
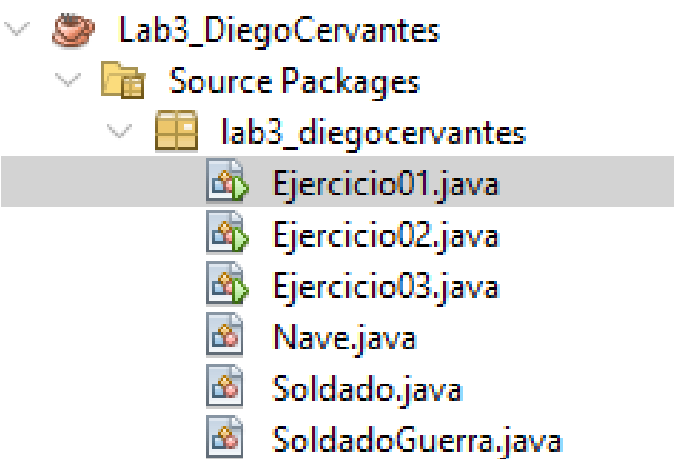
	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	<i>LABORATORIO FUNDAMENTOS PROGRAMACIÓN 2</i>				
TÍTULO DE LA PRÁCTICA:	<i>ARREGLOS DE OBJETOS</i>				
NÚMERO DE PRÁCTICA:	<i>03</i>	AÑO LECTIVO:	<i>2024</i>	NRO. SEMESTRE:	<i>//</i>
FECHA DE PRESENTACIÓN	<i>05/10/2024</i>	HORA DE PRESENTACIÓN	<i>23:55</i>		
INTEGRANTE (s) <i>Diego Aristides Cervantes Apaza</i>				NOTA (0-20)	
DOCENTE(s): <i>Lino Pinto</i>					

RESULTADOS Y PRUEBAS	
<p>I. EJERCICIOS RESUELTOS:</p> <p>Ejercicio 1. Analice, complete y pruebe el Código de la clase DemoBatalla</p> <div align="center" data-bbox="389 1361 1066 1818">  </div>	

```
1 //Autor: Diego Aristides Cervantes Apaza
2 //Problema: Completar el juego de naves
3 package lab3_diegocervantes;
4
5 import java.util.ArrayList;
6 import java.util.Scanner;
7 import java.util.Random;
8
9 public class Ejercicio01 {
10     public static void main(String[] args) {
11         Nave[] misNaves = new Nave[10];
12         Scanner sc = new Scanner(System.in);
13         String nomb, col;
14         int fil, punt;
15         boolean est;
16
17         // Captura de datos para cada nave
18         for (int i = 0; i < misNaves.length; i++) {
19             System.out.println("Nave " + (i + 1));
20             System.out.print("Nombre: ");
21             nomb = sc.next();
22             System.out.print("Fila: ");
23             fil = sc.nextInt();
24             System.out.print("Columna: ");
25             col = sc.next();
26             System.out.print("Estado (true/false): ");
27             est = sc.nextBoolean();
28             System.out.print("Puntos: ");
29             punt = sc.nextInt();
30
31             // Se crea un objeto Nave y se asigna su referencia a misNaves
32             misNaves[i] = new Nave();
33             misNaves[i].setNombre(nomb);
34             misNaves[i].setFila(fil);
35             misNaves[i].setColumna(col);
36             misNaves[i].setEstado(est);
37             misNaves[i].setPuntos(punt);
38         }
39
40         // Mostrar naves
41         System.out.println("\nNaves creadas:");
42         mostrarNaves(misNaves);
43
44         // Mostrar naves por nombre
```

```
45     mostrarPorNombre(misNaves);
46
47     // Mostrar naves por puntos
48     mostrarPorPuntos(misNaves);
49
50     // Mostrar la nave con más puntos
51     Nave naveMayorPuntos = naveConMasPuntos(misNaves);
52     System.out.println("\nNave con más puntos:");
53     System.out.println(naveMayorPuntos.getNombre() + " - Puntos: " + naveMayorPuntos.getPuntos());
54
55     // Desordenar naves aleatoriamente
56     Nave[] navesDesordenadas = desordenarNaves(misNaves);
57     System.out.println("\nNaves desordenadas:");
58     mostrarNaves(navesDesordenadas);
59 }
60
61 // Método para mostrar todas las naves
62 public static void mostrarNaves(Nave[] flota) {
63     for (Nave nave : flota) {
64         System.out.println(nave.getNombre() + " | Fila: " + nave.getFila() + " | Columna: " + nave.getColumna() +
65             " | Estado: " + nave.getEstado() + " | Puntos: " + nave.getPuntos());
66     }
67 }
68
69 // Método para mostrar todas las naves con un nombre dado
70 public static void mostrarPorNombre(Nave[] flota) {
71     Scanner sc = new Scanner(System.in);
72     System.out.print("\nIngrese nombre para buscar: ");
73     String nombreBuscado = sc.next();
74
75     for (Nave nave : flota) {
76         if (nave.getNombre().equalsIgnoreCase(nombreBuscado)) {
77             System.out.println(nave.getNombre() + " | Fila: " + nave.getFila() + " | Columna: " + nave.getColumna() +
78                 " | Estado: " + nave.getEstado() + " | Puntos: " + nave.getPuntos());
79         }
80     }
81 }
82
83 // Método para mostrar todas las naves con puntos menores o iguales a un valor dado
84 public static void mostrarPorPuntos(Nave[] flota) {
85     Scanner sc = new Scanner(System.in);
86     System.out.print("\nIngrese puntos máximos: ");
```

```
87     int puntosMax = sc.nextInt();
88
89     for (Nave nave : flota) {
90         if (nave.getPuntos() <= puntosMax) {
91             System.out.println(nave.getNombre() + " | Fila: " + nave.getFila() + " | Columna: " + nave.getColumna() +
92                                 " | Estado: " + nave.getEstado() + " | Puntos: " + nave.getPuntos());
93         }
94     }
95 }
96
97 // Método que devuelve la nave con mayor número de puntos
98 public static Nave naveConMasPuntos(Nave[] flota) {
99     Nave naveMayor = flota[0];
100    for (Nave nave : flota) {
101        if (nave.getPuntos() > naveMayor.getPuntos()) {
102            naveMayor = nave;
103        }
104    }
105    return naveMayor;
106 }
107
108 // Método que desordena las naves aleatoriamente
109 public static Nave[] desordenarNaves(Nave[] flota) {
110    Random rand = new Random();
111    Nave[] flotaDesordenada = flota.clone(); // Clonar para no modificar el arreglo original
112
113    for (int i = 0; i < flotaDesordenada.length; i++) {
114        int randomIndex = rand.nextInt(flotaDesordenada.length);
115        // Intercambiar posiciones
116        Nave temp = flotaDesordenada[i];
117        flotaDesordenada[i] = flotaDesordenada[randomIndex];
118        flotaDesordenada[randomIndex] = temp;
119    }
120    return flotaDesordenada;
121 }
122 }
```

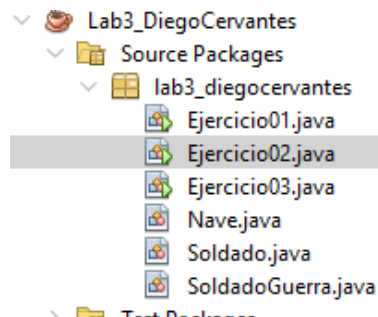
```

1  //Autor: Diego Aristides Cervantes Apaza
2  //Problema: Completar el juego de naves
3  package lab3_diego cervantes;
4      public class Nave {
5          private String nombre;
6          private int fila;
7          private String columna;
8          private boolean estado;
9          private int puntos;
10         // Metodos mutadores
11         public void setNombre( String n){
12             nombre = n;
13         }
14         public void setFila(int f){
15             fila = f;
16         }
17         public void setColumna(String c){
18             columna = c;
19         }
20         public void setEstado(boolean e){
21             estado = e;
22         }
23         public void setPuntos(int p){
24             puntos = p;
25         }
26         //Metodos accesorios
27         public String getNombre(){
28             return nombre;
29         }
30         public int getFila(){
31             return fila;
32         }
33         public String getColumna(){
34             return columna;
35         }
36
37         public boolean getEstado(){
38             return estado;
39         }
40
41         public int getPuntos(){
42             return puntos;
43         }
44     }


```

EJERCICIO 02:

Escribir un programa donde se creen 5 soldados considerando su nombre y nivel de vida. Ingresar sus datos y después mostrarlos. Restricción: aplicar arreglos con objetos



```
1 //Autor: Diego Aristides Cervantes Apaza
2 //Problema: Escribir un programa donde se creen 5 soldados considerando
3 //su nombre y nivel de vida. Ingresar sus datos y después mostrarlos.
4 //Restricción: aplicar arreglos con objetos
5
6 package lab3_diegocervantes;
7
8 import java.util.Scanner;
9 import java.util.Random;
10
11 public class Ejercicio02 {
12
13     public static void main(String[] args) {
14
15         //Declaración de objetos para las clases de ingreso de datos y obtención de aleatorios
16         Scanner scan = new Scanner(System.in);
17         Random num = new Random();
18
19         //Creación de arreglo de objetos Soldado
20         Soldado[] soldados = new Soldado[5];
21
22         //Ingreso de datos de nombres
23         System.out.println("Ingrese el nombre de los soldados:");
24
25         //Bucle para almacenar nombres ingresados y asignar vida aleatoria
26         for (int i = 0; i < soldados.length; i++) {
27             System.out.print("Nombre del soldado " + (i + 1) + ": ");
28             String nombre = scan.nextLine();
29             int vida = num.nextInt(5) + 1; // Genera vida aleatoria entre 1 y 5
30             soldados[i] = new Soldado(nombre, vida); // Crea un objeto Soldado y lo almacena en el arreglo
31         }
32
33         //Bucle para imprimir los datos de los soldados previamente ingresados y generados respectivamente
34         System.out.println("\nDatos de los soldados:");
35         for (int i = 0; i < soldados.length; i++) {
36             System.out.println("El nombre del soldado " + (i + 1) + " es " +
37                 soldados[i].getNombre() + " y su vida es " + soldados[i].getVida() + ".");
38         }
39     }
40 }
```

	<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: 7</p>

```

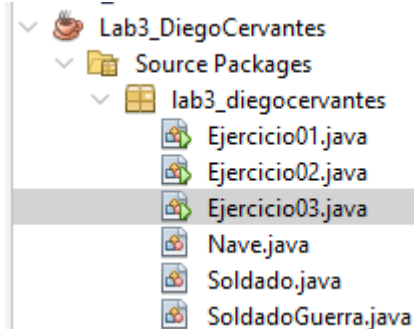
1  //Autor: Diego Aristides Cervantes Apaza
2  //Problema: Escribir un programa donde se creen 5 soldados considerando
3  //su nombre y nivel de vida. Ingresar sus datos y después mostrarlos.
4  //Restricción: aplicar arreglos con objetos
5
6
7  package lab3_diegocervantes;
8
9  public class Soldado {
10     private String nombre;
11     private int vida;
12
13     // Constructor de la clase Soldado
14     public Soldado(String nombre, int vida) {
15         this.nombre = nombre;
16         this.vida = vida;
17     }
18
19     // Método para obtener el nombre del soldado
20     public String getNombre() {
21         return nombre;
22     }
23
24     // Método para obtener la vida del soldado
25     public int getVida() {
26         return vida;
27     }
28 }

```

EJERCICIO 03:

Problema: Escribir un programa donde se creen 2 ejércitos, cada uno con un número aleatorio de soldados entre 1 y 5, considerando sólo su nombre. Sus datos se inicializan automáticamente con nombres tales como "Soldado0", "Soldado1", etc. Luego de crear los 2 ejércitos se deben mostrar los datos de todos los soldados de ambos ejércitos e indicar qué ejército fue el ganador.



Restricción: aplicar arreglos DE OBJETOS y métodos para inicializar los ejércitos, mostrar ejército y mostrar ejército ganador. La métrica a aplicar para indicar el ganador es el mayor número de soldados de cada ejército, puede haber empates. APLICAR ARREGLO DE OBJETOS.



```
1 //Autor: Diego Aristides Cervantes Apaza
2 //Problema: Escribir un programa donde se creen 2 ejércitos, cada uno con un número
3 //aleatorio de soldados entre 1 y 5, considerando sólo su nombre. Sus datos se
4 //inicializan automáticamente con nombres tales como "Soldado0", "Soldado1", etc.
5 //Luego de crear los 2 ejércitos se deben mostrar los datos de todos los soldados
6 //de ambos ejércitos e indicar qué ejército fue el ganador.
7 //Restricción: aplicar arreglos DE OBJETOS y métodos para inicializar los ejércitos,
8 //mostrar ejército y mostrar ejército ganador. La métrica a aplicar para indicar
9 //el ganador es el mayor número de soldados de cada ejército, puede haber empates.
10 //APLICAR ARREGLO DE OBJETOS.
11 package lab3_diegocervantes;
12
13 import java.util.Random;
14 public class Ejercicio03 {
15
16     public static void main(String[] args) {
17         //Creación de los ejércitos
18         SoldadoGuerra[] ejercito1 = inicializarEjercito(1);
19         SoldadoGuerra[] ejercito2 = inicializarEjercito(2);
20
21         //Mostrar los datos de ambos ejércitos
22         System.out.println("Ejército 1:");
23         mostrarEjercito(ejercito1);
24         System.out.println("\nEjército 2:");
25         mostrarEjercito(ejercito2);
26
27         //Determinar y mostrar el ejército ganador
28         mostrarGanador(ejercito1, ejercito2);
29     }
```



```
30
31 //Método para inicializar un ejército con un número aleatorio de soldados
32 public static SoldadoGuerra[] inicializarEjercito(int numeroEjercito) {
33     Random rand = new Random();
34     int cantidadSoldados = rand.nextInt(5) + 1; // Número de soldados entre 1 y 5
35     SoldadoGuerra[] ejercito = new SoldadoGuerra[cantidadSoldados];
36
37     //Crear cada soldado con nombres "Soldado0", "Soldado1", etc.
38     for (int i = 0; i < ejercito.length; i++) {
39         ejercito[i] = new SoldadoGuerra("Soldado" + i + "_E" + numeroEjercito);
40     }
41     return ejercito;
42 }
43
44 //Método para mostrar los datos de un ejército
45 public static void mostrarEjercito(SoldadoGuerra[] ejercito) {
46     for (int i = 0; i < ejercito.length; i++) {
47         System.out.println("Nombre del soldado " + (i + 1) + ": " + ejercito[i].getNombre());
48     }
49     System.out.println("Total de soldados: " + ejercito.length);
50 }
51
52 //Método para determinar y mostrar el ejército ganador
53 public static void mostrarGanador(SoldadoGuerra[] ejercito1, SoldadoGuerra[] ejercito2) {
54     System.out.println("\nResultados:");
55     System.out.println("Ejército 1 tiene " + ejercito1.length + " soldados.");
56     System.out.println("Ejército 2 tiene " + ejercito2.length + " soldados.");
57
58     //Comparar el número de soldados y determinar el ganador
59     if (ejercito1.length > ejercito2.length) {
60         System.out.println("Ejército 1 es el ganador!");
61     } else if (ejercito2.length > ejercito1.length) {
62         System.out.println("Ejército 2 es el ganador!");
63     } else {
64         System.out.println("Es un empate!");
65     }
66 }
67 }
```

	<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>

```

1 //Autor: Diego Aristides Cervantes Apaza
2 //Problema: Escribir un programa donde se creen 2 ejércitos, cada uno con un número
3 //aleatorio de soldados entre 1 y 5, considerando sólo su nombre. Sus datos se
4 //inicializan automáticamente con nombres tales como "Soldado0", "Soldado1", etc.
5 //Luego de crear los 2 ejércitos se deben mostrar los datos de todos los soldados
6 //de ambos ejércitos e indicar qué ejército fue el ganador.
7 //Restricción: aplicar arreglos DE OBJETOS y métodos para inicializar los ejércitos,
8 //mostrar ejército y mostrar ejército ganador. La métrica a aplicar para indicar
9 //el ganador es el mayor número de soldados de cada ejército, puede haber empates.
10 //APLICAR ARREGLO DE OBJETOS.
11 package lab3_diegocervantes;
12
13
14 public class SoldadoGuerra {
15     private String nombre;
16
17     // Constructor que inicializa el nombre del soldado
18     public SoldadoGuerra(String nombre) {
19         this.nombre = nombre;
20     }
21
22     // Método para obtener el nombre del soldado
23     public String getNombre() {
24         return nombre;
25     }
26 }

```

II. PRUEBAS

Ejercicio 1:

Analice, complete y pruebe el Código de la clase DemoBatalla

Output ×

Lab3_DiegoCervantes (run) ×

Lab3_DiegoCervantes (run) ×

```

run:
Nave 1
Nombre: a
Fila: 3
Columna: 2
Estado (true/false): true
Puntos: 5
Nave 2
Nombre: delmon
Fila: 9
Columna: 1
Estado (true/false): false
Puntos: 4
Nave 3
Nombre: damian
Fila: 4
Columna: 1
Estado (true/false): true
Puntos: 2
Nave 4
Nombre: omar
Fila: 7
Columna: 11
Estado (true/false): false
Puntos: 4
Nave 5
Nombre: ortiz
Fila: 8
Columna: 2
Estado (true/false): false
Puntos: 10
Nave 6
Nombre: nabuc
Fila: 6
Columna: 1
Estado (true/false): true
Puntos: 10
Nave 7
Nombre: uner
Fila: 6
Columna: 3
Estado (true/false): false
Puntos: 23

```

Output ×

Lab3_DiegoCervantes (run) ×

Lab3_DiegoCervantes (run) #2 ×

```

Puntos: 23
Nave 8
Nombre: onert
Fila: 8
Columna: 6
Estado (true/false): false
Puntos: 10
Nave 9
Nombre: ifer
Fila: 9
Columna: 3
Estado (true/false): false
Puntos: 20
Nave 10
Nombre: javier
Fila: 6
Columna: 1
Estado (true/false): false
Puntos: 30

Naves creadas:
a | Fila: 3 | Columna: 2 | Estado: true | Puntos: 5
delmon | Fila: 9 | Columna: 1 | Estado: false | Puntos: 4
damian | Fila: 4 | Columna: 1 | Estado: true | Puntos: 2
omar | Fila: 7 | Columna: 11 | Estado: false | Puntos: 4
ortiz | Fila: 8 | Columna: 2 | Estado: false | Puntos: 10
nabuc | Fila: 6 | Columna: 1 | Estado: true | Puntos: 10
uner | Fila: 6 | Columna: 3 | Estado: false | Puntos: 23
onert | Fila: 8 | Columna: 6 | Estado: false | Puntos: 10
ifer | Fila: 9 | Columna: 3 | Estado: false | Puntos: 20



Ingrese nombre para buscar: ifer                                s: 30
ifer | Fila: 9 | Columna: 3 | Estado: false | Puntos: 20

Ingrese puntos máximos: 23                                      20
a | Fila: 3 | Columna: 2 | Estado: true | Puntos: 5
delmon | Fila: 9 | Columna: 1 | Estado: false | Puntos: 4
damian | Fila: 4 | Columna: 1 | Estado: true | Puntos: 2
omar | Fila: 7 | Columna: 11 | Estado: false | Puntos: 4
ortiz | Fila: 8 | Columna: 2 | Estado: false | Puntos: 10
nabuc | Fila: 6 | Columna: 1 | Estado: true | Puntos: 10
uner | Fila: 6 | Columna: 3 | Estado: false | Puntos: 23
onert | Fila: 8 | Columna: 6 | Estado: false | Puntos: 10
ifer | Fila: 9 | Columna: 3 | Estado: false | Puntos: 20

Nave con más puntos:
javier - Puntos: 30

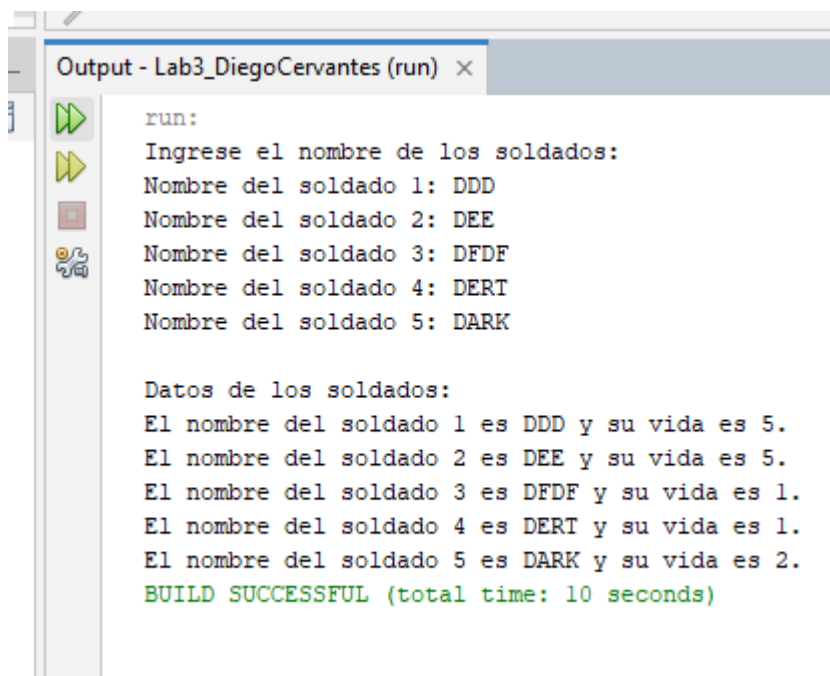
Naves desordenadas:
ifer | Fila: 9 | Columna: 3 | Estado: false | Puntos: 20
omar | Fila: 7 | Columna: 11 | Estado: false | Puntos: 4
javier | Fila: 6 | Columna: 1 | Estado: false | Puntos: 30
damian | Fila: 4 | Columna: 1 | Estado: true | Puntos: 2
onert | Fila: 8 | Columna: 6 | Estado: false | Puntos: 10
uner | Fila: 6 | Columna: 3 | Estado: false | Puntos: 23
nabuc | Fila: 6 | Columna: 1 | Estado: true | Puntos: 10
delmon | Fila: 9 | Columna: 1 | Estado: false | Puntos: 4
ortiz | Fila: 8 | Columna: 2 | Estado: false | Puntos: 10
a | Fila: 3 | Columna: 2 | Estado: true | Puntos: 5
BUILD SUCCESSFUL (total time: 45 minutes 20 seconds)

```

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

Ejercicio 2:

Escribir un programa donde se creen 5 soldados considerando su nombre y nivel de vida. Ingresar sus datos y después mostrarlos. Restricción: aplicar arreglos con objetos



```

run:
Ingrese el nombre de los soldados:
Nombre del soldado 1: DDD
Nombre del soldado 2: DEE
Nombre del soldado 3: DFDF
Nombre del soldado 4: DERT
Nombre del soldado 5: DARK

Datos de los soldados:
El nombre del soldado 1 es DDD y su vida es 5.
El nombre del soldado 2 es DEE y su vida es 5.
El nombre del soldado 3 es DFDF y su vida es 1.
El nombre del soldado 4 es DERT y su vida es 1.
El nombre del soldado 5 es DARK y su vida es 2.
BUILD SUCCESSFUL (total time: 10 seconds)

```

Ejercicio 3:

Problema: Escribir un programa donde se creen 2 ejércitos, cada uno con un número aleatorio de soldados entre 1 y 5, considerando sólo su nombre. Sus datos se inicializan automáticamente con nombres tales como "Soldado0", "Soldado1", etc. Luego de crear los 2 ejércitos se deben mostrar los datos de todos los soldados de ambos ejércitos e indicar qué ejército fue el ganador.

Restricción: aplicar arreglos DE OBJETOS y métodos para inicializar los ejércitos, mostrar ejército y mostrar ejército ganador. La métrica a aplicar para indicar el ganador es el mayor número de soldados de cada ejército, puede haber empates. APLICAR ARREGLO DE OBJETOS.

```

run:
Ejército 1:
Nombre del soldado 1: Soldado0_E1
Nombre del soldado 2: Soldado1_E1
Total de soldados: 2

Ejército 2:
Nombre del soldado 1: Soldado0_E2
Nombre del soldado 2: Soldado1_E2
Nombre del soldado 3: Soldado2_E2
Nombre del soldado 4: Soldado3_E2
Total de soldados: 4

Resultados:
Ejército 1 tiene 2 soldados.
Ejército 2 tiene 4 soldados.
¡Ejército 2 es el ganador!
BUILD SUCCESSFUL (total time: 0 seconds)

```

III. CUESTIONARIO:



CONCLUSIONES

Se ha utilizado los requerimientos el uso de arreglo de objetos que tienen lugar a cabo en la resolución del problema mismo.

METODOLOGÍA DE TRABAJO

Se ha utilizado prueba y error para la creación final y acabada del proyecto en los “commits” de la aplicación GITHUB donde se evidencia tal avance, del día de hoy. Todo ello trabajado en APACHE-NetBeans.

REFERENCIAS Y BIBLIOGRAFÍA

	<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: 14</p>