Informe de Laboratorio 04

Tema: Arreglos de objetos, búsqueda, ordenamiento

Nota

Estudiante	Escuela	Asignatura
Roni Companocca Checco rcompanocca@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Semestre: II Código: 20210558

Laboratorio	Tema	Duración
04	Arreglos de objetos, búsqueda, ordenamiento	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 26 Septiembre 2023	Al 27 Septiembre 2023

1. TAREA

1.1. Objetivos:

- Crear e inicializar arreglos de objetos
- Realizar Búsquedas secuencial y binaria en un arreglo de objetos
- Implementar método de ordenamiento de burbuja en arreglos de objetos.
- Arreglos de Objetos: Búsquedas y Ordenamiento

1.2. Competencias a alcanzar:

- Diseña, responsablemente, sistemas, componentes o procesos para satisfacer necesidades dentro de restricciones realistas: económicas, medio ambientales, sociales, políticas, éticas, de salud, de seguridad, manufacturación y sostenibilidad.
- Aplica de forma flexible, técnicas, métodos, principios, normas, estándares y herramientas de ingeniería necesarias para la construcción de software e implementación de sistemas de información.

1.3. Consideraciones de evaluacion:

- No deberá utilizar constructores no vistos en clase
- No podrá modificar el código base entregado para este laboratorio
- Deberá utilizar nombre de variables significativos
- Deberá realizar pruebas adicionales
- El alumno deberá indicar en su código con quien colaboró
- El alumno será requerido de realizar modificaciones en su código y responder a preguntas sobre el mismo
- Todos los ejercicios deberán traerse terminados en caso de ser tarea para la casa
- Si tiene ejercicios sin terminar no importa, se revisará el avance y se discutirá sobre las dificultades encontradas.

1.4. Indicaciones generales:

- Todos los ejercicios deberán ser guardados en el mismo Proyecto
- El Proyecto deberá tener el nombre del Laboratorio y el nombre del alumno, así por ejemplo: Laboratorio 1 – Juan Perez
- Cada Clase deberá tener el nombre del ejercicio, así por ejemplo: Ejercicio1
- Utilice nombres de variables significativos y todas las recomendaciones de estilo
- Especialmente, su código deberá estar correctamente indentado
- Deberá pasar TODOS los casos de prueba

2. EQUIPOS, MATERIALES Y TEMAS UTILIZADOS

- Sistema Operativo Windows
- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.

3. URL DE REPOSITORIO GITHUB

- URL para el Repositorio GitHub.
- `https://github.com/RONI-COMPANOCCA-CHECCO`
- URL para el laboratorio 02 en el Repositorio GitHub.
- `https://github.com/RONI-COMPANOCCA-CHECCO/FP2-LAB4`

4. ACTIVIDADES

4.1. EJERCICIO PROPUESTO

- Cree un Proyecto llamado Laboratorio4
- Usted podrá reutilizar las dos clases Nave.java y DemoBatalla.java. creadas en Laboratorio 3
- Completar el Código de la clase DemoBatalla
- la clase DemoBatalla.java

```
       // RONI COMPANOCCA CHECCO
// CUI: 20210558
// LABORATORIO 04
// FUNDAMENTOS DE PROGRAMACION
import java.util.Scanner;

public class DemoBatalla {
    public static void main(String[] args) {
        Nave[] misNaves = new Nave[8];
        Scanner sc = new Scanner(System.in);
        String nomb, col;
        int fil, punt;
        boolean est;

        for (int i = 0; i < misNaves.length; i++) {
            System.out.println("Nave " + (i + 1));
            System.out.print("Nombre: ");
            nomb = sc.next();
            System.out.print("Fila: ");
            fil = sc.nextInt();
            System.out.print("Columna: ");
            col = sc.next();
            System.out.print("Estado: ");
            est = sc.nextBoolean();
            System.out.print("Puntos: ");
            punt = sc.nextInt();
            misNaves[i] = new Nave();
            misNaves[i].setNombre(nomb);
            misNaves[i].setFila(fil);
            misNaves[i].setColumna(col);
            misNaves[i].setEstado(est);
            misNaves[i].setPuntos(punt);
        }

        System.out.println("\nNaves creadas:");
        mostrarNaves(misNaves);

        System.out.print("Ingrese el nombre de la nave a buscar: ");
        String nombreBuscado = sc.next();
        mostrarPorNombre(misNaves, nombreBuscado);

        System.out.print("Ingrese el nmero de puntos mximo: ");
        int puntosMaximos = sc.nextInt();
        mostrarPorPuntos(misNaves, puntosMaximos);
```

```
        System.out.println("\nNave con mayor nmero de puntos:");
        Nave naveMayorPuntos = mostrarMayorPuntos(misNaves);
        if (naveMayorPuntos != null) {
            naveMayorPuntos.mostrarInformacion();
        } else {
            System.out.println("No se encontr ninguna nave activa.");
        }

        ordenarPorPuntosBurbuja(misNaves);
        System.out.println("\nNaves ordenadas por puntos (Burbuja):");
        mostrarNaves(misNaves);

        ordenarPorNombreBurbuja(misNaves);
        System.out.println("\nNaves ordenadas por nombre (Burbuja):");
        mostrarNaves(misNaves);

        ordenarPorPuntosSeleccion(misNaves);
        System.out.println("\nNaves ordenadas por puntos (Seleccin):");
        mostrarNaves(misNaves);

        ordenarPorNombreSeleccion(misNaves);
        System.out.println("\nNaves ordenadas por nombre (Seleccin):");
        mostrarNaves(misNaves);

        ordenarPorPuntosInsercion(misNaves);
        System.out.println("\nNaves ordenadas por puntos (Insercin):");
        mostrarNaves(misNaves);

        ordenarPorNombreInsercion(misNaves);
        System.out.println("\nNaves ordenadas por nombre (Insercin):");
        mostrarNaves(misNaves);
    }

    static void mostrarNaves(Nave[] flota) {
        for (Nave nave : flota) {
            nave.mostrarInformacion();
        }
    }

    static void mostrarPorNombre(Nave[] flota, String nombre) {
        boolean encontrado = false;
        for (Nave nave : flota) {
            if (nave.getNombre().equalsIgnoreCase(nombre)) {
                nave.mostrarInformacion();
                encontrado = true;
                break;
            }
        }
        if (!encontrado) {
            System.out.println("Nave no encontrada.");
        }
    }

    static void mostrarPorPuntos(Nave[] flota, int puntosMaximos) {
        System.out.println("Naves con puntos menores o iguales a " + puntosMaximos +
            ":");
```

```
        for (Nave nave : flota) {
            if (nave.getPuntos() <= puntosMaximos) {
                nave.mostrarInformacion();
            }
        }
    }

    static Nave mostrarMayorPuntos(Nave[] flota) {
        Nave mayorPuntos = null;
        for (Nave nave : flota) {
            if (mayorPuntos == null || nave.getPuntos() > mayorPuntos.getPuntos()) {
                mayorPuntos = nave;
            }
        }
        return mayorPuntos;
    }

    static void ordenarPorPuntosBurbuja(Nave[] flota) {
        int n = flota.length;
        boolean intercambio;
        do {
            intercambio = false;
            for (int i = 0; i < n - 1; i++) {
                if (flota[i].getPuntos() > flota[i + 1].getPuntos()) {
                    // Intercambiar las naves
                    Nave temp = flota[i];
                    flota[i] = flota[i + 1];
                    flota[i + 1] = temp;
                    intercambio = true;
                }
            }
        } while (intercambio);
    }

    static void ordenarPorNombreBurbuja(Nave[] flota) {
        int n = flota.length;
        boolean intercambio;
        do {
            intercambio = false;
            for (int i = 0; i < n - 1; i++) {
                if (flota[i].getNombre().compareToIgnoreCase(flota[i + 1].getNombre()) >
                    0) {
                    // Intercambiar las naves
                    Nave temp = flota[i];
                    flota[i] = flota[i + 1];
                    flota[i + 1] = temp;
                    intercambio = true;
                }
            }
        } while (intercambio);
    }

}
```

- la clase Nave.java

```java
public class Nave {
    private String nombre;
    private int fila;
    private String columna;
    private boolean estado;
    private int puntos;

    // Constructor
    public Nave(String nombre, int fila, String columna, boolean estado, int puntos) {
        this.nombre = nombre;
        this.fila = fila;
        this.columna = columna;
        this.estado = estado;
        this.puntos = puntos;
    }

    // Mtodos mutadores
    public void setNombre(String n) {
        nombre = n;
    }

    public void setFila(int f) {
        fila = f;
    }

    public void setColumna(String c) {
        columna = c;
    }

    public void setEstado(boolean e) {
        estado = e;
    }

    public void setPuntos(int p) {
        puntos = p;
    }

    // Mtodos accesores
    public String getNombre() {
        return nombre;
    }

    public int getFila() {
        return fila;
    }

    public String getColumna() {
        return columna;
    }

    public boolean getEstado() {
        return estado;
    }

    public int getPuntos() {
        return puntos;
```

```
    }

    // Mtodo para mostrar informacin de la nave
    public void mostrarInformacion() {
        System.out.println("Nombre: " + nombre);
        System.out.println("Fila: " + fila);
        System.out.println("Columna: " + columna);
        System.out.println("Estado: " + (estado ? "Activa" : "Inactiva"));
        System.out.println("Puntos: " + puntos);
    }

    // Mtodo para recibir dao
    public void recibirDanio(int cantidadDanio) {
        if (estado) {
            puntos -= cantidadDanio;
            if (puntos <= 0) {
                estado = false;
                puntos = 0;
            }
        }
    }
}
```

5. CUESTIONARIO

5.1.

Cómo se declara e inicializa un arreglo bidimensional?

- Se debe inicializar de esta forma

```
int[][] arregloBidimensional = {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12}
};
```

5.2.

Qué ventajas tienen los arreglos bidimensionales con respecto a los arreglos unidimensionales?

- Los arreglos bidimensionales tienen varias ventajas con respecto a los arreglos unidimensionales cuando se trata de organizar y trabajar con datos que tienen una estructura de dos dimensiones, como una matriz o una tabla

5.3.

Para que se utilizan los métodos burbuja, inserción y selección? ¿Qué finalidad tienen?

- Los métodos de ordenación burbuja, inserción y selección son algoritmos de ordenación utilizados para organizar elementos en una lista o arreglo en un orden específico, generalmente en orden ascendente o descendente.

6. REFERENCIAS

- M. Aedo, "Fundamentos de Programación 2 - Tópicos de Programación Orientada a Objetos", Primera Edición, 2021, Editorial UNSA.
- `https://github.com/rescobedoq/programacion.git`
- J. Dean, Ïntroduction to programming with Java: A Problem Solving Approach", Third Edition, 2021, McGraw-Hill.
- C. T. Wu, .ªn Introduction to Object-Oriented Programming with Java", Fifth Edition, 2010, McGraw-Hill.
- P. Deitel, "Java How to Program", Eleventh Edition, 2017, Prentice Hall.