





| | | |
|---|---|---|
|  | <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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 1</p> |

INFORME DE LABORATORIO

No 07

| INFORMACIÓN BÁSICA | | | | | |
|---|---|-----------------------------|-----------------|-----------------------|-------------------------------------|
| ASIGNATURA: | <i>Fundamentos de la Programación 2</i> | | | | |
| TÍTULO DE LA PRÁCTICA: | | | | | |
| NÚMERO DE PRÁCTICA: | <i>06</i> | AÑO LECTIVO: | <i>2024-B</i> | NRO. SEMESTRE: | <i>II</i> |
| FECHA DE PRESENTACIÓN | <i>15/11/2024</i> | HORA DE PRESENTACIÓN | <i>18:20:00</i> | | |
| INTEGRANTE (s) <i>Subia Huaicane Edson Fabricio</i> | | | | NOTA (0-20) | <i>Nota colocada por el docente</i> |
| DOCENTE(s): <i>Lino José Pinto Oppe</i> | | | | | |

| RESULTADOS Y PRUEBAS |
|--|
| <ul style="list-style-type: none"> EJERCICIOS RESUELTOS: <p>Actividad VIDEOJUEGO de SOLDADOS:</p> <ul style="list-style-type: none"> Cree un Proyecto llamado Laboratorio7 Usted deberá crear las dos clases Soldado.java y VideoJuego4.java. Puede reutilizar lo desarrollado en Laboratorios anteriores. Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero). El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para el tablero utilizar la estructura de datos más adecuada. Tendrá 2 Ejércitos (utilizar la estructura de datos más adecuada). 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 y sus puntos de vida (usar caracteres como _ y otros y 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 la métrica usada para decidir al ganador de la batalla). Hacer el programa iterativo. <p>En este enlace se encuentra mi repositorio y los commits que realicé para la creación y/o mejora de este programa: https://github.com/Q3son/Videojuego_Soldados.git</p> |

| | | |
|---|---|---|
|  | <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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 2</p> |

Mis COMMITS:

- Este es el primer commit destacable que realicé, acomodé el método para Inicializar el juego y que este reciba correctamente los algoritmos de ordenamiento en ArrayList:

Creación de Videojuego4:

- Se creó este programa para adicionar métodos de algoritmos de ordenamiento y hacer el código más eficiente, comprensible y explicativo.

Edson Fabricio Subia Huacane - 1b2e271 +174 -0

1 changed file VideoJuego_SoldadosVideojuego4.java

```

Vide...java 109 +
110 + // 8. Método para ordenar soldados de un ejército por poder (nivel de vida) - Algoritmo de selección (mejorado)
111 + public void rankingDePoder(ArrayList<Soldado> ejercito) {
112 +     for (int i = 0; i < ejercito.size() - 1; i++) {
113 +         int indexMax = i;
114 +         Soldado maxSoldado = ejercito.get(i); // Guardar el soldado actual como máximo
115 +
116 +         // Encontrar el soldado con el nivel de vida más alto en la sublista sin ordenar
117 +         for (int j = i + 1; j < ejercito.size(); j++) {
118 +             if (ejercito.get(j).getNivelVida() > maxSoldado.getNivelVida()) {
119 +                 indexMax = j;
120 +                 maxSoldado = ejercito.get(j);
121 +             }
122 +         }
123 +
124 +         // Solo hacer el intercambio si el índice del máximo ha cambiado
125 +         if (indexMax != i) {
126 +             ejercito.set(indexMax, ejercito.get(i));
127 +             ejercito.set(i, maxSoldado);
128 +         }
129 +     }
130 +
131 +     System.out.println("Ranking de poder de soldados en " + (ejercito == ejercito1 ? "Ejército 1" : "Ejército 2") + " (ordenados por nivel de vida):");
132 +     for (Soldado soldado : ejercito) {
133 +         System.out.println(soldado);
134 +     }
135 +     System.out.println();

```

- Para esta versión acomodé el método que ordena a los soldados por su poder, para que funcione correctamente.

Optimización del método que ordena a los soldados por su poder:

- Se logró una Reducción de Asignaciones: Guardamos el soldado actual como el máximo (maxSoldado) y lo comparamos en el bucle. Esto evita la necesidad de hacer múltiples llamadas get para acceder al elemento.

- Realizamos un Intercambio Condicional: Solo intercambiamos si indexMax ha cambiado, reduciendo así operaciones innecesarias.

Edson Fabricio Subia Huacane - 3527f00 +0 -1



1 changed file VideoJuego_SoldadosVideojuego4.java

```

Vide...java 137 -
138 + // 9. Método para decidir cuál ejército gana la batalla en función del total de vida
139 + public void determinarGanador() {
140 +     int vidaTotalEjercito1 = nivelVidaTotal(ejercito1);
141 +     int vidaTotalEjercito2 = nivelVidaTotal(ejercito2);
142 +
143 +     System.out.println("Vida total del Ejército 1: " + vidaTotalEjercito1);
144 +     System.out.println("Vida total del Ejército 2: " + vidaTotalEjercito2);
145 +
146 +     if (vidaTotalEjercito1 > vidaTotalEjercito2) {
147 +         System.out.println("¡Ejército 1 gana la batalla!");
148 +     } else if (vidaTotalEjercito2 > vidaTotalEjercito1) {
149 +         System.out.println("¡Ejército 2 gana la batalla!");
150 +     } else {
151 +         System.out.println("La batalla termina en empate.");
152 +     }
153 + }
154 +
155 + // 10. Método principal para ejecutar el juego
156 + public static void main(String[] args) {
157 +     Videojuego4 juego = new Videojuego4(10, 10); // Inicia el juego con 10 soldados en cada ejército
158 +     juego.mostrarTablero();
159 +
160 +     System.out.println("Soldado con mayor nivel de vida en Ejército 1: " + juego.soldadoConMayorVida(juego.ejercito1));

```

- Se actualizaron los métodos de ordenamiento y puntajes para funcionar con ArrayList

| | | |
|---|---|---|
|  | <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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 3</p> |

- **Para esta versión, cree un método para que los soldados se ordenen por su nombre, usando el algoritmo Burbuja.**

Método que ordena a los soldados por nombre: - Aplicamos el algoritmo de...

... burbuja. - Se aplicó un bucle do-while que continúa ejecutándose mientras se detecten intercambios, asegurando que los soldados queden en orden alfabético una vez que no haya más intercambios en una pasada. Se compararon los nombres de soldados adyacentes en la lista con el método compareTo, y cuando un soldado estaba en una posición incorrecta en relación con el siguiente, se intercambiaron utilizando una variable temporal.

- El uso de intercambioRealizado como bandera permitió identificar cuando ya no era necesario continuar con las pasadas, optimizando el proceso.

Edson Fabricio Subia Huaicane <esubiahu@unsa.edu.pe>
aa2fca09b459a3c12eb450e47c908fcc0069486e

23 added lines 1 removed lines

1 changed file VideoJuego_Soldados\Videojuego4.java

```

137 + // 9. Nuevo método para ordenar soldados de un ejército por nombre (algoritmo de burbuja)
138 + public void ordenarPorNombre(ArrayList<Soldado> ejercito) {
139 +     boolean intercambioRealizado;
140 +     do {
141 +         intercambioRealizado = false;
142 +         for (int i = 0; i < ejercito.size() - 1; i++) {
143 +             if (ejercito.get(i).getNombre().compareTo(ejercito.get(i + 1).getNombre()) > 0) {
144 +                 Soldado temp = ejercito.get(i);
145 +                 ejercito.set(i, ejercito.get(i + 1));
146 +                 ejercito.set(i + 1, temp);
147 +                 intercambioRealizado = true;
148 +             }
149 +         }
150 +     } while (intercambioRealizado);
151 +
152 +     System.out.println("Soldados en " + (ejercito == ejercito1 ? "Ejército 1" : "Ejército 2") + " ordenados por nombre:");
153 +     for (Soldado soldado : ejercito) {
154 +         System.out.println(soldado);
155 +     }
156 +     System.out.println();
157 + }

```

- **Para la versión final definitiva, logré que el programa funcione de forma iterativa**

Programa Iterativo:

- Se agregó un bucle iterativo al método main para permitir al usuario jugar otra vez o salir del juego.



Edson Fabricio Subia Huaicane 5c410e3 +26 -16

1 changed file VideoJuego_Soldados\Videojuego4.java

```

178 + Scanner scanPro = new Scanner(System.in);
179 + boolean jugarOtraVez = true;
180 +
181 + while (jugarOtraVez) {
182 +     Videojuego4 juego = new Videojuego4(10, 10); // Inicia el juego con 10 soldados en cada ejército
183 +     juego.mostrarTablero();
184 +
185 +     System.out.println("Soldado con mayor nivel de vida en Ejército 1: " + juego.soldadoConMayorVida(juego.ejercito1));
186 +     System.out.println("Soldado con mayor nivel de vida en Ejército 2: " + juego.soldadoConMayorVida(juego.ejercito2));
187 +
188 +     System.out.println("Promedio de nivel de vida en Ejército 1: " + juego.promedioNivelVida(juego.ejercito1));
189 +     System.out.println("Promedio de nivel de vida en Ejército 2: " + juego.promedioNivelVida(juego.ejercito2));
190 +
191 +     juego.mostrarDatosEjercito(juego.ejercito1, "Ejército 1");
192 +     juego.mostrarDatosEjercito(juego.ejercito2, "Ejército 2");
193 +
194 +     juego.rankingDePoder(juego.ejercito1);
195 +     juego.rankingDePoder(juego.ejercito2);
196 +
197 +     juego.determinarGanador();
198 +
199 +     System.out.print("¿Quieres jugar otra vez? (s/n): ");
200 +     String respuesta = scanPro.nextLine().trim().toLowerCase();
201 +     jugarOtraVez = respuesta.equals("s");
202 + }
203 + System.out.println("¡Gracias por jugar!");

```

| | | |
|---|---|---|
|  | <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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 4</p> |

En la siguiente sección mostraré el código fuente y ejecución de la versión final de mi código fuente del programa, trabajado en Visual Studio, en cada captura de pantalla se visualizará el buen funcionamiento de los nuevos métodos adicionados y fundamentados correctamente. (El código fuente se visualiza mucho mejor en mi repositorio)

```

You, 3 days ago | 1 author (You)
1 import java.util.*;
You, 3 days ago | 1 author (You)
2 // BY: SUBIA_EDSON_FP2
3
4 public class Videojuego4 {
5
6     private static final int TAMAÑO_TABLERO = 10; // Tablero de 10x10
7     private ArrayList<ArrayList<Soldado>> tablero;
8     private ArrayList<Soldado> ejercito1;
9     private ArrayList<Soldado> ejercito2;
10
11     // 1. Constructor para inicializar el juego con soldados en ambos ejércitos
12     public Videojuego4(int cantidadSoldadosEjercito1, int cantidadSoldadosEjercito2) {
13         tablero = new ArrayList<>(TAMAÑO_TABLERO);
14         for (int i = 0; i < TAMAÑO_TABLERO; i++) {
15             tablero.add(new ArrayList<>(TAMAÑO_TABLERO));
16             for (int j = 0; j < TAMAÑO_TABLERO; j++) {
17                 tablero.get(i).add(e:null);
18             }
19         }
20
21         ejercito1 = new ArrayList<>(cantidadSoldadosEjercito1);
22         ejercito2 = new ArrayList<>(cantidadSoldadosEjercito2);
23
24         inicializarEjercito(ejercito1, cantidadSoldadosEjercito1, etiquetaEjercito:"X1");
25         inicializarEjercito(ejercito2, cantidadSoldadosEjercito2, etiquetaEjercito:"X2");
26     }
27
28     // 2. Método para inicializar un ejército con soldados aleatorios
29     private void inicializarEjercito(ArrayList<Soldado> ejercito, int cantidad, String etiquetaEjercito) {
30         Random random = new Random();
31         int soldadosCreados = 0;
32
33         while (soldadosCreados < cantidad) {
34             int fila = random.nextInt(TAMAÑO_TABLERO);
35             int columna = random.nextInt(TAMAÑO_TABLERO);
36
37             if (tablero.get(fila).get(columna) == null) { // Verifica que el espacio esté vacío
38                 String nombre = "Soldado" + soldadosCreados + etiquetaEjercito;
39                 int nivelVida = random.nextInt(bound:5) + 1;
40                 Soldado soldado = new Soldado(nombre, nivelVida, fila, columna);
41
42                 tablero.get(fila).set(columna, soldado); // Asignar el soldado al tablero
43                 ejercito.add(soldado); // Guardar el soldado en el ejército
44                 soldadosCreados++;
45             }
46         }
47
48     // 3. Método para mostrar el tablero en la consola
49     public void mostrarTablero() {
50         System.out.println(x:"Tablero de Soldados:");
51         String lineaDivisoria = new String(new char[110]).replace(target:"\0", replacement:"-");
52
53         for (int fila = 0; fila < TAMAÑO_TABLERO; fila++) {
54             System.out.println(lineaDivisoria);
55             System.out.print(s:"|");
56
57             for (int columna = 0; columna < TAMAÑO_TABLERO; columna++) {
58                 String celda = " "; // Espacio vacío
59                 Soldado soldado = tablero.get(fila).get(columna);
60
61                 if (soldado != null) {
62                     celda = String.format(format:"%-10s", soldado.getNombre()); // Nombre del soldado
63                 }
64
65                 System.out.print(celda + "|");
66             }
67             System.out.println();
68         }
69         System.out.println(lineaDivisoria); // Línea divisoria final

```

```

70     }
71
72     // 4. Método para mostrar datos del soldado con mayor nivel de vida de un ejército
73     public Soldado soldadoConMayorVida(ArrayList<Soldado> ejercito) {
74         Soldado maxSoldado = ejercito.get(index:0);
75         for (Soldado soldado : ejercito) {
76             if (soldado.getNivelVida() > maxSoldado.getNivelVida()) {
77                 maxSoldado = soldado;
78             }
79         }
80         return maxSoldado;
81     }
82
83     // 5. Método para calcular el promedio de nivel de vida de un ejército
84     public double promedioNivelVida(ArrayList<Soldado> ejercito) {
85         int suma = 0;
86         for (Soldado soldado : ejercito) {
87             suma += soldado.getNivelVida();
88         }
89         return (double) suma / ejercito.size();
90     }
91
92     // 6. Método para calcular el nivel de vida total de un ejército
93     public int nivelVidaTotal(ArrayList<Soldado> ejercito) {
94         int total = 0;
95         for (Soldado soldado : ejercito) {
96             total += soldado.getNivelVida();
97         }
98         return total;
99     }
100
101     // 7. Método para mostrar datos de los soldados de un ejército
102     public void mostrarDatosEjercito(ArrayList<Soldado> ejercito, String nombreEjercito) {
103         System.out.println("Datos de los soldados en " + nombreEjercito + " en orden de creación:");
104         for (Soldado soldado : ejercito) {

```

```

104         for (Soldado soldado : ejercito) {
105             System.out.println(soldado);
106         }
107         System.out.println();
108     }
109
110     // 8. Método para ordenar soldados de un ejército por poder (nivel de vida) - Algoritmo de selección (mejorado)
111     public void rankingDePoder(ArrayList<Soldado> ejercito) {
112         for (int i = 0; i < ejercito.size() - 1; i++) {
113             int indexMax = i;
114             Soldado maxSoldado = ejercito.get(i); // Guardar el soldado actual como máximo
115
116             // Encontrar el soldado con el nivel de vida más alto en la sublista sin ordenar
117             for (int j = i + 1; j < ejercito.size(); j++) {
118                 if (ejercito.get(j).getNivelVida() > maxSoldado.getNivelVida()) {
119                     indexMax = j;
120                     maxSoldado = ejercito.get(j);
121                 }
122             }
123
124             // Solo hacer el intercambio si el índice del máximo ha cambiado
125             if (indexMax != i) {
126                 ejercito.set(indexMax, ejercito.get(i));
127                 ejercito.set(i, maxSoldado);
128             }
129         }
130
131         System.out.println("Ranking de poder de soldados en " + (ejercito == ejercito1 ? "Ejército 1" : "Ejército 2") + " (ordenados por niv
132         for (Soldado soldado : ejercito) {
133             System.out.println(soldado);
134         }
135         System.out.println();
136     }
137
138     // 9. Nuevo método para ordenar soldados de un ejército por nombre (algoritmo de burbuja)
139     public void ordenarPorNombre(ArrayList<Soldado> ejercito) {

```

```
138 public void ordenarPorNombre(ArrayList<Soldado> ejercito) {
139     boolean intercambioRealizado;
140     do {
141         intercambioRealizado = false;
142         for (int i = 0; i < ejercito.size() - 1; i++) {
143             if (ejercito.get(i).getNombre().compareTo(ejercito.get(i + 1).getNombre()) > 0) {
144                 Soldado temp = ejercito.get(i);
145                 ejercito.set(i, ejercito.get(i + 1));
146                 ejercito.set(i + 1, temp);
147                 intercambioRealizado = true;
148             }
149         }
150     } while (intercambioRealizado);
151
152     System.out.println("Soldados en " + (ejercito == ejercito1 ? "Ejército 1" : "Ejército 2") + " ordenados por nombre.");
153     for (Soldado soldado : ejercito) {
154         System.out.println(soldado);
155     }
156     System.out.println();
157 }
158
159 // 10. Método para decidir cuál ejército gana la batalla en función del total de vida
160 public void determinarGanador() {
161     int vidaTotalEjercito1 = nivelVidaTotal(ejercito1);
162     int vidaTotalEjercito2 = nivelVidaTotal(ejercito2);
163
164     System.out.println("Vida total del Ejército 1: " + vidaTotalEjercito1);
165     System.out.println("Vida total del Ejército 2: " + vidaTotalEjercito2);
166
167     if (vidaTotalEjercito1 > vidaTotalEjercito2) {
168         System.out.println(x:"¡Ejército 1 gana la batalla!");
169     } else if (vidaTotalEjercito2 > vidaTotalEjercito1) {
170         System.out.println(x:"¡Ejército 2 gana la batalla!");
171     } else {
172         System.out.println(x:"La batalla termina en empate.");
173     }
```

```
174 }
175
176 // 10. Método principal para ejecutar el juego
177 public static void main(String[] args) {
178     Scanner scanPro = new Scanner(System.in);
179     boolean jugarOtraVez = true;
180
181     while (jugarOtraVez) {
182         Videojuego4 juego = new Videojuego4(cantidadSoldadosEjercito1:10, cantidadSoldadosEjercito2:10); // Inicia el juego con 10 soldados en cada ejército
183         juego.mostrarTablero();
184
185         System.out.println("Soldado con mayor nivel de vida en Ejército 1: " + juego.soldadoConMayorVida(juego.ejercito1));
186         System.out.println("Soldado con mayor nivel de vida en Ejército 2: " + juego.soldadoConMayorVida(juego.ejercito2));
187
188         System.out.println("Promedio de nivel de vida en Ejército 1: " + juego.promedioNivelVida(juego.ejercito1));
189         System.out.println("Promedio de nivel de vida en Ejército 2: " + juego.promedioNivelVida(juego.ejercito2));
190
191         juego.mostrarDatosEjercito(juego.ejercito1, nombreEjercito:"Ejército 1");
192         juego.mostrarDatosEjercito(juego.ejercito2, nombreEjercito:"Ejército 2");
193
194         juego.rankingDePoder(juego.ejercito1);
195         juego.rankingDePoder(juego.ejercito2);
196
197         juego.determinarGanador();
198
199         System.out.print(s:"¿Quieres jugar otra vez? (s/n): ");
200         String respuesta = scanPro.nextLine().trim().toLowerCase();
201         jugarOtraVez = respuesta.equals(anobject:"s");
202     }
203     System.out.println(x:"¡Gracias por jugar!");
204 }
205 }
```

EJECUCIÓN DEL PROGRAMA (v6.0.1): "JUEGO DE NAVE EBAS"

```
PS C:\Users\Edson> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
juego3'
Tablero de Soldados:
-----
|           |           |           |           |           |           |           |           |
|Soldado1X1|           |           |           |           |           |           |           |
|           |           |           |           |           |           |           |           |
|           |           |           |           |           |           |           |           |
|           |           |           |           |Soldado6X1|Soldado2X2|           |           |Soldado0X1|
|           |Soldado8X1|           |           |           |           |           |           |Soldado6X2|
|Soldado5X1|           |           |           |           |           |           |Soldado1X2|           |
|           |           |           |           |           |           |           |           |Soldado8X2|
|           |           |Soldado0X2|           |           |           |           |           |           |
|           |Soldado9X1|           |           |Soldado7X1|           |           |           |           |
|           |           |Soldado3X1|           |Soldado9X2|           |           |Soldado7X2|           |
-----
Soldado con mayor nivel de vida en Ejército 1: Nombre: Soldado0X1, Nivel de vida: 5, Posición: (3, 9)
Soldado con mayor nivel de vida en Ejército 2: Nombre: Soldado0X2, Nivel de vida: 5, Posición: (7, 2)
Promedio de nivel de vida en Ejército 1: 2.9
Promedio de nivel de vida en Ejército 2: 3.2
```

Datos de los soldados en Ejército 1 en orden de creación:

Nombre: Soldado0X1, Nivel de vida: 5, Posición: (8, 2)
Nombre: Soldado1X1, Nivel de vida: 1, Posición: (6, 6)
Nombre: Soldado2X1, Nivel de vida: 2, Posición: (6, 2)
Nombre: Soldado3X1, Nivel de vida: 5, Posición: (2, 2)
Nombre: Soldado4X1, Nivel de vida: 1, Posición: (1, 3)
Nombre: Soldado5X1, Nivel de vida: 2, Posición: (5, 4)
Nombre: Soldado6X1, Nivel de vida: 3, Posición: (7, 1)
Nombre: Soldado7X1, Nivel de vida: 2, Posición: (5, 9)
Nombre: Soldado8X1, Nivel de vida: 1, Posición: (2, 6)
Nombre: Soldado9X1, Nivel de vida: 2, Posición: (3, 5)

Datos de los soldados en Ejército 2 en orden de creación:

Nombre: Soldado0X2, Nivel de vida: 4, Posición: (4, 9)
Nombre: Soldado1X2, Nivel de vida: 1, Posición: (6, 8)
Nombre: Soldado2X2, Nivel de vida: 5, Posición: (8, 1)
Nombre: Soldado3X2, Nivel de vida: 4, Posición: (9, 4)
Nombre: Soldado4X2, Nivel de vida: 5, Posición: (4, 8)
Nombre: Soldado5X2, Nivel de vida: 5, Posición: (0, 8)
Nombre: Soldado6X2, Nivel de vida: 5, Posición: (1, 9)
Nombre: Soldado7X2, Nivel de vida: 5, Posición: (0, 6)
Nombre: Soldado8X2, Nivel de vida: 1, Posición: (9, 6)
Nombre: Soldado9X2, Nivel de vida: 4, Posición: (7, 7)



Ranking de poder de soldados en Ejército 1 (ordenados por nivel de vida):

Nombre: Soldado0X1, Nivel de vida: 5, Posición: (8, 2)
Nombre: Soldado3X1, Nivel de vida: 5, Posición: (2, 2)
Nombre: Soldado6X1, Nivel de vida: 3, Posición: (7, 1)
Nombre: Soldado5X1, Nivel de vida: 2, Posición: (5, 4)
Nombre: Soldado2X1, Nivel de vida: 2, Posición: (6, 2)
Nombre: Soldado7X1, Nivel de vida: 2, Posición: (5, 9)
Nombre: Soldado9X1, Nivel de vida: 2, Posición: (3, 5)
Nombre: Soldado1X1, Nivel de vida: 1, Posición: (6, 6)
Nombre: Soldado8X1, Nivel de vida: 1, Posición: (2, 6)
Nombre: Soldado4X1, Nivel de vida: 1, Posición: (1, 3)

Ranking de poder de soldados en Ejército 2 (ordenados por nivel de vida):

Nombre: Soldado2X2, Nivel de vida: 5, Posición: (8, 1)
Nombre: Soldado4X2, Nivel de vida: 5, Posición: (4, 8)
Nombre: Soldado5X2, Nivel de vida: 5, Posición: (0, 8)
Nombre: Soldado6X2, Nivel de vida: 5, Posición: (1, 9)
Nombre: Soldado7X2, Nivel de vida: 5, Posición: (0, 6)
Nombre: Soldado0X2, Nivel de vida: 4, Posición: (4, 9)
Nombre: Soldado3X2, Nivel de vida: 4, Posición: (9, 4)
Nombre: Soldado9X2, Nivel de vida: 4, Posición: (7, 7)
Nombre: Soldado8X2, Nivel de vida: 1, Posición: (9, 6)
Nombre: Soldado1X2, Nivel de vida: 1, Posición: (6, 8)

Vida total del Ejército 1: 24
Vida total del Ejército 2: 39
¡Ejército 2 gana la batalla!
¿Quieres jugar otra vez? (s/n):

| | | |
|--|--|---|
|  | <p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p> |  |
| <p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 8</p> |

¿Con qué valores comprobaste que tu práctica estuviera correcta?



Comprobé la práctica utilizando valores aleatorios al generar soldados para dos ejércitos. Asigné nombres autogenerados como "Soldado0X1", "Soldado1X1", variando aleatoriamente la fila y columna de cada soldado y asegurando que no se repitieran las posiciones en el tablero. También generé valores de nivel de vida aleatorios entre 1 y 5 para verificar que el tablero se mostrara correctamente y que cada ejército mantuviera sus soldados en posiciones distintas. Para probar el ranking, utilicé los métodos de ordenamiento de selección y burbuja en soldados de cada ejército.

¿Qué resultado esperabas obtener para cada valor de entrada? Esperaba que el programa tuviera el siguiente comportamiento:

- Al ordenar soldados por nivel de vida (método de selección y burbuja): Los soldados de cada ejército se ordenarían correctamente de acuerdo con sus niveles de vida, mostrando el orden descendente esperado.
- Al mostrar el soldado con mayor nivel de vida por ejército: Se esperaba que se identificara correctamente el soldado con el mayor nivel de vida en cada ejército.
- Al calcular el promedio y el total de puntos de vida por ejército: El programa mostraría los valores adecuados para el promedio de vida y el total de vida de cada ejército.
- Al determinar el ejército ganador: Se esperaba que el ejército con mayor total de puntos de vida fuera declarado como ganador.

¿Qué valor o comportamiento obtuviste para cada valor de entrada? Los resultados obtenidos fueron los esperados:

- Al ordenar soldados por nivel de vida: Los soldados de ambos ejércitos se mostraron ordenados correctamente en función de sus niveles de vida, validando la eficacia del algoritmo de selección y burbuja para el ranking.
- Al mostrar el soldado con mayor nivel de vida: El programa identificó correctamente el soldado con el mayor nivel de vida en cada ejército.
- Al calcular el promedio y total de puntos de vida por ejército: Los resultados mostraron los valores esperados, confirmando que los métodos de cálculo se implementaron correctamente.
- Al determinar el ejército ganador: El programa identificó correctamente al ejército con el mayor total de puntos de vida, cumpliendo con las métricas definidas para decidir el ganador de la batalla.

| | | |
|---|---|---|
|  | <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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 9</p> |



1. CUESTIONARIO:

Los datos más importantes que consideré para realizar el código del juego de Soldados, especialmente en relación con las nuevas funcionalidades y requisitos, fueron los siguientes:

2. **Atributos del Soldado:** Cada soldado cuenta con atributos clave como nombre, fila, columna y nivel de vida. Estos atributos son fundamentales para definir las características de cada soldado en el tablero. El nivel de vida es especialmente crítico, ya que determina la resistencia de cada soldado en la batalla y, por ende, influye en la estrategia del juego.
3. **ArrayList bidimensional:** Utilizar un arreglo bidimensional para almacenar los objetos de tipo Soldado es esencial. Este arreglo simula el tablero de juego, permitiendo que los soldados se ubiquen en posiciones específicas y facilitando la gestión de sus interacciones durante la partida. Además, esta estructura es crucial para aplicar los métodos de ordenamiento que permiten organizar a los soldados de acuerdo a su nivel de vida o posición.
4. **Generación aleatoria de soldados:** La creación de soldados con valores aleatorios, como el nivel de vida y las posiciones, asegura un entorno dinámico y desafiante para el juego. Esto incluye garantizar que no haya dos soldados en la misma posición del tablero, lo que agrega un nivel de estrategia adicional al juego.
5. **Métodos de ordenamiento:** La implementación de métodos de ordenamiento es crucial para organizar a los soldados según diferentes criterios:
 - **Ordenamiento por selección:** Se utilizó para ordenar a los soldados por nivel de vida, permitiendo identificar rápidamente cuáles son los más débiles y cuáles están en mejor estado. Esto es útil para tomar decisiones estratégicas durante la partida.
 - **Ordenamiento por burbuja:** Este método se utilizó para organizar a los soldados por nombre, facilitando la búsqueda y gestión del ejército, lo que mejora la experiencia del usuario.
6. **Cálculo de promedios y totales:** Los métodos que calculan el promedio y total de puntos de vida por ejército son esenciales para evaluar la fuerza general de cada ejército. Esto ayuda a determinar quién ganará la batalla, basándose en el total de vida de los soldados, y aporta a la lógica del juego.
7. **Determinación del ejército ganador:** Definir qué ejército ganará la batalla es fundamental para la dinámica del juego. Se basa en la métrica del total de puntos de vida, permitiendo a los jugadores evaluar el rendimiento de sus ejércitos de manera clara y concisa.
8. **Interacción con el usuario:** Todos estos elementos mejoran la experiencia del usuario, al permitirle ver los soldados organizados y calcular los resultados de manera eficiente. Los jugadores pueden evaluar fácilmente la capacidad de su ejército y tomar decisiones informadas, enriqueciendo así la jugabilidad.

CONCLUSIONES

Colocar las conclusiones, apreciaciones reflexivas, opiniones finales a cerca de los resultados obtenidos de la sesión de laboratorio.

| | | |
|---|---|---|
|  | <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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> | | |
| <p>Aprobación: 2022/03/01</p> | <p>Código: GUIA-PRLE-001</p> | <p>Página: 10</p> |

La implementación del juego de Soldados ha sido exitosa tanto en su estructura como en la interacción que ofrece al usuario. La incorporación de algoritmos de ordenamiento ha permitido gestionar los atributos de los soldados, como el nivel de vida y el nombre, de manera eficiente. Al ordenar por nivel de vida utilizando el método de selección, el jugador puede visualizar fácilmente qué soldados tienen mayor resistencia, lo que facilita la toma de decisiones estratégicas. Por otro lado, el ordenamiento por nombre mediante el método de burbuja mejora la gestión alfabética de los soldados, optimizando la experiencia de búsqueda y análisis.

METODOLOGÍA DE TRABAJO

Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el procedimiento/secuencia de pasos en forma general.

- a) **Comprensión del problema:** En esta etapa, revisé cada una de las actividades propuestas, identificando cuidadosamente las restricciones y los objetivos a alcanzar.
- b) **Diseño del algoritmo:** Planifiqué la secuencia lógica necesaria para implementar la solución, aplicando los conocimientos adquiridos en Fundamentos de Programación I y II.
- c) **Codificación:** Procedí a implementar los programas solicitados, asegurándome de utilizar correctamente los arreglos y métodos.
- d) **Pruebas:** Realicé pruebas adicionales para verificar que el código funcionara de manera correcta con diferentes casos de prueba.

REFERENCIAS Y BIBLIOGRAFÍA

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

M. W. Aedo López, *Fundamentos de programación I: Java Básico*, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, jul. 2019. ISBN: 978-612-4337-55-0. 116 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 21 x 29.7 cm

https://github.com/LINOPINTO2023/FundProq2/blob/main/entregaLaboratorio01/Hilacondo_Emanuel_LABORATORIO_01.pdf

https://github.com/Q3son/Videojuego_Soldados.git

RÚBRICA PARA EL CONTENIDO DEL INFORME Y DEMOSTRACIÓN

El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumplió con el ítem correspondiente.

Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.

El alumno debe autocalificarse en la columna Estudiante de acuerdo con la siguiente tabla:

Tabla 1: Niveles de desempeño

| Nivel | | | | |
|--------|----------------------|-----------------|--------------------|---------------------|
| Puntos | Insatisfactorio 25 % | En Proceso 50 % | Satisfactorio 75 % | Sobresaliente 100 % |
| 2.0 | 0.5 | 1.0 | 1.5 | 2.0 |
| 4.0 | 1.0 | 2.0 | 3.0 | 4.0 |

| 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 | 3 | |
| TOTAL | | 20 | 8 | 19 | |