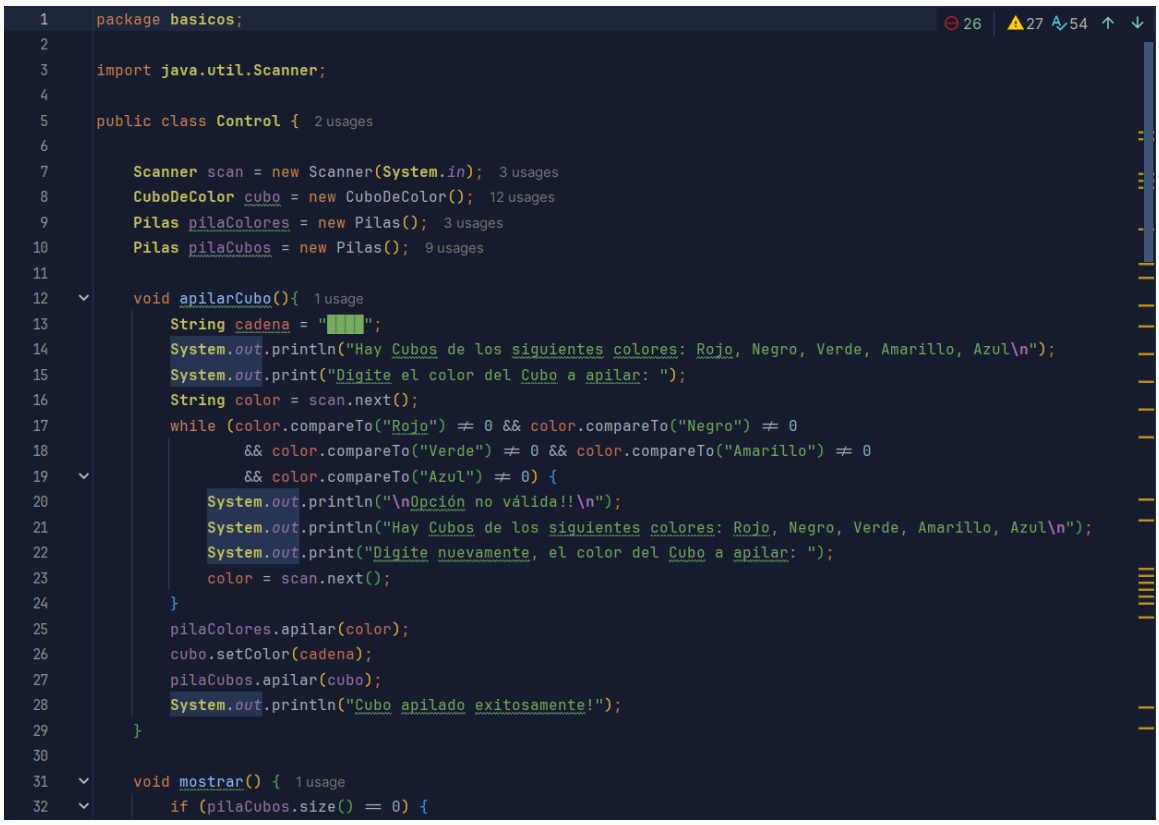
Clase Elemento:

```
1   package basicos;
2
    7 usages
3   public class CubodeColor {
        2 usages
4       String color;
5
        5 usages
6       public String getColor() {return color;}
7
        1 usage
8       public void setColor(String color) {this.color = color;}
9   }
```

Clase Pilas:

```java
package basicos;

import java.util.LinkedList;

4 usages
public class Pilas {
    6 usages
    LinkedList pila = new LinkedList();

    2 usages
    int size() { return pila.size(); }

    2 usages
    void apilar(Object obj) { pila.addFirst(obj); }

    2 usages
    Object desapilar(){
        if (pila.isEmpty()) {
            return null;
        }else{
            return pila.removeFirst();
        }
    }

    no usages
    Object cima() { return pila.getFirst(); }

    6 usages
    Object obtener(int index) { return pila.get(index); }
}
```

Clase Control:



```java
package basicos;

import java.util.Scanner;

public class Control {

    Scanner scan = new Scanner(System.in);
    CuboDeColor cubo = new CuboDeColor();
    Pilas pilaColores = new Pilas();
    Pilas pilaCubos = new Pilas();

    void apilarCubo(){
        String cadena = "";
        System.out.println("Hay Cubos de los siguientes colores: Rojo, Negro, Verde, Amarillo, Azul\n");
        System.out.print("Digite el color del Cubo a apilar: ");
        String color = scan.next();
        while (color.compareTo("Rojo") != 0 && color.compareTo("Negro") != 0
                && color.compareTo("Verde") != 0 && color.compareTo("Amarillo") != 0
                && color.compareTo("Azul") != 0) {
            System.out.println("\nOpción no válida!!\n");
            System.out.println("Hay Cubos de los siguientes colores: Rojo, Negro, Verde, Amarillo, Azul\n");
            System.out.print("Digite nuevamente, el color del Cubo a apilar: ");
            color = scan.next();
        }
        pilaColores.apilar(color);
        cubo.setColor(cadena);
        pilaCubos.apilar(cubo);
        System.out.println("Cubo apilado exitosamente!");
    }

    void mostrar() {
        if (pilaCubos.size() == 0) {
            System.out.println("No hay Cubos apilados!");
        } else {
            System.out.println("Listando Cubos: \n");

            for (int i = 0; i < pilaCubos.size(); i++) {
                String color = (String) pilaColores.obtener(i);
                switch (color) {
                    case "Rojo":
                        cubo = (CuboDeColor) pilaCubos.obtener(i);
                        System.out.println("\u001B[31m" + cubo.getColor() + "\u001B[0m");
                        break;
                    case "Negro":
                        cubo = (CuboDeColor) pilaCubos.obtener(i);
                        System.out.println("\u001B[30m" + cubo.getColor() + "\u001B[0m");
                        break;
                    case "Verde":
                        cubo = (CuboDeColor) pilaCubos.obtener(i);
                        System.out.println("\u001B[32m" + cubo.getColor() + "\u001B[0m");
                        break;
                    case "Amarillo":
                        cubo = (CuboDeColor) pilaCubos.obtener(i);
                        System.out.println("\u001B[33m" + cubo.getColor() + "\u001B[0m");
                        break;
                    case "Azul":
                        cubo = (CuboDeColor) pilaCubos.obtener(i);
                        System.out.println("\u001B[34m" + cubo.getColor() + "\u001B[0m");
                        break;
                }
            }
        }
```

```
63      }
64
65      void desapilarCubo(){  1 usage
66          if (pilaCubos.desapilar() == null) {
67              System.out.println("No hay Cubos apilados!!");
68          }else{
69              String color = (String) pilaColores.desapilar();
70              System.out.println("Se ha desapilado un Cubo de color: " + color);
71          }
72      }
73
74      void menu(){  2 usages
75          int opc;
76          do {
77              System.out.println("\n****MENÚ DE OPCIONES****\n");
78              System.out.println("1. Apilar un Cubo");
79              System.out.println("2. Mostrar Cubos apilados");
80              System.out.println("3. Desapilar un Cubo");
81              System.out.println("4. Salir del programa");
82              System.out.print("\nDigite una opción: ");
83              opc = scan.nextInt();
84              System.out.println("");
85
86              switch (opc) {
87                  case 1:
88                      this.apilarCubo();
89                      break;
90                  case 2:
91                      this.mostrar();
92                      break;
```

```
93                  case 3:
94                      this.desapilarCubo();
95                      break;
96                  case 4:
97                      System.out.println("Fin del Programa");
98                      break;
99                  default:
100                     System.out.println("Opción no válida!!");
101 C                   menu();
102                     break;
103             }
104         } while (opc != 4);
105     }
106 }
```

Clase Prueba:

```java
package basicos;

public class PruebaCubos {
    public static void main(String[] args) {
        Control obj = new Control();
        obj.menu();
    }
}
```

Código Ejecutado:

a) Apilar 5 Cubos de colores, los cuales están marcados con el color. El programa deberá pedir al usuario de qué color es el cubo a agregar, para poder simular el proceso.

```
****MENÚ DE OPCIONES****

1. Apilar un Cubo
2. Mostrar Cubos apilados
3. Desapilar un Cubo
4. Salir del programa

Digite una opción: 1

Hay Cubos de los siguientes colores: Rojo, Negro, Verde, Amarillo, Azul

Digite el color del Cubo a apilar: Rojo
Cubo apilado exitosamente!
```

```
Digite el color del Cubo a apilar: Azul
Cubo apilado exitosamente!
```

```
Digite el color del Cubo a apilar: Amarillo
Cubo apilado exitosamente!
```

```
Digite el color del Cubo a apilar: Verde
Cubo apilado exitosamente!
```

```
Digite el color del Cubo a apilar: Negro
Cubo apilado exitosamente!
```

b) Listar los Cubos que están en la pila, para ello mostrara de cada cubo el color.

```
****MENU****
1. Apilar un Cubo
2. Mostrar Cubos Apilados
3. Desapilar un Cubo
4. Salir del programa

Digite una opción: 2

Listando Cubos:
```

c) Eliminar un Cubo de la pila indicando cual fue eliminado.

```
****MENU****
1. Apilar un Cubo
2. Mostrar Cubos Apilados
3. Desapilar un Cubo
4. Salir del programa

Digite una opción: 3

Se ha desapilado un Cubo de color Negro
```

- Mostrando la Lista de Cubos al ya haber desapilado un Cubo.

```
****MENÚ DE OPCIONES****

1. Apilar un Cubo
2. Mostrar Cubos apilados
3. Desapilar un Cubo
4. Salir del programa

Digite una opción: 2

Listando Cubos:
```

d) ¿Qué pasa si al tratar de eliminar un cubo de la pila, esta se encuentra vacía?

```
****MENU****
1. Apilar un Cubo
2. Mostrar Cubos Apilados
3. Desapilar un Cubo
4. Salir del programa

Digite una opción: 3

No hay Cubos Apilados!
```

COLAS

Trabajo 10

Ejercicio de Colas (TALLER DE COLAS.pdf)

Punto 1

Para el ingreso a un centro de diagnóstico automotor, los carros se atienden por orden de llegada. De cada vehículo se toma el número de placa, la cedula del propietario y el modelo. Realice un programa donde a través de un menú permita:

- **a)** Simular la llegada de un Vehículo
- **b)** Listar los Vehículos en la fila, utilizando una tabla así: Placa / Modelo / Cédula del Propietario
- **c)** Simular la atención de un Vehículo e indicar la Placa del Vehículo que está siendo atendido.

Solución

Clases: Vehículo, Cola, Principal (JavaMain), Control.

- Desarrollo

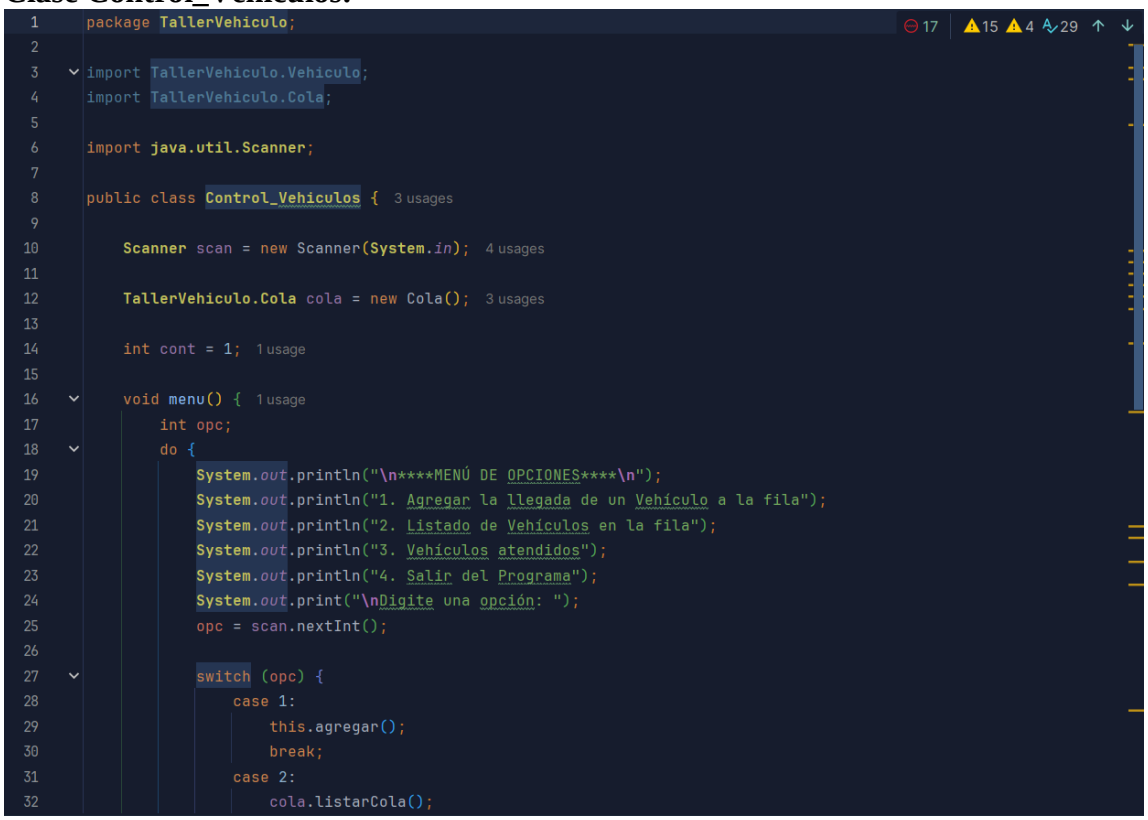
Clase Vehiculo:

```java
package TallerVehiculo;

public class Vehiculo {  7 usages

    String Modelo;  2 usages
    int Placa, Cedula;  2 usages

    public String getModelo() {  1 usage
        return Modelo;
    }

    public void setModelo(String modelo) {  1 usage
        Modelo = modelo;
    }

    public int getPlaca() {  2 usages
        return Placa;
    }

    public void setPlaca(int placa) {  1 usage
        Placa = placa;
    }

    public int getCedula() {  1 usage
        return Cedula;
    }

    public void setCedula(int cedula) {  1 usage
        Cedula = cedula;
    }
}
```

Clase Cola:

```java
package TallerVehiculo;

import java.util.Collections;
import java.util.LinkedList;

public class Cola {  3 usages

    LinkedList cola = new LinkedList();  6 usages

    //Agregar elementos a la cola. Agrega por el final
    public void enqueue(Object o) { cola.addLast(o); }

    public void listarCola(){  2 usages
        Vehiculo vehicle;
        if(cola.isEmpty()) {
            System.out.println("No se puede listar. La cola esta vacia");

        } else {
            System.out.println("Listado de Vehiculos\n");
            System.out.println("( Inicio )");
            System.out.println("Placa / Modelo / Cédula del Propietario");

            for (Object it : cola) {
                vehicle = (Vehiculo) it;
                System.out.println(vehicle.getPlaca() + " / " + vehicle.getModelo() + " / " + vehicle.getCedula());
            }
            System.out.print("( Fin )");
        }
    }

    //Eliminar elementos de la cola. Elimina del principio
    public void dequeue(){  1 usage
        Vehiculo vehAtendido;
        if(cola.isEmpty()) {
            System.out.println("No se puede eliminar. La cola esta vacia");

        } else {
            vehAtendido = (Vehiculo) cola.removeFirst(); //Por ejemplo, retorna un Objeto Persona
            //Lo que está en la derecha es lo que va a Retornar el Método

            System.out.println("\nAtendido el Vehículo, con la Placa: " + vehAtendido.getPlaca());
        }
    }

    public void ordenar(){  no usages
        System.out.print("Ordenando Vehículos");
        Collections.sort(cola);
        listarCola();
    }
}
```

Clase Control_Vehiculos:



```java
package TallerVehiculo;

import TallerVehiculo.Vehiculo;
import TallerVehiculo.Cola;

import java.util.Scanner;

public class Control_Vehiculos {  3 usages

    Scanner scan = new Scanner(System.in);  4 usages

    TallerVehiculo.Cola cola = new Cola();  3 usages

    int cont = 1;  1 usage

    void menu() {  1 usage
        int opc;
        do {
            System.out.println("\n****MENÚ DE OPCIONES****\n");
            System.out.println("1. Agregar la llegada de un Vehículo a la fila");
            System.out.println("2. Listado de Vehículos en la fila");
            System.out.println("3. Vehículos atendidos");
            System.out.println("4. Salir del Programa");
            System.out.print("\nDigite una opción: ");
            opc = scan.nextInt();

            switch (opc) {
                case 1:
                    this.agregar();
                    break;
                case 2:
                    cola.listarCola();
```

```java
                    System.out.println();
                    break;
                case 3:
                    cola.dequeue();
                    break;
            }
        } while (opc != 4);
    }

    void agregar(){  1 usage
        System.out.println("\nAgregar la llegada del Vehículo " + cont++ + " a la Fila");
        System.out.print("Número de Placa: ");
        int placa = scan.nextInt();
        System.out.print("Cédula del Propietario: ");
        int cedula = scan.nextInt();
        System.out.print("Modelo: ");
        String modelo = scan.next();

        //Seteo
        Vehiculo vehiculo = new Vehiculo();
        vehiculo.setPlaca(placa);
        vehiculo.setCedula(cedula);
        vehiculo.setModelo(modelo);

        //Agrego a la cola de elementos
        cola.enqueue(vehiculo);
        System.out.println("\nVehículo agregado a la fila!!");
    }
}
```

Clase Principal:

```java
package TallerVehiculo;

import TallerVehiculo.Control_Vehiculos;

public class Principal {
    public static void main(String[] args) {
        Control_Vehiculos con = new Control_Vehiculos();
        con.menu();
    }
}
```

Código Ejecutado:

a) Simular la llegada de un Vehículo:

```
****MENÚ DE OPCIONES****

1. Agregar la llegada de un Vehículo a la fila
2. Listado de Vehículos en la fila
3. Vehículos atendidos
4. Salir del Programa

Digite una opción: 1

Agregar la llegada del Vehículo 1 a la Fila
Número de Placa: 123342
Cédula del Propietario: 2837834
Modelo: AUDI

Vehículo agregado a la fila!!
```

b) Listar los Vehículos en la fila, utilizando una tabla así: Placa / Modelo / Cédula del Propietario

```
****MENÚ DE OPCIONES****

1. Agregar la llegada de un Vehículo a la fila
2. Listado de Vehículos en la fila
3. Vehículos atendidos
4. Salir del Programa

Digite una opción: 2
Listado de Vehículos

( Inicio )
Placa / Modelo / Cédula del Propietario
123342 / AUDI / 2837834
389438 / Bentley / 492832
5930293 / BMW / 6940342
79320943 / Bugatti / 8874328
( Fin )
```

c) Simular la atención de un Vehículo e indicar la Placa del Vehículo que está siendo atendido.

```
****MENÚ DE OPCIONES****

1. Agregar la llegada de un Vehículo a la fila
2. Listado de Vehículos en la fila
3. Vehículos atendidos
4. Salir del Programa

Digite una opción: 3

Atendido el Vehículo, con la Placa: 123342
```

- Listado de Vehículos, cuando ya se atendió a uno

```
****MENÚ DE OPCIONES****

1. Agregar la llegada de un Vehículo a la fila
2. Listado de Vehículos en la fila
3. Vehículos atendidos
4. Salir del Programa

Digite una opción: 2
Listado de Vehículos

( Inicio )
Placa / Modelo / Cédula del Propietario
389438 / Bentley / 492832
5930293 / BMW / 6940342
79320943 / Bugatti / 8874328
( Fin )
```

FUNCIONES RECURSIVAS

Trabajo 11

Ejercicios de Funciones Recursivas (Taller 1 Funciones Recursivas.pdf)

Punto 1

a) Haciendo uso de las funciones recursivas predecir el término n de la siguiente serie: 1,2,2,4,8,32, la cual comienza con los términos 1 y 2, y los demás números son generados a partir del producto de los números anteriores.

b) Haciendo uso de una función recursiva, calcular la potencia de un número: "x a la n", dados el valor de x y el valor de n.

c) Haciendo uso de una función recursiva, invertir las cifras de un número de 4 cifras. Ej: 328 = 823

- Desarrollo

a) Clase Serie:

```java
import java.util.Scanner;

public class Serie {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("\nDigite el primer término de la serie: ");
        int first = scanner.nextInt();

        System.out.print("\nDigite el segundo término de la serie: ");
        int second = scanner.nextInt();

        System.out.print("\nDigite el término que desea predecir (n): ");
        int n = scanner.nextInt();

        int result = predictTerm(first, second, n);
        System.out.println("\nEl término " + n + " de la serie es: " + result);
    }

    public static int predictTerm(int first, int second, int n) {  3 usages
        if (n == 1) {
            return first;
        } else if (n == 2) {
            return second;
        } else {
            return predictTerm(first, second, n: n - 1) * predictTerm(first, second, n: n - 2);
        }
    }
}
```

- **Código Ejecutado:**

```
Digite el primer término de la serie: 1

Digite el segundo término de la serie: 2

Digite el término que desea predecir (n): 7

El término 7 de la serie es: 256
```

b) Clase Potencia:

```java
import java.util.Scanner;

public class Potencia {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.print("\nDigite la Base: ");
        double x = scan.nextInt();

        System.out.print("\nDigite el Exponente: ");
        int n = scan.nextInt();

        double Resultado = Exponencial(x, n);
        System.out.println("\n" + x + " elevado a la " + n + " es igual a " + Resultado);
    }

    public static double Exponencial(double x, int n) {
        if (n == 0) {
            return 1;
        }

        if (n == 1) {
            return x;
        }

        //Si n es negativo, calculamos la potencia recíproca
        if (n < 0) {
            return 1 / Exponencial(x, -n);
        }

        //Caso recursivo: calcular x^n = x * x^(n-1)
        return x * Exponencial(x, n - 1);
    }
}
```

- **Código Ejecutado:**