
	<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

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	<i>Fundamentos de la Programación 02</i>				
TÍTULO DE LA PRÁCTICA:	<i>Arreglos de objetos</i>				
NÚMERO DE PRÁCTICA:	<i>03</i>	AÑO LECTIVO:	<i>2024-B</i>	NRO. SEMESTRE:	<i>//</i>
FECHA DE PRESENTACIÓN	<i>05/10/2024</i>	HORA DE PRESENTACIÓN	<i>16:58:45</i>		
INTEGRANTE (s) <i>Mauro Snayder Sullca Mamani</i>				NOTA (0-20)	
DOCENTE(s): <i>Ing. Lino Jose Pinto Oppe</i>					

RESULTADOS Y PRUEBAS	
<p>I. EJERCICIOS RESUELTOS:</p> <p>1. Analice, complete y pruebe el Código de la clase DemoBatalla</p> <pre> 5 package Laboratorio_03; 6 7 /** 8 * 9 * @author mauro 10 */ 11 12 public class Nave { 13 // Creamos los atributos 14 private String nombre; 15 private int fila; 16 private String columna; 17 private boolean estado; 18 private int puntos; 19 20 // Creamos los Set y Get de cada atributo 21 public String getNombre() { 22 return nombre; 23 } 24 25 public void setNombre(String nombre) { 26 this.nombre = nombre; 27 } 28 29 public int getFila() { 30 return fila; 31 } 32 33 public void setFila(int fila) { 34 this.fila = fila; 35 } 36 37 public String getColumna() { 38 return columna; 39 } </pre>	

```



40
41 public void setColumna(String columna) {
42     this.columna = columna;
43 }
44
45 public boolean isEstado() {
46     return estado;
47 }
48
49 public void setEstado(boolean estado) {
50     this.estado = estado;
51 }
52
53 public int getPuntos() {
54     return puntos;
55 }
56
57 public void setPuntos(int puntos) {
58     this.puntos = puntos;
59 }
60
61 // Creamos el toString y un metodo que imprimirá el nombre y puntos
62 public String toString() {
63     return "Nave(" + "nombre=" + nombre + ", fila=" + fila + ", columna=" + columna + ", estado=" + estado + ", puntos=" + puntos + ')';
64 }
65
66 public String navePunto() {
67     return "Nave(" + "nombre=" + nombre + ", puntos=" + puntos + ')';
68 }
69
70

```

```

5 package Laboratorio_03;
6
7 /**
8  *
9  * @author mauro
10  */
11 import java.util.*;
12
13 public class DemoBatalla {
14     public static void main (String[] args){
15         Nave[] misNaves=new Nave[10]; //creamos un arreglo para 10 objetos tipo nave
16         Scanner scan=new Scanner(System.in);
17         String nomb,col;
18         int fil,punt;
19         boolean est;
20         for (int i=0;i<misNaves.length;i++){ // Ingresamos los datos para cada nave
21             System.out.println("Nave " + (i+1));
22             System.out.print("Nombre: ");
23             nomb = scan.next();
24             System.out.print("Fila: ");
25             fil = scan.nextInt();
26             System.out.print("Columna: ");
27             col = scan.next();
28             System.out.print("Estado (true,false): ");
29             est = scan.nextBoolean();
30             System.out.print("Puntos: ");
31             punt = scan.nextInt();
32
33             misNaves[i] = new Nave(); //Se crea un objeto Nave y se asigna su referencia a misNaves
34
35             misNaves[i].setNombre(nomb);
36             misNaves[i].setFila(fil);
37             misNaves[i].setColumna(col);
38             misNaves[i].setEstado(est);
39             misNaves[i].setPuntos(punt);
40         }
41         System.out.println("\nNaves creadas:");
42         mostrarNaves(misNaves);
43         mostrarPorNombre(misNaves,scan);
44         mostrarPorPuntos(misNaves,scan);
45         System.out.println("\nNave con mayor número de puntos: " + mostrarMayorPuntos(misNaves));
46     }
47     //Método que muestra todas las naves
48     public static void mostrarNaves(Nave [] flota){
49         for (int i=0;i<flota.length;i++){
50             System.out.println(flota[i].toString());
51         }
52     }

```

	<p style="text-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: 3

```

53
54 //Método que muestra todas las naves de un nombre que se pide por teclado
55 public static void mostrarPorNombre(Nave [] flota,Scanner scan){
56     System.out.println("Ingrese el nombre de su nave: ");
57     String nombre=scan.next();
58     for (int i=0;i<flota.length;i++){
59         if (flota[i].getNombre().equals(nombre)) // Si los nombres son iguales, se imprime los datos de la nave
60             System.out.println(flota[i].toString());
61     }
62 }
63
64 //Método que muestra todas las naves con un número de puntos inferior o igual
65 //al número de puntos que se pide por teclado
66 public static void mostrarPorPuntos(Nave [] flota,Scanner scan){
67     System.out.println("Ingrese un numero de puntos: ");
68     int numero=scan.nextInt();
69     for (int i=0;i<flota.length;i++){
70         if (flota[i].getPuntos()<=numero) //Si la nave tiene menos puntos que "numero", entoces se imprime
71             System.out.println(flota[i].navePunto());
72     }
73 }
74
75 //Método que devuelve la Nave con mayor número de Puntos
76 public static Nave mostrarMayorPuntos(Nave [] flota){
77     Nave mayor=flota[0]; //Definimos temporalmente a la nave mayor
78     for (int i=1;i<flota.length-1;i++){
79         if (flota[i].getPuntos()>mayor.getPuntos()){ //Si existen una nave con mayor puntos
80             mayor=flota[i]; //entonces se actualiza la variable mayor
81         }
82     }
83     return mayor;
84 }
85 //Metodo que devuelve un arreglo de objetos con todos los objetos previamente ingresados
86 //pero aleatoriamente desordenados
87 public static Nave[] aleatoriamenteDesordenados(Nave[] flota){
88     Nave[] misNavesDes=new Nave[flota.length]; //Creamos un nuevo arreglo
89     System.arraycopy(flota,0,misNavesDes,0,flota.length); //Copiamos los valores del arreglo original
90     Nave cambio; // Creamos una variable que nos ayudara a poder hacer cambios dentro del arreglo
91     for (int i=0;i<flota.length;i++){
92         cambio=misNavesDes[i]; //"cambio" tendra los valores de una parte del arreglo
93         int aleatorio=(int) (Math.random()*flota.length); // Generamos una posicion random
94         misNavesDes[i]=misNavesDes[aleatorio]; // La posicion "i" tendra un intercambio de posicion con
95         misNavesDes[aleatorio]=cambio; // la posicion "aleatorio"
96     }
97     return misNavesDes;
98 }
99 }

```

2. Solucionar la Actividad 4 de la Práctica 1 pero usando arreglo de objetos

```

5 package Laboratorio_03;
6
7 /**
8  *
9  * @author Mauro Snyder
10  */
11 public class Soldado {
12     // Creamos los atributos
13     private String nombre;
14     private int vida;
15
16     // Creamos los Set y Get de cada atributo
17     public String getNombre() {
18         return nombre;
19     }
20
21     public void setNombre(String nombre) {
22         this.nombre = nombre;
23     }
24
25     public int getVida() {
26         return vida;
27     }
28
29     public void setVida(int vida) {
30         this.vida = vida;
31     }
32     // Creamos el toString
33     public String toString() {
34         return nombre+"\t->"+vida;
35     }
36 }

```

```

5 package Laboratorio_03;
6
7 /**
8  *
9  * @author Usuario24B
10 */
11 import java.util.*;
12
13 public class Actividad04 {
14     public static void main(String[] args) {
15         Scanner scan=new Scanner(System.in);
16         Soldado[] soldados=new Soldado[5]; //Creamos un arreglo para almacenar 5 objetos de tipo Soldado
17         for(int i=0;i<soldados.length;i++){ //Ingresamos los datos de los 5 soldados
18             Soldado soldado1=new Soldado(); //Creamos "soldado1" que no ayudara a poder ingresar los datos
19                                     //de los soldados en el arreglo
20             System.out.println("Ingrese el nombre del soldado "+(i+1)+" : ");
21             soldado1.setNombre(scan.next());
22             System.out.println("Ingrese la vida de su soldado: ");
23             soldado1.setVida(scan.nextInt());
24             soldados[i]=soldado1; //Ponemos los datos del soldado en el arreglo
25         }
26         System.out.println("NOMBRE \t VIDA");
27         for(int k=0;k<soldados.length;k++) //Imprimimos los datos de los 5 soldados
28             System.out.println(soldados[k].toString());
29     }
30 }



```

3. Solucionar la Actividad 5 de la Práctica 1 pero usando arreglos de objetos

```

5 package Laboratorio_03;
6
7 /**
8  *
9  * @author Usuario24B
10 */
11
12 public class Actividad05 {
13     public static void main(String[] args) {
14         // Inicializamos dos ejércitos con un número aleatorio de soldados entre 1 y 5
15         Soldado[] ejercito1=inicializarEjercito((int) (Math.random()*5+1));
16         Soldado[] ejercito2=inicializarEjercito((int) (Math.random()*5+1));
17         // Mostramos los dos ejércitos
18         mostrarEjercito(ejercito1,1);
19         mostrarEjercito(ejercito2,2);
20         // Mostramos el ganador en función de la cantidad de soldados
21         mostrarGanador(ejercito1,ejercito2);
22     }
23     // Método para inicializar un ejército con un número determinado de soldados
24     public static Soldado[] inicializarEjercito(int num){
25         // Creamos un arreglo para almacenar los soldados
26         Soldado[] ejercito=new Soldado[num];
27         // Llenamos el arreglo con soldados nuevos
28         for(int i=0;i<ejercito.length;i++){
29             Soldado soldado1=new Soldado();
30             soldado1.setNombre("soldado"+i);// Asignamos un nombre a cada soldado
31             ejercito[i]=soldado1; // Almacenamos el soldado en el arreglo
32         }
33         return ejercito;
34     }
35 }

```

	<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: 5

```

35 // Asignamos un nombre a cada soldado
36 public static void mostrarEjercito(Soldado[] ejercito,int tipo){
37     System.out.println("\nEjercito "+tipo+": ");
38     // Asignamos un nombre a cada soldado
39     for(int i=0;i<ejercito.length;i++)
40         System.out.println(ejercito[i].getNombre());
41 }
42 // Asignamos un nombre a cada soldado
43 public static void mostrarGanador(Soldado[] ejercito1,Soldado[] ejercito2){
44     // Asignamos un nombre a cada soldado
45     if(ejercito1.length>ejercito2.length)
46         System.out.println("\nGana el ejercito 1 con "+ejercito1.length+" soldados.");
47     else if(ejercito1.length<ejercito2.length)
48         System.out.println("\nGana el ejercito 2 con "+ejercito2.length+" soldados.");
49     else
50         System.out.println("\nEmpatan los ejercitos con "+ejercito1.length+" soldados.");
51 }
52 }

```

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?

Se utilizaron diferentes conjuntos de objetos dentro de un arreglo, como soldado o nave con atributos como nombre, vida, etc. Estos valores de prueba fueron elegidos para representar distintos casos que podrían aparecer en el contexto de la práctica.

¿Qué resultado esperabas obtener para cada valor de entrada?

Se esperaban resultados específicos según las operaciones realizadas: al buscar un objeto por un atributo (como el nombre), o también por la puntuación, como también poder desordenar las posiciones de un arreglo, se anticipaba que el resultado fuera correcto según los valores de los objetos.

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

Los resultados obtenidos se compararon con las expectativas y confirmaron la efectividad de las operaciones: al buscar un objeto, se obtuvo el objeto esperado; al realizar cálculos, los resultados coincidieron con lo anticipado; y al filtrar objetos, la lista resultante contenía únicamente aquellos que cumplían con las condiciones deseadas.

En esta ejecución estamos viendo el ingreso de 5 naves y los datos que nos imprime después de ingresar los datos, como por ejemplo mostrar los datos de todas las naves ingresadas, buscar una nave en específico, mostrar las naves con puntuación menor a 15 y la nave con mayor puntuación.

Output - Laboratorios (run)

```

run:
Nave 1
Nombre: nave1
Fila: 2
Columna: 4
Estado (true,false): true
Puntos: 40
Nave 2
Nombre: nave2
Fila: 5
Columna: 6
Estado (true,false): false
Puntos: 20
Nave 3
Nombre: nave3
Fila: 5
Columna: 5
Estado (true,false): true
Puntos: 24
Nave 4
Nombre: nave4
Fila: 4
Columna: 4
Estado (true,false): false
Puntos: 15
Nave 5
Nombre: nave5
Fila: 8
Columna: 5
Estado (true,false): true
Puntos: 22
Nave 6
Nombre: nave6
Fila: 3
Columna: 2
Estado (true,false): true
Puntos: 28
Nave 7
Nombre: nave7
Fila: 5
Columna: 2
Estado (true,false): false
Puntos: 31

```

Output - Laboratorios (run)

```

run:
Nave 1
Nombre: nave1
Fila: 2
Columna: 3
Estado (true,false): true
Puntos: 12
Nave 2
Nombre: nave2
Fila: 4
Columna: 5
Estado (true,false): true
Puntos: 15
Nave 3
Nombre: nave3
Fila: 6
Columna: 9
Estado (true,false): false
Puntos: 19
Nave 4
Nombre: nave4
Fila: 2
Columna: 5
Estado (true,false): true
Puntos: 11
Nave 5
Nombre: nave5
Fila: 6
Columna: 7
Estado (true,false): false
Puntos: 20

Naves creadas:
Nave(nombre=nave1, fila=2, columna=3, estado=true, puntos=12)
Nave(nombre=nave2, fila=4, columna=5, estado=true, puntos=15)
Nave(nombre=nave3, fila=6, columna=9, estado=false, puntos=19)
Nave(nombre=nave4, fila=2, columna=5, estado=true, puntos=11)
Nave(nombre=nave5, fila=6, columna=7, estado=false, puntos=20)
Ingrese el nombre de su nave:
nave4
Nave(nombre=nave4, fila=2, columna=5, estado=true, puntos=11)
Ingrese un numero de puntos:
15
Nave(nombre=nave1, puntos=12)
Nave(nombre=nave2, puntos=15)
Nave(nombre=nave4, puntos=11)

Nave con mayor numero de puntos: Nave(nombre=nave5, fila=6, columna=7, estado=false, puntos=20)
BUILD SUCCESSFUL (total time: 1 minute 3 seconds)

```



En la siguiente ejecución estamos viendo el ingreso de datos de 5 soldados y como lo imprime esos datos ingresados.

Output - Laboratorios (run)

```

run:
Ingrese el nombre del soldado 1:
pablo
Ingrese la vida de su soldado:
3
Ingrese el nombre del soldado 2:
Tyrone
Ingrese la vida de su soldado:
2
Ingrese el nombre del soldado 3:
austin
Ingrese la vida de su soldado:
7
Ingrese el nombre del soldado 4:
Tasha
Ingrese la vida de su soldado:
4
Ingrese el nombre del soldado 5:
Unicua
Ingrese la vida de su soldado:
1
NOMBRE  VIDA
pablo   ->3
Tyrone  ->2
austin  ->7
Tasha   ->4
Unicua  ->1
BUILD SUCCESSFUL (total time: 48 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: 7

En la ejecución del último ejercicio estamos viendo de como se generan 2 ejércitos con números de soldados aleatorios y como se determina quien gana la batalla.

```
Output - Laboratorios (run)

run:

Ejercito 1:
soldado0
soldado1
soldado2

Ejercito 2:
soldado0
soldado1
soldado2
soldado3

Gana el ejercito 2 con 4 soldados.
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
Output - Laboratorios (run)

run:

Ejercito 1:
soldado0
soldado1

Ejercito 2:
soldado0
soldado1

Empatan los ejércitos con 2 soldados.
BUILD SUCCESSFUL (total time: 0 seconds)
```

III. COMMITS:

Entramos a nuestra carpeta donde están nuestros archivos y añadimos los cambios.

```
Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   Laboratorio_03/Actividad04.java
    modified:   Laboratorio_03/Actividad05.java
    modified:   Laboratorio_03/DemoBatalla.java
    modified:   Laboratorio_03/Nave.java
    modified:   Laboratorio_03/Soldado.java

no changes added to commit (use "git add" and/or "git commit -a")

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git add .

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   Laboratorio_03/Actividad04.java
    modified:   Laboratorio_03/Actividad05.java
    modified:   Laboratorio_03/DemoBatalla.java
    modified:   Laboratorio_03/Nave.java
    modified:   Laboratorio_03/Soldado.java
```

Hacemos un commit.

```
Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git commit -m "terminado"
[master c0b05e3] terminado
5 files changed, 50 insertions(+), 31 deletions(-)
```

Subimos nuestro commit a nuestro repositorio remoto.



```
Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git push -u origin master
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 20 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 1.84 KiB | 1.84 MiB/s, done.
Total 8 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/MauroSullcaMamani/FDLP2_LAB.git
 1d8b4fb..c0b05e3 master -> master
branch 'master' set up to track 'origin/master'.
```

Link de mi repositorio: https://github.com/MauroSullcaMamani/FDLP2_LAB.git

RUBRICA:

Contenido y demostración		Puntos	Checklis t	Estudiant e	Profeso r
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	✓	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	✓	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	✓	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	✓	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	✓	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	✓	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	✓	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	✓	3	
TOTAL		20		18	

Tabla 2: Rúbrica para contenido del Informe y demostración

	<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>

CONCLUSIONES

El uso de arreglos de objetos permite organizar datos relacionados de manera estructurada, facilitando su gestión y acceso, lo que mejora la eficiencia en el manejo de colecciones de datos. Proporcionan métodos claros para manipular datos a través de sus propiedades y métodos, aumentando la legibilidad y mantenimiento del código.

METODOLOGÍA DE TRABAJO

Lo primero que hice, es leer cada los enunciados de cada ejercicio y también tomar en cuenta las restricciones que nos da para así poder buscar una solución al problema. Después observar el código y entender la funcionalidad de cada uno y completar las partes que están incompletas. Y por último comprobar nuestro código ingresando varias veces valores de prueba para ver que nuestro código está funcionando correctamente.

REFERENCIAS Y BIBLIOGRAFÍA

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.