
	UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

## INFORME DE LABORATORIO

### (formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	FUNDAMENTOS DE PROGRAMACIÓN 2				
TÍTULO DE LA PRÁCTICA:	ArrayList				
NÚMERO DE PRÁCTICA:	6	AÑO LECTIVO:	2024	NRO. SEMESTRE:	2024-B
FECHA DE PRESENTACIÓN	25/10/2024	HORA DE PRESENTACIÓN	18/20/00		
INTEGRANTE (s) Riveros Vilca Alberth Edwar				NOTA (0-20)	
DOCENTE(s): Ing. Lino Jose Pinto Oppe					

RESULTADOS Y PRUEBAS
<p><b>I. EJERCICIOS RESUELTOS:</b></p> <ol style="list-style-type: none"> <li>1. Cree un Proyecto llamado Laboratorio6</li> <li>2. Usted deberá crear las dos clases Soldado.java y VideoJuego3.java. Puede reutilizar lo desarrollado en Laboratorios anteriores.</li> <li>3. Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).</li> <li>4. El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Pero ahora el tablero debe ser un ArrayList bidimensional.</li> <li>5. Tendrá 2 Ejércitos. Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados (distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento. Finalmente, que muestre qué ejército ganará la batalla (indicar</li> </ol>

la métrica usada para decidir al ganador de la batalla).

### CLASE SOLDADO:

```
public class Soldado { 30 usages new *
    private String nombre; 4 usages
    private int fila; 4 usages
    private int columna; 4 usages
    private int nivelVida; 4 usages
    // Metodos mutadores
    public Soldado(String nombre, int fila, int columna, int nivelVida) { 1 usage
        this.nombre = nombre;
        this.fila = fila;
        this.columna = columna;
        this.nivelVida = nivelVida;
    }
    public void setNombre(String n){ no usages new *
        nombre = n;
    }
    public void setFila(int f){ no usages new *
        fila = f;
    }
    public void setColumna(int c){ no usages new *
        columna = c;
    }

    public void setNivelVida(int p){ no usages new *
        nivelVida = p;
    }
    // Metodos accesoros
    public String getNombre(){ 2 usages new *
        return nombre;
    }
    public int getFila(){ no usages new *
        return fila;
    }
    public int getColumna(){ no usages new *
        return columna;
    }

    public int getNivelVida(){ 7 usages new *
        return nivelVida;
    }
    @Override new *
    public String toString(){
        return "[Nombre: "+nombre+"\tFila: "+(fila+1)+"\tColumna: "+(columna+1)+
            "\tnivel de Vida: "+ nivelVida +"]"+"\\n";
    }
}
```

**CLASE VIDEOJUEGO 2:**

```
import java.util.*;

public class VideoJuego2 { new *
    public static void main(String[] args) { new *
        Random rand = new Random();
        ArrayList<ArrayList<Boolean>> casillasOcupadas = new ArrayList<>();
        ArrayList<ArrayList<Soldado>> ejercito1 = new ArrayList<>();
        ArrayList<ArrayList<Soldado>> ejercito2 = new ArrayList<>();
        ArrayList<String> nombresEjercito1 = new ArrayList<>();
        ArrayList<String> nombresEjercito2 = new ArrayList<>();
        int numSoldados1 = rand.nextInt( bound: 10) + 1;
        int numSoldados2 = rand.nextInt( bound: 10) + 1;

        for (int i = 0; i < 10; i++) {
            casillasOcupadas.add(new ArrayList<Boolean>());
            ejercito1.add(new ArrayList<Soldado>());
            ejercito2.add(new ArrayList<>());
            for (int j = 0; j < 10; j++) {
                casillasOcupadas.get(i).add(false);
                ejercito1.get(i).add(null);
                ejercito2.get(i).add(null);
            }
        }

        crearEjercito(rand, numSoldados1, ejercito1, casillasOcupadas, nombresEjercito1, ejercitoNum: 1);
        crearEjercito(rand, numSoldados2, ejercito2, casillasOcupadas, nombresEjercito2, ejercitoNum: 2);

        showBoard(casillasOcupadas, ejercito1, ejercito2);

        System.out.println("Ejercito 1:");
        mostrarDatosEjercito(ejercito1, nombresEjercito1, numSoldados1);
        System.out.println("Ejercito 2:");
        mostrarDatosEjercito(ejercito2, nombresEjercito2, numSoldados2);
        determinarGanador(ejercito1, ejercito2);
    }
}
```

```
// Determina el ganador basándose en la vida total de ambos ejércitos
public static void determinarGanador(ArrayList<ArrayList<Soldado>> ejercito1, 1usage new *
                                   ArrayList<ArrayList<Soldado>> ejercito2) {

    int totalVidaEjercito1 = calcularTotalVida(ejercito1);
    int totalVidaEjercito2 = calcularTotalVida(ejercito2);

    System.out.printf("Total Vida Ejército 1: %d\n", totalVidaEjercito1);
    System.out.printf("Total Vida Ejército 2: %d\n", totalVidaEjercito2);

    if (totalVidaEjercito1 > totalVidaEjercito2) {
        System.out.println("El Ejército 1 gana la batalla.");
    } else if (totalVidaEjercito2 > totalVidaEjercito1) {
        System.out.println("El Ejército 2 gana la batalla.");
    } else {
        System.out.println("La batalla termina en empate.");
    }
}

// Muestra la información del ejército, incluyendo el soldado con mayor vida y el promedio
public static void mostrarDatosEjercito(ArrayList<ArrayList<Soldado>> ejercito, ArrayList<String> nombresEjercito, int numSoldados) {
    String mayorVida = findMaxLifeSoldier(ejercito);
    System.out.println("Soldado de Mayor Vida: " + mayorVida);

    double promedioVida = calcularPromedioVida(ejercito, numSoldados);
    System.out.println("Promedio de nivel de vida: " + promedioVida);

    System.out.println("Soldados en el orden de creación:");
    armyCreationOrder(nombresEjercito); // Mostrar los nombres en el orden de creación

    ArrayList<Soldado> soldadosFila = toUnidimensional(ejercito);
    bubbleSortLife(soldadosFila); // Ordenar soldados por vida usando bubbleSort
    System.out.println("Ranking de poder (Bubble Sort):");
    showArmyInfo(soldadosFila);

    soldadosFila = toUnidimensional(ejercito);
    insertionSortLife(soldadosFila); // Ordenar soldados por vida usando insertionSort
    System.out.println("Ranking de poder (Insertion Sort):");
    showArmyInfo(soldadosFila);
}
```

```
// Crea un ejército aleatorio de soldados y los agrega a la lista
public static void crearEjercito(Random rand, int numSoldados, ArrayList<ArrayList<Soldado>> ejercito, 2 usages new *
    ArrayList<ArrayList<Boolean>> casillasOcupadas,
    ArrayList<String> nombresEjercito, int ejercitoNum) {

    int count = 0;
    while (count < numSoldados) {
        int randColumn = rand.nextInt( bound: 10);
        int randRow = rand.nextInt( bound: 10);
        if (!casillasOcupadas.get(randRow).get(randColumn)) {
            casillasOcupadas.get(randRow).set(randColumn, true);
            String nombreSoldado = "Soldado" + count + "X" + ejercitoNum;
            nombresEjercito.add(nombreSoldado); // Agregar el nombre al ArrayList
            ejercito.get(randRow).set(randColumn, new Soldado(nombreSoldado, randRow, randColumn,
                nivelVida: rand.nextInt( bound: 5) + 1));
            count++;
        }
    }
}

// Muestra el tablero de batalla
public static void showBoard(ArrayList<ArrayList<Boolean>> casillasOcupadas, ArrayList<ArrayList<Soldado>> ejercito1,
    ArrayList<ArrayList<Soldado>> ejercito2) {

    System.out.print("\n\t");
    for (char i = 'A'; i < 'K'; i++) {
        System.out.print(i + "   ");
    }
    System.out.println();
    System.out.print("   ");
    for (int l = 0; l < 12; l++) {
        System.out.print("_____");
    }
    System.out.println();
    for (int j = 0; j < 10; j++) {
        System.out.print((j + 1) + (j != 9 ? "   " : " "));
        for (int k = 0; k < 10; k++) {
            System.out.print("|");
            if (casillasOcupadas.get(j).get(k)) {
                String soldadoNombre = "";
                if (!ejercito1.get(j).isEmpty() && ejercito1.get(j).get(k) != null) {
                    soldadoNombre = "+";
                } else if (!ejercito2.get(j).isEmpty() && ejercito2.get(j).get(k) != null) {
                    soldadoNombre = "*";
                }
                System.out.print(soldadoNombre);
            } else {
                System.out.print(" ");
            }
            System.out.print("| ");
        }
        System.out.println();
        System.out.print("   ");
        for (int l = 0; l < 12; l++) {
            System.out.print("_____");
        }
        System.out.println();
    }
}
```

```
// Muestra los nombres de los soldados en el orden de creación
public static void armyCreationOrder(ArrayList<String> nombresEjercito) { 1usage new *
    for (String nombre : nombresEjercito) {
        System.out.println(nombre);
    }
    System.out.println();
}

// Convierte el ejército bidimensional en un ArrayList unidimensional
public static ArrayList<Soldado> toUnidimensional(ArrayList<ArrayList<Soldado>> soldados) { 2 usages
    ArrayList<Soldado> unidimensional = new ArrayList<>();
    for (ArrayList<Soldado> fila : soldados) {
        for (Soldado soldado : fila) {
            if (soldado != null) {
                unidimensional.add(soldado);
            }
        }
    }
    return unidimensional;
}

// Muestra la información de cada soldado en el ejército
public static void showArmyInfo(ArrayList<Soldado> ejercito) { 2 usages new *
    for (Soldado soldado : ejercito) {
        if (soldado != null) {
            System.out.print(soldado);
        }
    }
}

// Calcula la vida total de todos los soldados en el ejército
public static int calcularTotalVida(ArrayList<ArrayList<Soldado>> soldados) { 3 usages new *
    int totalVida = 0;
    for (ArrayList<Soldado> fila : soldados) {
        for (Soldado soldado : fila) {
            if (soldado != null) {
                totalVida += soldado.getNivelVida();
            }
        }
    }
    return totalVida;
}

// Calcula el promedio de vida de los soldados
public static double calcularPromedioVida(ArrayList<ArrayList<Soldado>> soldados, int numSoldados) {
    int totalVida = calcularTotalVida(soldados);
    return (double) totalVida / numSoldados;
}
```

```
// Encuentra el soldado con mayor vida en el ejército
public static String findMaxLifeSoldier(ArrayList<ArrayList<Soldado>> soldados) { 1usage new *
    Soldado max = null;
    for (ArrayList<Soldado> fila : soldados) {
        for (Soldado soldado : fila) {
            if (soldado != null && (max == null || soldado.getNivelVida() > max.getNivelVida())) {
                max = soldado;
            }
        }
    }
    return max.toString();
}

// Ordena el ejército por vida utilizando el método insertionSort
public static void insertionSortLife(ArrayList<Soldado> ejercito) { 1usage new *
    for (int i = 1; i < ejercito.size(); i++) {
        Soldado key = ejercito.get(i);
        int j = i - 1;
        while (j >= 0 && ejercito.get(j).getNivelVida() > key.getNivelVida()) {
            ejercito.set(j + 1, ejercito.get(j));
            j--;
        }
        ejercito.set(j + 1, key);
    }
}

// Ordena el ejército por vida utilizando el método bubbleSort
public static void bubbleSortLife(ArrayList<Soldado> ejercito) { 1usage new *
    for (int i = 0; i < ejercito.size() - 1; i++) {
        for (int j = 0; j < ejercito.size() - i - 1; j++) {
            if (ejercito.get(j).getNivelVida() > ejercito.get(j + 1).getNivelVida()) {
                Soldado temp = ejercito.get(j);
                ejercito.set(j, ejercito.get(j + 1));
                ejercito.set(j + 1, temp);
            }
        }
    }
}
```

### **MÉTRICA USADA: MAYOR CANTIDAD DE VIDA DE LOS EJÉRCITOS**

## **II. PRUEBAS**

***¿Con que valores comprobaste que tu práctica estuviera correcta?***

*Con valores generados y aleatorios dentro del main con ayuda de métodos au.*

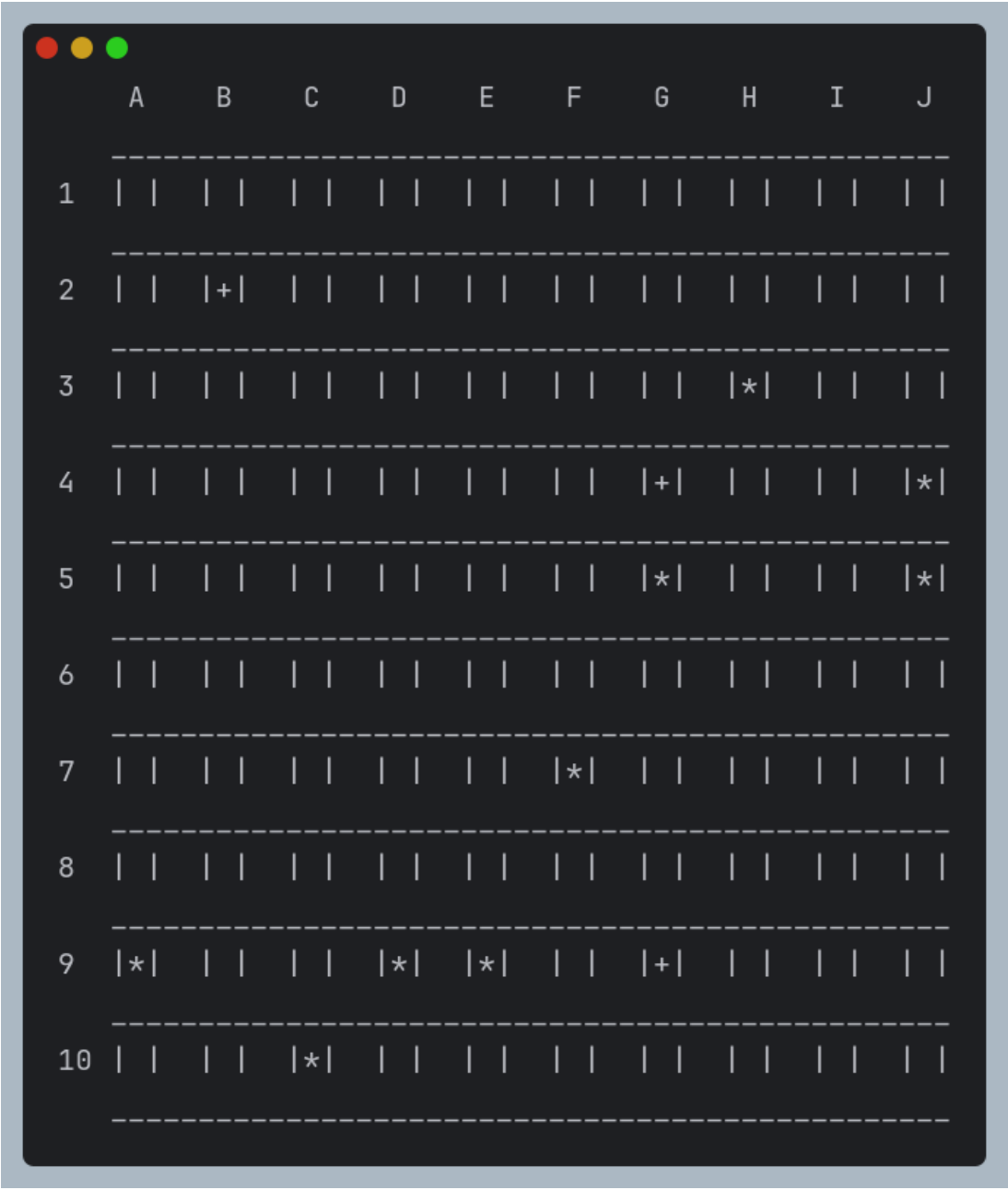
***¿Qué resultado esperabas obtener para cada valor de entrada?***

*Que aparecería en el tablero graficado por consola, la información de los soldados de ambos ejércitos, su orden de creación su ordenamiento con dos métodos inserción y burbuja y el ganador entre ambos*

*¿Qué valor o comportamiento obtuviste para cada valor de entrada?*



*Obtuve los valores esperados,y corregí algunos errores en la forma de presentación del tablero por consola.*

**EJECUCIÓN:**



	A	B	C	D	E	F	G	H	I	J
1										
2			+							
3								*		
4							+			*
5							*			*
6										
7						*				
8										
9	*			*	*		+			
10			*							



	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 9</p>

```

Ejercito 1:
Soldado de Mayor Vida: [Nombre: Soldado2X1  Fila: 9 Columna: 7  nivel de Vida: 5]

Promedio de nivel de vida: 2.3333333333333335
Soldados en el orden de creación:
Soldado0X1
Soldado1X1
Soldado2X1

Ranking de poder (Bubble Sort):
[Nombre: Soldado1X1 Fila: 2 Columna: 2  nivel de Vida: 1]
[Nombre: Soldado0X1 Fila: 4 Columna: 7  nivel de Vida: 1]
[Nombre: Soldado2X1 Fila: 9 Columna: 7  nivel de Vida: 5]
Ranking de poder (Insertion Sort):
[Nombre: Soldado1X1 Fila: 2 Columna: 2  nivel de Vida: 1]
[Nombre: Soldado0X1 Fila: 4 Columna: 7  nivel de Vida: 1]
[Nombre: Soldado2X1 Fila: 9 Columna: 7  nivel de Vida: 5]

```

```



Ejército 2:
Soldado de Mayor Vida: [Nombre: Soldado2X2  Fila: 5 Columna: 10 nivel de Vida: 4]

Promedio de nivel de vida: 2.6666666666666665
Soldados en el orden de creación:
Soldado0X2
Soldado1X2
Soldado2X2
Soldado3X2
Soldado4X2
Soldado5X2
Soldado6X2
Soldado7X2
Soldado8X2

Ranking de poder (Bubble Sort):
[Nombre: Soldado8X2 Fila: 9 Columna: 5  nivel de Vida: 1]
[Nombre: Soldado6X2 Fila: 3 Columna: 8  nivel de Vida: 2]
[Nombre: Soldado3X2 Fila: 4 Columna: 10 nivel de Vida: 2]
[Nombre: Soldado7X2 Fila: 5 Columna: 7  nivel de Vida: 2]
[Nombre: Soldado0X2 Fila: 7 Columna: 6  nivel de Vida: 3]
[Nombre: Soldado1X2 Fila: 9 Columna: 1  nivel de Vida: 3]
[Nombre: Soldado4X2 Fila: 9 Columna: 4  nivel de Vida: 3]
[Nombre: Soldado2X2 Fila: 5 Columna: 10 nivel de Vida: 4]
[Nombre: Soldado5X2 Fila: 10 Columna: 3  nivel de Vida: 4]
Ranking de poder (Insertion Sort):
[Nombre: Soldado8X2 Fila: 9 Columna: 5  nivel de Vida: 1]
[Nombre: Soldado6X2 Fila: 3 Columna: 8  nivel de Vida: 2]
[Nombre: Soldado3X2 Fila: 4 Columna: 10 nivel de Vida: 2]
[Nombre: Soldado7X2 Fila: 5 Columna: 7  nivel de Vida: 2]
[Nombre: Soldado0X2 Fila: 7 Columna: 6  nivel de Vida: 3]
[Nombre: Soldado1X2 Fila: 9 Columna: 1  nivel de Vida: 3]
[Nombre: Soldado4X2 Fila: 9 Columna: 4  nivel de Vida: 3]
[Nombre: Soldado2X2 Fila: 5 Columna: 10 nivel de Vida: 4]
[Nombre: Soldado5X2 Fila: 10 Columna: 3  nivel de Vida: 4]
```

```

Total Vida Ejército 1: 7
Total Vida Ejército 2: 24
El Ejército 2 gana la batalla.
```

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 11</p>

### III. CUESTIONARIO:

#### CAPTURAS DE LOS COMMIT:

```

git branch -M main
git add .
git status
En la rama main
Tu rama está actualizada con 'origin/main'.

Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
nuevos archivos: Soldado.java
nuevos archivos: VideoJuego2.java

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que será confirmado)
  ../RIVEROS_VILCA_LABORATORIO_04/.idea/

git commit -m "lab06-finalizado"
[main e889387] lab06-finalizado
2 files changed, 261 insertions(+)
create mode 100644 RIVEROS_VILCA_LABORATORIO_06/Soldado.java
create mode 100644 RIVEROS_VILCA_LABORATORIO_06/VideoJuego2.java

```

```

git push -u origin main
Enumerando objetos: 6, listo.
Contando objetos: 100% (6/6), listo.
Compresión delta usando hasta 12 hilos
Comprimiendo objetos: 100% (5/5), listo.
Escribiendo objetos: 100% (5/5), 2.54 KiB | 2.54 MiB/s, listo.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:rivX241/RIVEROS_VILCA_LABORATORIOS.git
   ad585ab..d68fb1d  main -> main
rama 'main' configurada para rastrear 'origin/main'.

```

Commits

main

All users All time

Commits on Oct 25, 2024

lab06-finalizado

rivX241 committed 2 minutes ago

d68fb1d

Commits on Oct 22, 2024

avance\_lab06\_gradle

rivX241 committed 3 days ago

ad585ab

Commits on Oct 18, 2024

informe05-actualizado3



rivX241 committed last week

171cfa8

informe05-actualizado3

3aff30f

Cambie la rama a main, añadí el informe al área de stage y hice un commit lab05-finalizado y realice el git push –u origin main el origin ya estaba previamente configurado para todos los laboratorios.

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLE-001	<b>Página:</b> 12

**LINK:**[https://github.com/rivX241/RIVEROS\\_VILCA\\_LABORATORIOS](https://github.com/rivX241/RIVEROS_VILCA_LABORATORIOS)

## CONCLUSIONES

*Cuando empecé a trabajar con ArrayLists bidimensionales en Java, me di cuenta de lo útiles que son para gestionar datos, especialmente cuando trato con objetos. Es como tener un tablero donde puedo acceder fácilmente a los elementos usando dos índices, lo que me facilita un montón la vida. Sin embargo, ordenar estas estructuras puede ser un poco complicado. La naturaleza bidimensional significa que tengo que pensar de manera diferente para acceder a los objetos, y eso a veces se vuelve un verdadero rompecabezas.*

*Además, en el ejercicio en el que estoy trabajando, los datos están dispersos por todas partes y no son contiguos, lo que hace que la implementación de un algoritmo de ordenación sea aún más difícil.*

## METODOLOGÍA DE TRABAJO

- Primero, leí el problema con atención y revisé todos los requisitos y restricciones para entenderlo bien y plantear una solución adecuada. Quería asegurarme de no pasar nada por alto.*
- Luego, me puse a identificar las herramientas y la lógica que necesitaba para resolverlo. Esto me ayudó a tener claro qué enfoque seguir y cómo estructurar mi código.*
- Después, codifiqué la solución y la probé*
- bé con algunos datos de entrada para ver si funcionaba como esperaba. Al principio, no todo salió perfecto, pero eso es parte del proceso.*
- Finalmente, hice algunas pruebas y corregí los errores que encontré. Siempre hay algo que ajustar, pero al final logré que todo funcionara como debería.*

## REFERENCIAS Y BIBLIOGRAFÍA

*Colocare las referencias utilizadas para el desarrollo de la práctica en formato IEEE*

E. G. Castro Gutiérrez y M. W. Aedo López, *Fundamentos de programación 2: tópicos de programación orientada a objetos*, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021. ISBN: 978-612-5035-20-2. 170 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 20.5 x 29 cm.

Rubrica:

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	2	
TOTAL		20		18	