


	<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: 1</p>

INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programación 02				
TÍTULO DE LA PRÁCTICA:	Arreglos de Objetos				
NÚMERO DE PRÁCTICA:	03	AÑO LECTIVO:	2024	NRO. SEMESTRE:	2
FECHA DE PRESENTACIÓN	04/10/2024	HORA DE PRESENTACIÓN	22/20/00		
INTEGRANTE (s) German Arturo Chipana Jerónimo				NOTA (0-20)	
DOCENTE(s): Pinto Oppe Lino José					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <p><i>El estudiante coloca la evidencia de los ejercicios propuestos realizados en la sesión de laboratorio, en el tiempo o duración indicado por el docente.</i></p> <p><i>El docente debe colocar la retroalimentación por cada ejercicio que el estudiante/grupo ha presentado</i></p> <p>EJERCICIO 01:</p> <p>DEMO BATALLA</p> <p>Analizar y completar el código de la clase Demo Batalla.</p> <p><u>MAIN:</u></p>

```
1  /*
2     Autor: Chipana Jeronimo German Arturo
3     Proposito: Ejercicio 01
4  */
5  package ejercicio01lab03;
6
7  import java.util.*;
8  public class Ejercicio01Lab03 {
9
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         for (int i = 0; i < misNaves.length; i++) {
17             System.out.println("Nave " + (i+1));
18             System.out.print("Nombre: ");
19             nomb = sc.next();
20             System.out.println("Fila ");
21             fil = sc.nextInt();
22             System.out.print("Columna: ");
23             col = sc.next();
24             System.out.print("Estado: ");
25             est = sc.nextBoolean();
26             System.out.print("Puntos: ");
27             punt = sc.nextInt();
28             misNaves[i] = new Nave(); //Se crea un objeto Nave y se asigna su referencia a misNaves
29             misNaves[i].setNombre(nomb);
30             misNaves[i].setFila(fil);
31             misNaves[i].setColumna(col);
32             misNaves[i].setEstado(est);
33             misNaves[i].setPuntos(punt);
34         }
35         System.out.println("\nNaves creadas:");
36         mostrarNaves(misNaves);
37         mostrarPorNombre(misNaves);
38         mostrarPorPuntos(misNaves);
39         System.out.println("\nNave con mayor número de puntos: " + mostrarMayorPuntos(misNaves));
40         Nave [] navesDesordenadas=desordenarNaves(misNaves);
41         System.out.println("Naves desordenadas: ");
42         for(Nave nave : navesDesordenadas){
43             System.out.println(nave.toString());
44         }
45     }
46     //Método para mostrar todas las naves
47     public static void mostrarNaves(Nave [] flota){
48         for(Nave nave : flota){
49             System.out.println(nave.toString());
50         }
51     }
```

	<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>

```



46 //Método para mostrar todas las naves
47 public static void mostrarNaves(Nave [] flota){
48     for(Nave nave : flota){
49         System.out.println(nave.toString());
50     }
51 }
52 //Método para mostrar todas las naves de un nombre que se pide por teclado
53 public static void mostrarPorNombre(Nave [] flota){
54     Scanner scan=new Scanner(System.in);
55     System.out.println("Ingrese nombre de la nave a buscar: ");
56     String naveBuscar=scan.next();
57     System.out.println("Naves encontradas con el nombre "+naveBuscar+" : ");
58     for(Nave nave : flota){
59         if(nave.getNombre().equals(naveBuscar)){
60             System.out.println(nave.toString());
61         }
62     }
63 }
64 //Método para mostrar 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){
67     Scanner scan=new Scanner(System.in);
68     System.out.println("Ingrese el numero max de puntos: ");
69     int maxPuntos=scan.nextInt();
70     System.out.println("Naves con puntos inferiores a "+maxPuntos);
71     for(Nave nave : flota){
72         if(nave.getPuntos()<=maxPuntos){
73             System.out.println(nave.toString());
74         }
75     }
76 }
77 //Método que devuelve la Nave con mayor número de Puntos
78 public static Nave mostrarMayorPuntos(Nave [] flota){
79     Nave naveMaxPuntos=flota[0];
80     for(Nave nave : flota){
81         if(nave.getPuntos()>naveMaxPuntos.getPuntos()){
82             naveMaxPuntos=nave;
83         }
84     }
85     return naveMaxPuntos;
86 }
87 //Crear un método que devuelva un nuevo arreglo de objetos con todos los objetos previamente ingresados
88 //pero aleatoriamente desordenados
89 public static Nave[] desordenarNaves(Nave[] flota) {
90     for (int i=flota.length-1;i>0;i--) {
91         // Generar un número aleatorio entre 0 y i
92         int j=(int)(Math.random()*(i+1));
93         // Intercambiar flota[i] con flota[j]
94         Nave temp=flota[i];
95         flota[i]=flota[j];
96         flota[j]=temp;
97     }
98     return flota; // Devolver el arreglo desordenado
99 }
100 }

```

CLASE NAVE:

```
8 public class Nave {
9     private String nombre;
10    private int fila;
11    private String columna;
12    private boolean estado;
13    private int puntos;
14    // Metodos mutadores
15    public void setNombre( String n){
16        nombre = n;
17    }
18    public void setFila(int f){
19        fila = f;
20    }
21    public void setColumna(String c){
22        columna = c;
23    }
24    public void setEstado(boolean e){
25        estado = e;
26    }
27    public void setPuntos(int p){
28        puntos = p;
29    }
30    // Metodos accesorios
31    public String getNombre(){
32        return nombre;
33    }
34    public int getFila(){
35        return fila;
36    }
37    public String getColumna(){
38        return columna;
39    }
40    public boolean getEstado(){
41        return estado;
42    }
43    public int getPuntos(){
44        return puntos;
45    }
46    // Completar con otros métodos necesarios
47    public String toString(){
48        return "Nave{" +
49            "nombre=" + nombre +
50            ", fila=" + fila +
51            ", columna=" + columna +
52            ", estado=" + estado +
53            ", puntos=" + puntos +
54            '}';
55    }
56 }
57
```

EJERCICIO 4 LABORATORIO 01

	<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: 5</p>

MAIN:

```

1  /*
2     Autor: Chipana Jeronimo German Arturo
3     Proposito: Ejercicio 04
4  */
5  package ejercicio4lab1;
6
7  import java.util.*;
8  public class Ejercicio4Lab1 {
9
10     public static void main(String[] args) {
11         Scanner scan=new Scanner(System.in);
12         // Crea un arreglo de objetos Soldados con capacidad para 5 soldados
13         Soldados[] soldado=new Soldados[5];
14         // Bucle para inicializar cada soldado
15         for(int i=0;i<soldado.length;i++){
16             // Crea un nuevo objeto Soldados para cada posición del arreglo
17             soldado[i]=new Soldados();
18             // Solicita al usuario que ingrese el nombre del soldado
19             System.out.println("Ingrese nombre soldado "+(i+1)+": ");
20             String nombre=scan.next();
21             // Solicita al usuario que ingrese la vida del soldado
22             System.out.println("Ingrese vida de soldado "+(i+1)+": ");
23             int vida=scan.nextInt();
24             // Asigna el nombre y la vida al objeto Soldados
25             soldado[i].setNombre(nombre);
26             soldado[i].setVida(vida);
27         }
28         // Muestra los nombres y vidas de los soldados ingresados
29         System.out.println("Nombres de soldados: ");
30         for(int j=0;j<soldado.length;j++){
31             System.out.println("Nombre soldado "+(j+1)+": "+soldado[j].getNombre()+
32                 "\tVida soldado "+(j+1)+": "+soldado[j].getVida());
33         }
34     }
35 }

```

CLASE SOLDADOS:

```

1  /*
2     Clase Soldados
3  */
4  package ejercicio4lab1;
5
6
7  public class Soldados {
8      private String nombre;
9      private int vida;
10     // Metodos mutadores
11     public void setVida(int vida) {
12         this.vida = vida;
13     }
14
15     public void setNombre(String nombre) {
16         this.nombre = nombre;
17     }
18     // Metodos accesoros
19     public String getNombre() {
20         return nombre;
21     }
22
23     public int getVida() {
24         return vida;
25     }
26 }
27

```



EJERCICIO 5 LABORATORIO 01

MAIN:

```
1  /*
2   * Autor: Chipana Jeronimo German Arturo
3   * Proposito: Ejercicio 05
4   */
5  package ejercicio5lab1;
6
7  public class Ejercicio5Lab1 {
8
9      public static void main(String[] args) {
10         // Inicializa dos arreglos de Ejercito con una cantidad aleatoria de soldados (de 1 a 5)
11         Ejercito[] ejercito1=new Ejercito[(int) (Math.random()*5+1)];
12         Ejercito[] ejercito2=new Ejercito[(int) (Math.random()*5+1)];
13         // Llama al método para inicializar los soldados en ambos ejércitos
14         inicializarEjercitos(ejercito1);
15         inicializarEjercitos(ejercito2);
16         // Muestra la cantidad de soldados en el primer ejército
17         System.out.println("EJERCITO 1 : "+(ejercito1.length));
18         // Muestra los nombres de los soldados del primer ejército
19         mostrarEjercito(ejercito1);
20         // Muestra la cantidad de soldados en el segundo ejército
21         System.out.println("EJERCITO 2 : "+(ejercito2.length));
22         // Muestra los nombres de los soldados del segundo ejército
23         mostrarEjercito(ejercito2);
24         // Determina y muestra el ganador entre los dos ejércitos
25         mostrarEjercitoGanador(ejercito1,ejercito2);
26     }
27     // Método que inicializa un ejército con soldados
28     public static void inicializarEjercitos(Ejercito[] ejercito){
29         for(int i=0;i<ejercito.length;i++){
30             ejercito[i]=new Ejercito();
31             String nombre="Soldado"+(i+1);
32             ejercito[i].setSoldado(nombre);
33         }
34     }
35     // Método que muestra los nombres de los soldados de un ejército
36     public static void mostrarEjercito(Ejercito[] ejercito){
37         for(int i=0;i<ejercito.length;i++){
38             System.out.println(ejercito[i].getSoldado());
39         }
40     }
41     // Método que determina y muestra el ganador entre dos ejércitos
42     public static void mostrarEjercitoGanador(Ejercito[] ejercito1,Ejercito[] ejercito2){
43         if(ejercito1.length==ejercito2.length){
44             if(ejercito1.length==ejercito2.length)
45                 System.out.println("Hubo un EMPATE de ejércitos, ambos con "+ejercito1.length+" soldados...");
46             else
47                 System.out.println("Gano el EJERCITO 1 con "+ejercito1.length+" soldados!");
48         }
49         else
50             System.out.println("Gano el EJERCITO 2 con "+ejercito2.length+" soldados!");
51     }
52 }
53
```

CLASE EJERCITO:

```
1  /*
2   * Clase Ejercito
3   */
4  package ejercicio5lab1;
5
6  public class Ejercito {
7      private String soldado;
8      // Metodo accesor
9      public String getSoldado() {
10         return soldado;
11     }
12     // Metodo mutador
13     public void setSoldado(String soldado) {
14         this.soldado = soldado;
15     }
16 }
17
```

	<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>

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?

En el caso del ejercicio del demo batalla ingresando el nombre, fila, columna, estado, puntos de cada nave; ósea ingresando String, int, boolean.

En el caso de la actividad 4 ingresando los nombres y la vida de cada soldado, siendo estas de tipo String y int respectivamente.

En el caso de la actividad 5 solo imprime la cantidad de soldados y que ejercito ganó.

¿Qué resultado esperabas obtener para cada valor de entrada?

En el caso del demo batalla esperaba que los datos que yo ingresara para cada nave, se almacenaran en el arreglo de la clase Nave y que cuando yo llame a los métodos me devolvieran la respuesta correcta según le pida.

En el caso de la actividad 4 esperaba que el nombre y la vida de cada soldado se guarde en el arreglo de objetos para luego imprimirlos.

En el caso de la actividad 5 esperaba que lo que se imprimiera fuera el resultado correcto.

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

En el demo batalla de que se almacenaran los datos de mis naves, con sus respectivos datos correspondientes.

En la actividad 4 se pudo imprimir los nombres y vidas respectivos de cada soldado de forma tabular.

En la actividad 5 se pudo imprimir el resultado correcto de cada batalla de ejércitos debido a la cantidad de soldados de cada uno.

DEMO BATALLA

1. Primero se ingresaron los datos de las 10 naves creadas.

```

Output - Ejercicio01Lab03 (run) x
run:
Nave 1
Nombre: Galactus
Fila
2
Columna: 4
Estado: true
Puntos: 23
Nave 2
Nombre: Roma
Fila
5
Columna: 3
Estado: true
Puntos: 56
Nave 3
Nombre: Italia
Fila
2
Columna: 5
Estado: true
Puntos: 12
Nave 4
Nombre: Lactea
Fila
7
Columna: 5
Estado: true
Puntos: 65
Nave 5
Nombre: Lipzig
Fila
6
Columna: 3
Estado: true
Puntos: 52
Nave 6
Nombre: Bayer
Fila
8
Columna: 3
Estado: true
Puntos: 25



```

2. Se mostraron las naves con sus datos respectivos.

```

Naves creadas:
Nave{nombre= Galactus, fila=2, columna=4, estado=true, puntos=23}
Nave{nombre= Roma, fila=5, columna=3, estado=true, puntos=56}
Nave{nombre= Italia, fila=2, columna=5, estado=true, puntos=12}
Nave{nombre= Lactea, fila=7, columna=5, estado=true, puntos=65}
Nave{nombre= Lipzig, fila=6, columna=3, estado=true, puntos=52}
Nave{nombre= Bayer, fila=8, columna=3, estado=true, puntos=25}
Nave{nombre= Torola, fila=6, columna=6, estado=true, puntos=46}
Nave{nombre= Rosas, fila=7, columna=4, estado=true, puntos=76}
Nave{nombre= Cali, fila=5, columna=2, estado=true, puntos=67}
Nave{nombre= Trulex, fila=2, columna=5, estado=true, puntos=45}

```


	<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>

3. Se pidió el nombre de una nave y se imprimieron sus datos.

```
Ingrese nombre de la nave a buscar:
Torola
Naves encontradas con el nombre Torola :
Nave{nombre= Torola, fila=6, columna=6, estado=true, puntos=46}
```

4. Se ingresó el máximo número de puntos para mostrar las naves con puntos iguales o inferiores.

```
Ingrese el numero max de puntos:
67
Naves con puntos inferiores a 67
Nave{nombre= Galactus, fila=2, columna=4, estado=true, puntos=23}
Nave{nombre= Roma, fila=5, columna=3, estado=true, puntos=56}
Nave{nombre= Italia, fila=2, columna=5, estado=true, puntos=12}
Nave{nombre= Lactea, fila=7, columna=5, estado=true, puntos=65}
Nave{nombre= Lipzig, fila=6, columna=3, estado=true, puntos=52}
Nave{nombre= Bayer, fila=8, columna=3, estado=true, puntos=25}
Nave{nombre= Torola, fila=6, columna=6, estado=true, puntos=46}
Nave{nombre= Cali, fila=5, columna=2, estado=true, puntos=67}
Nave{nombre= Trulex, fila=2, columna=5, estado=true, puntos=45}
```

5. Se mostró los datos de la nave con mayor puntaje.

```
Nave con mayor número de puntos: Nave{nombre= Rosas, fila=7, columna=4, estado=true, puntos=76}
Naves desordenadas:
```

6. Se imprimió las naves desordenadas.

```
Naves desordenadas:
Nave{nombre= Rosas, fila=7, columna=4, estado=true, puntos=76}
Nave{nombre= Lipzig, fila=6, columna=3, estado=true, puntos=52}
Nave{nombre= Roma, fila=5, columna=3, estado=true, puntos=56}
Nave{nombre= Cali, fila=5, columna=2, estado=true, puntos=67}
Nave{nombre= Lactea, fila=7, columna=5, estado=true, puntos=65}
Nave{nombre= Torola, fila=6, columna=6, estado=true, puntos=46}
Nave{nombre= Galactus, fila=2, columna=4, estado=true, puntos=23}
Nave{nombre= Trulex, fila=2, columna=5, estado=true, puntos=45}
Nave{nombre= Italia, fila=2, columna=5, estado=true, puntos=12}
Nave{nombre= Bayer, fila=8, columna=3, estado=true, puntos=25}
BUILD SUCCESSFUL (total time: 2 minutes 56 seconds)
```

ACTIVIDAD 4 LABORATORIO 01

1. Primero se pide al usuario ingresar el nombre y la vida de cada soldado.

```

Output - Ejercicio4Lab1 (run) x
run:
Ingrese nombre soldado 1:
German
Ingrese vida de soldado 1:
6
Ingrese nombre soldado 2:
Arturo
Ingrese vida de soldado 2:
2
Ingrese nombre soldado 3:
Santiago
Ingrese vida de soldado 3:
5
Ingrese nombre soldado 4:
Joaquin
Ingrese vida de soldado 4:
4
Ingrese nombre soldado 5:
Enrique
Ingrese vida de soldado 5:
5

```

2. Se imprime el nombre y vida respectivo de cada soldado de forma tabular.

```

Nombres de soldados:
Nombre soldado 1: German      Vida soldado 1: 6
Nombre soldado 2: Arturo      Vida soldado 2: 2
Nombre soldado 3: Santiago    Vida soldado 3: 5
Nombre soldado 4: Joaquin     Vida soldado 4: 4
Nombre soldado 5: Enrique     Vida soldado 5: 5
BUILD SUCCESSFUL (total time: 33 seconds)
|

```

ACTIVIDAD 5 LABORATORIO 01

1. En esta prueba ambos ejércitos empatan con la misma cantidad de soldados.

```

Output - Ejercicio5Lab1 (run) x
run:
EJERCITO 1 : 5
Soldado0
Soldado1
Soldado2
Soldado3
Soldado4
EJERCITO 2 : 5
Soldado0
Soldado1
Soldado2
Soldado3
Soldado4
Hubo un EMPATE de ejércitos, ambos con 5 soldados...
BUILD SUCCESSFUL (total time: 0 seconds)

```

2.En esta prueba el ejercito 1 gana por la mayor cantidad de soldados.

```

Output - Ejercicio5Lab1 (run) x
run:
EJERCITO 1 : 4
Soldado0
Soldado1
Soldado2
Soldado3
EJERCITO 2 : 1
Soldado0
Gano el EJERCITO 1 con 4 soldados!
BUILD SUCCESSFUL (total time: 0 seconds)

```

MIS COMMITS:

PRIMERA VERSION:

```

user@DESKTOP-CS7PIHP MINGW64 ~/Desktop/CHIPANA_JERONIMO_LABORATORIO_03 (main)
$ git add .

user@DESKTOP-CS7PIHP MINGW64 ~/Desktop/CHIPANA_JERONIMO_LABORATORIO_03 (main)
$ git commit -m "Primera version oficial"
[main 0954239] Primera version oficial
2 files changed, 119 insertions(+)
create mode 100644 Ejercicio01Lab03.java
create mode 100644 Nave.java

user@DESKTOP-CS7PIHP MINGW64 ~/Desktop/CHIPANA_JERONIMO_LABORATORIO_03 (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.43 KiB | 1.43 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/ChipanaGerman/CHIPANA_JERONIMO_LABORATORIO_03.git
a366ec2..0954239 main -> main

user@DESKTOP-CS7PIHP MINGW64 ~/Desktop/CHIPANA_JERONIMO_LABORATORIO_03 (main)
$

```

20:54
2/10/2024

Commit Message	Commit Hash	Time Ago
ChipanaGerman Primera version oficial	0954239	2 minutes ago
Ejercicio01Lab03.java	Primera version oficial	2 minutes ago
Nave.java	Primera version oficial	2 minutes ago


SEGUNDA VERSION:

```
user@DESKTOP-CS7PIHP MINGW64 ~/Desktop/CHIPANA_JERONIMO_LABORATORIO_03 (main)
$ git add .

user@DESKTOP-CS7PIHP MINGW64 ~/Desktop/CHIPANA_JERONIMO_LABORATORIO_03 (main)
$ git commit -m "Segunda version oficial(mitad)"
[main 6a3a596] Segunda version oficial(mitad)
 2 files changed, 38 insertions(+), 6 deletions(-)

user@DESKTOP-CS7PIHP MINGW64 ~/Desktop/CHIPANA_JERONIMO_LABORATORIO_03 (main)
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 839 bytes | 839.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/ChipanaGerman/CHIPANA_JERONIMO_LABORATORIO_03.git
 0954239..6a3a596  main -> main
```

15:47
3/10/2024



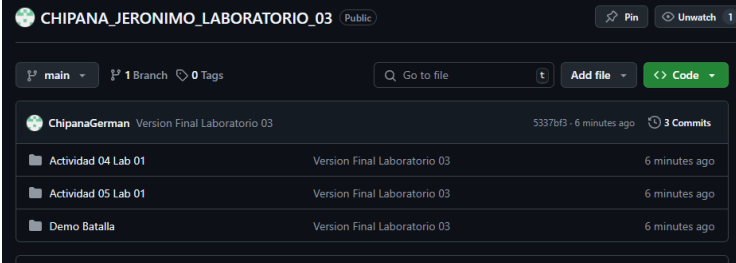
TERCERA VERSION:

```
user@DESKTOP-CS7PIHP MINGW64 ~/Desktop/CHIPANA_JERONIMO_LABORATORIO_03 (main)
$ git add .

user@DESKTOP-CS7PIHP MINGW64 ~/Desktop/CHIPANA_JERONIMO_LABORATORIO_03 (main)
$ git commit -m "Version Final Laboratorio 03"
[main 5337bf3] Version Final Laboratorio 03
 6 files changed, 165 insertions(+), 19 deletions(-)
 create mode 100644 Actividad 04 Lab 01/Ejercicio4Lab1.java
 create mode 100644 Actividad 04 Lab 01/Soldados.java
 create mode 100644 Actividad 05 Lab 01/Ejercicio5Lab1.java
 create mode 100644 Actividad 05 Lab 01/Ejercito.java
 rename Ejercicio01Lab03.java => Demo Batalla/Ejercicio01Lab03.java (81%)
 rename Nave.java => Demo Batalla/Nave.java (86%)

user@DESKTOP-CS7PIHP MINGW64 ~/Desktop/CHIPANA_JERONIMO_LABORATORIO_03 (main)
$ git push origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (11/11), 3.73 KiB | 1.86 MiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/ChipanaGerman/CHIPANA_JERONIMO_LABORATORIO_03.git
 66cdd9f..5337bf3  main -> main
```

15:08
4/10/2024



III. 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	X	1	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	3	
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	4	X	4	



	evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).				
TOTAL		20		18	

Colocar la evidencia de las respuestas realizadas al cuestionario enunciado en la guía práctica de laboratorio.

CONCLUSIONES	
<p><i>Colocar las conclusiones, apreciaciones reflexivas, opiniones finales a cerca de los resultados obtenidos de la sesión de laboratorio.</i></p> <p>CONCLUSIÓN:</p> <p>En esta sesión de laboratorio se comprendió y aplicó el manejo de arreglos de objetos en Java, logrando almacenar y manipular datos de manera eficiente. La correcta implementación de métodos y la solución de los ejercicios fortalecieron las habilidades en programación orientada a objetos, así como en la depuración y documentación del código.</p>	

METODOLOGÍA DE TRABAJO
<p><i>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.</i></p> <ol style="list-style-type: none"> 1. Se analizaron los ejercicios propuestos y se comprendió la lógica de los problemas. 2. Se implementó el código inicial utilizando arreglos de objetos para las clases solicitadas. 3. Se realizaron pruebas para verificar el funcionamiento correcto del programa, introduciendo diferentes valores de entrada. 4. Se corrigieron errores encontrados durante las pruebas. 5. Finalmente, se documentó el código y se realizaron los commits correspondientes en GitHub.

REFERENCIAS Y BIBLIOGRAFÍA
<p><i>Colocar las referencias utilizadas para el desarrollo de la práctica en formato IEEE</i></p> <p>M. Aedo López, Práctica de Laboratorio 3: Arreglos de Objetos, Universidad Nacional de San Agustín, 2023. https://github.com/ChipanaGerman/CHIPANA_JERONIMO_LABORATORIO_03</p>

	<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: 15</p>