

Informe de Laboratorio 03

Tema: Arreglos de Objetos

Nota	

${f Estudiante}$	Escuela	${f Asign atura}$
		Fundamentos de la
Christian Mestas Zegarra	Escuela Profesional de	Programación 2
cmestasz@unsa.edu.pe	Ingeniería de Sistemas	Semestre: II
		Código: 1701213

Laboratorio	${f Tema}$	Duración
03	Arreglos de Objetos	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 18 Setiembre 2023	Al 25 Setiembre 2023

1. Tarea

- Actividad 1: Analice, complete y pruebe el Código de la clase DemoBatalla.
- Actividad 2: Solucionar la Actividad 4 de la Práctica 1 pero usando arreglo de objetos.
- Actividad 3: Solucionar la Actividad 5 de la Práctica 1 pero usando arreglos de objetos.



2. Equipos, materiales y temas utilizados

- Sistema Operativo Microsoft Windows 10 Pro 64 bits
- Visual Studio Code 1.82.2
- Java Development Kit 17.0.1
- Git 2.41.0.windows.1
- Windows PowerShell 5.1.19041.3031
- Cuenta en GitHub con el correo institucional.
- Arreglos de objetos.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- https://github.com/cmestasz/fp2-23b.git
- URL para el laboratorio 01 en el Repositorio GitHub.
- https://github.com/cmestasz/fp2-23b/tree/main/fase01/lab01

4. Actividades con el repositorio GitHub

Creando plantillas

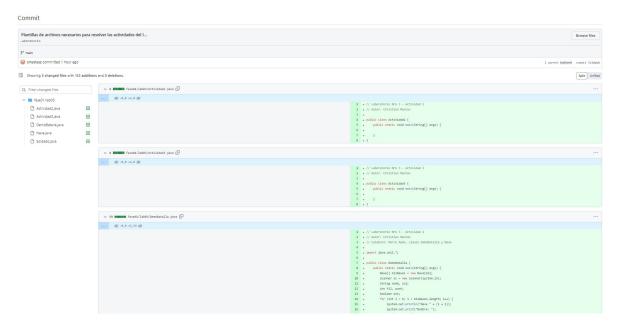
\$ mkdir lab03
\$ cd lab03
\$ code Nave.java
\$ code DemoBatalla.java
\$ code Soldado.java
\$ code Actividad2.java
\$ code Actividad3.java

Primer Commit / Plantillas

- \$ git add
- \$ git commit -m "Plantillas de archivos necesarios para resolver las actividades del laboratorio"
- \$ git push







Primer Commit.

Actualizando Nave.java y DemoBatalla.java

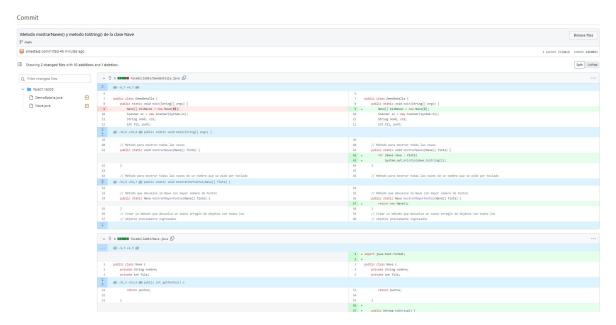
```
$ code Nave.java
$ code DemoBatalla.java
```

Segundo - Sexto Commit / Nave.java y DemoBatalla.java

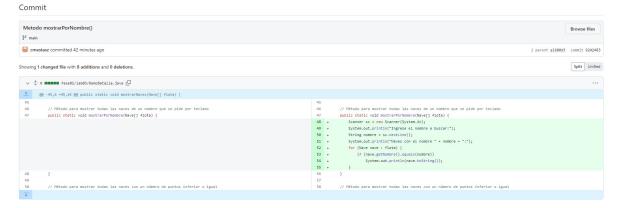
```
$ git add .
$ git commit -m "Metodo mostrarNaves() y metodo toString() de la clase Nave"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarPorNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarPorPuntos()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarMayorPuntos()"
$ code DemoBatalla.java
$ git commit -m "Metodo mostrarMayorPuntos()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo desordenar()"
$ git push
```







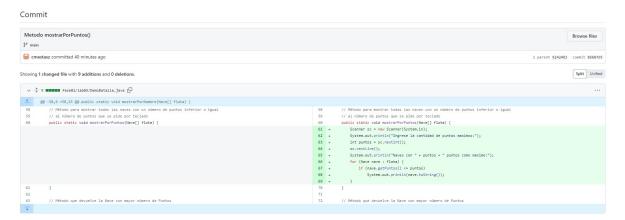
Segundo Commit.



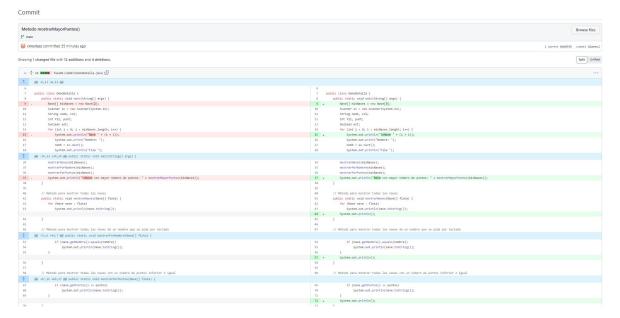
Tercer Commit.







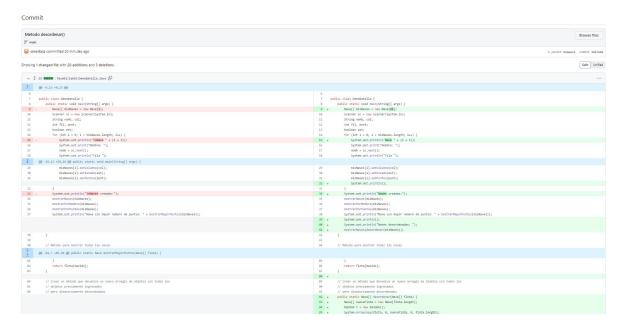
Cuarto Commit.



Quinto Commit.







Sexto Commit.

Actualizando Soldado.java

\$ code Soldado.java

Septimo Commit / Soldado.java

\$ git add Soldado.java
\$ git commit -m "Clase Soldado"
\$ git push





```
Clare Soldado

Prant

Character committed 17 minutes ago

1 parent 194766 consist ee00579

Showing I changed file with 27 additions and 1 deletion.

5 public clare Soldado

2 public clare Soldado

1 public clare Soldado

1 public clare Soldado

2 public clare Soldado

1 public clare Soldado

1 public clare Soldado

2 public clare Soldado

1 public clare Soldado

1
```

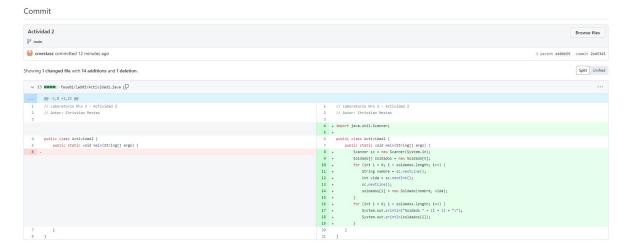
Septimo Commit.

Actualizando Actividad2.java

\$ code Actividad2.java

Octavo Commit / Soldado.java

```
$ git add Actividad2.java
$ git commit -m "Actividad 2"
$ git push
```



Octavo Commit.

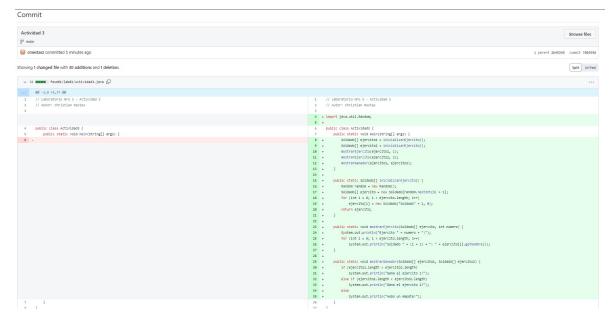


Actualizando Actividad3.java

\$ code Actividad3.java

Noveno Commit / Actividad3.java

```
$ git add Actividad3.java
$ git commit -m "Actividad 3"
$ git push
```



Noveno Commit.





5. Código desarrollado

Nave.java

```
public class Nave {
       private String nombre;
       private int fila;
       private String columna;
       private boolean estado;
       private int puntos;
       // Metodos mutadores
       public void setNombre(String n) {
10
           nombre = n;
12
       public void setFila(int f) {
           fila = f;
14
16
       public void setColumna(String c) {
17
18
           columna = c;
19
20
       public void setEstado(boolean e) {
21
           estado = e;
23
       public void setPuntos(int p) {
           puntos = p;
26
27
28
       // Metodos accesores
29
       public String getNombre() {
30
           return nombre;
32
33
       }
34
       public int getFila() {
35
36
           return fila;
39
       public String getColumna() {
40
           return columna;
41
42
       }
43
44
       public boolean getEstado() {
           return estado;
46
47
       }
48
       public int getPuntos() {
50
           return puntos;
```





- Clase que guarda nombre, fila, columna, estado y puntos de la nave.
- Posee tanto setters como getters para todos los atributos.
- Posee el metodo toString() para poder imprimir el objeto.



DemoBatalla.java

```
// Laboratorio Nro 3 - Actividad 1
   // Autor: Christian Mestas
   // Colaboro: Marco Aedo, clases DemoBatalla y Nave
   import java.util.*;
   public class DemoBatalla {
       public static void main(String[] args) {
           Nave[] misNaves = new Nave[10];
           Scanner sc = new Scanner(System.in);
           String nomb, col;
11
           int fil, punt;
           boolean est:
13
           for (int i = 0; i < misNaves.length; i++) {</pre>
              System.out.println("Nave " + (i + 1));
              System.out.print("Nombre: ");
              nomb = sc.next();
17
              System.out.println("Fila ");
18
              fil = sc.nextInt();
19
              System.out.print("Columna: ");
20
              col = sc.next();
21
              System.out.print("Estado: ");
              est = sc.nextBoolean();
              System.out.print("Puntos: ");
              punt = sc.nextInt();
              misNaves[i] = new Nave(); // Se crea un objeto Nave y se asigna su referencia a
                   misNaves
              misNaves[i].setNombre(nomb);
              misNaves[i].setFila(fil);
              misNaves[i].setColumna(col);
              misNaves[i].setEstado(est);
30
              misNaves[i].setPuntos(punt);
31
              System.out.println();
33
           System.out.println("Naves creadas:");
           mostrarNaves(misNaves);
           mostrarPorNombre(misNaves);
36
           mostrarPorPuntos(misNaves);
37
           System.out.println("Nave con mayor numero de puntos: " +
               mostrarMayorPuntos(misNaves));
           System.out.println();
           System.out.println("Naves desordenadas: ");
           mostrarNaves(desordenar(misNaves));
41
42
43
       // Metodo para mostrar todas las naves
44
       public static void mostrarNaves(Nave[] flota) {
45
           for (Nave nave : flota)
               System.out.println(nave.toString());
           System.out.println();
48
       }
49
50
       // Metodo para mostrar todas las naves de un nombre que se pide por teclado
       public static void mostrarPorNombre(Nave[] flota) {
           Scanner sc = new Scanner(System.in);
```



```
System.out.println("Ingrese el nombre a buscar:");
54
           String nombre = sc.nextLine();
55
           System.out.println("Naves con el nombre " + nombre + ":");
56
           for (Nave nave : flota) {
               if (nave.getNombre().equals(nombre))
58
                   System.out.println(nave.toString());
           System.out.println();
61
        }
63
        // Metodo para mostrar todas las naves con un numero de puntos inferior o igual
64
        // al numero de puntos que se pide por teclado
        public static void mostrarPorPuntos(Nave[] flota) {
           Scanner sc = new Scanner(System.in);
           System.out.println("Ingrese la cantidad de puntos maximos:");
           int puntos = sc.nextInt();
69
           sc.nextLine();
70
           System.out.println("Naves con " + puntos + " puntos como maximo:");
71
           for (Nave nave : flota) {
72
               if (nave.getPuntos() <= puntos)</pre>
73
                   System.out.println(nave.toString());
74
           System.out.println();
       }
        // Metodo que devuelve la Nave con mayor numero de Puntos
        public static Nave mostrarMayorPuntos(Nave[] flota) {
80
           int maxIdx = 0;
81
           for (int i = 0; i < flota.length; i++) {</pre>
82
               if (flota[i].getPuntos() > flota[maxIdx].getPuntos())
83
                   maxIdx = i;
84
85
           return flota[maxIdx];
        }
        // Crear un metodo que devuelva un nuevo arreglo de objetos con todos los
        // objetos previamente ingresados
        // pero aleatoriamente desordenados
        public static Nave[] desordenar(Nave[] flota) {
           Nave[] nuevaFlota = new Nave[flota.length];
           Random r = new Random();
           System.arraycopy(flota, 0, nuevaFlota, 0, flota.length);
95
           for (int idx = 0; idx < nuevaFlota.length; idx++) {</pre>
96
               int nIdx = r.nextInt(nuevaFlota.length);
97
               Nave t = nuevaFlota[idx];
98
               nuevaFlota[idx] = nuevaFlota[nIdx];
99
               nuevaFlota[nIdx] = t;
100
           return nuevaFlota;
       }
    7
104
```

- Se instancian 10 objetos de la clase nave en el arreglo misNaves, todos los datos son llenados por el usuario.
- mostrarNaves() imprime todas las naves usando su metodo toString().





- mostrarPorNombre() imprime todas las naves con el nombre que el usuario ingrese.
- mostrarPorPuntos() imprime todas las naves con menor o igual numero de puntos a lo que el usuario ingrese.
- mostrarMayorPuntos() retorna la nave que tiene la mayor cantidad de puntos, que luego es impresa.
- desordenar() retorna un nuevo arreglo de naves con las naves desordenadas al azar, que luego es impreso con el método mostrarNaves().



Soldado.java

```
public class Soldado {
       private String nombre;
       private int vida;
       public Soldado(String nombre, int vida) {
           setNombre(nombre);
           setVida(vida);
10
       public void setNombre(String nombre) {
           this.nombre = nombre;
11
       public void setVida(int vida) {
           this.vida = vida;
15
16
17
       public String getNombre() {
18
           return nombre;
19
20
21
22
       public int getVida() {
           return vida;
24
25
       public String toString() {
26
           return ("Nombre: " + nombre + ". Vida: " + vida);
27
28
29
```

- Clase que guarda nombre y vida del soldado.
- Posee tanto setters como getters para los atributos.
- Posee el metodo toString() para poder imprimir el objeto.



Actividad2.java

```
// Laboratorio Nro 3 - Actividad 2
   // Autor: Christian Mestas
   import java.util.Scanner;
   public class Actividad2 {
6
       public static void main(String[] args) {
           Scanner sc = new Scanner(System.in);
           Soldado[] soldados = new Soldado[5];
10
           for (int i = 0; i < soldados.length; i++) {</pre>
              String nombre = sc.nextLine();
11
              int vida = sc.nextInt();
              sc.nextLine();
               soldados[i] = new Soldado(nombre, vida);
           }
           for (int i = 0; i < soldados.length; i++) {</pre>
16
               System.out.println("Soldado " + (i + 1) + ":");
17
              System.out.println(soldados[i]);
18
19
       }
20
   }
```

- Los 5 nombres y sus respectivas vidas son leídas, para luego instanciar objetos de la clase soldado con sus respectivos atributos.
- Luego los atributos son impresos en el orden en el que se ingresaron.



Actividad3.java

```
// Laboratorio Nro 3 - Actividad 3
   // Autor: Christian Mestas
   import java.util.Random;
   public class Actividad3 {
6
       public static void main(String[] args) {
           Soldado[] ejercito1 = inicializarEjercito();
           Soldado[] ejercito2 = inicializarEjercito();
           mostrarEjercito(ejercito1, 1);
           mostrarEjercito(ejercito2, 2);
11
           mostrarGanador(ejercito1, ejercito2);
       }
       public static Soldado[] inicializarEjercito() {
           Random random = new Random();
           Soldado[] ejercito = new Soldado[random.nextInt(5) + 1];
17
           for (int i = 0; i < ejercito.length; i++)</pre>
18
               ejercito[i] = new Soldado("Soldado" + i, 0);
19
           return ejercito;
20
       }
21
22
       public static void mostrarEjercito(Soldado[] ejercito, int numero) {
           System.out.println("Ejercito " + numero + ":");
24
           for (int i = 0; i < ejercito.length; i++)</pre>
               System.out.println("Soldado " + (i + 1) + ": " + ejercito[i].getNombre());
       }
       public static void mostrarGanador(Soldado[] ejercito1, Soldado[] ejercito2) {
           if (ejercito1.length > ejercito2.length)
30
              System.out.println("Gana el ejercito 1!");
31
           else if (ejercito1.length < ejercito2.length)</pre>
32
              System.out.println("Gana el ejercito 2!");
33
34
           else
35
              System.out.println("Hubo un empate!");
36
       }
37
   }
```

- Se crean 2 ejércitos con el método inicializarEjercito(), que crea un arreglo de soldados de tamaño al azar entre 1 y 5.
- mostrar Ejercito() imprime una lista de los soldados de cada ejército.
- mostrarGanador() compara los tamaños de los ejércitos y declara al ganador como el ejército mas grande.



6. Ejecución del código

```
### STATES OF THE PROPERTY OF
```

DemoBatalla.java

```
Windows PowerShell

PS F:\Documents\UNSA\A1S2\FP2 lab\fp2-23b\fase01\lab03> java Actividad2

Jose Luis

56
Perez Manrique

46
Marco Marco

77
Mario Pepe

67
Empirino Josifastico

145
Soldado 1:
Nombre: Jose Luis. Vida: 56
Soldado 2:
Nombre: Perez Manrique. Vida: 46
Soldado 3:
Nombre: Marco Marco. Vida: 77
Soldado 4:
Nombre: Mario Pepe. Vida: 67
Soldado 5:
Nombre: Empirino Josifastico. Vida: 145
PS F:\Documents\UNSA\A1S2\FP2 lab\fp2-23b\fase01\lab03> _______
```

Actividad2.java





```
Windows PowerShell

PS F:\Documents\UNSA\A1S2\FP2 lab\fp2-23b\fase01\lab03> java Actividad3
Ejercito 1:
Soldado 1: Soldado0
Soldado 2: Soldado1
Soldado 3: Soldado2
Soldado 4: Soldado3
Soldado 5: Soldado4
Ejercito 2:
Soldado 1: Soldado0
Soldado 2: Soldado0
Soldado 2: Soldado1
Fjercito 2:
Soldado 3: Soldado0
Soldado 3: Soldado0
Soldado 3: Soldado0
Soldado 3: Soldado1
Soldado 3: Soldado0
Soldado 3: Soldado0
Soldado 3: Soldado1
Soldado 3: Soldado1
Soldado 3: Soldado1
Soldado 3: Soldado3
Soldado 3: Soldado1
Soldado 3: Soldado3
Soldado 4: Soldado3
Soldado 4: Soldado3
Soldado 5: So
```

Actividad3.java



7. Estructura de laboratorio 03

■ El contenido que se entrega en este laboratorio es el siguiente:

```
lab03/
|--- Nave.java
|--- DemoBatalla.java
|--- Soldado.java
|--- Actividad2.java
|--- Actividad3.java
|--- Informe.tex
|--- Informe.pdf
|--- img
   |--- logo_abet.png
   |--- logo_episunsa.png
   |--- logo_unsa.jpg
   |--- commit01.jpg
   |--- commit02.jpg
   |--- commit03.jpg
   |--- commit04.jpg
   |--- commit05.jpg
   |--- commit06.jpg
   |--- commit07.jpg
   |--- commit08.jpg
   |--- commit09.jpg
   |--- ejec01.jpg
   l--- ejec02.jpg
   |--- ejec03.jpg
```



8. Rúbricas

8.1. Entregable Informe

Tipo de Informe

Informe		
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.	

8.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobatoria, siempre y cuando cumpla con todos los items.
- El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

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



Rúbrica para contenido del Informe y demostración

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	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	1.5	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	1.5	
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 estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	1.5	
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			16.5	

9. Referencias

■ Aedo, M. y Castro, E. (2021). FUNDAMENTOS DE PROGRAMACIÓN 2 - Tópicos de Programación Orientada a Objetos. Editorial UNSA.