Informe de Laboratorio 03

Tema: Arreglos de Objetos

Nota

Estudiante	Escuela	Asignatura
Christian Mestas Zegarra cmestasz@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la Programación 2 Semestre: II Código: 1701213

Laboratorio	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 estándar.

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`

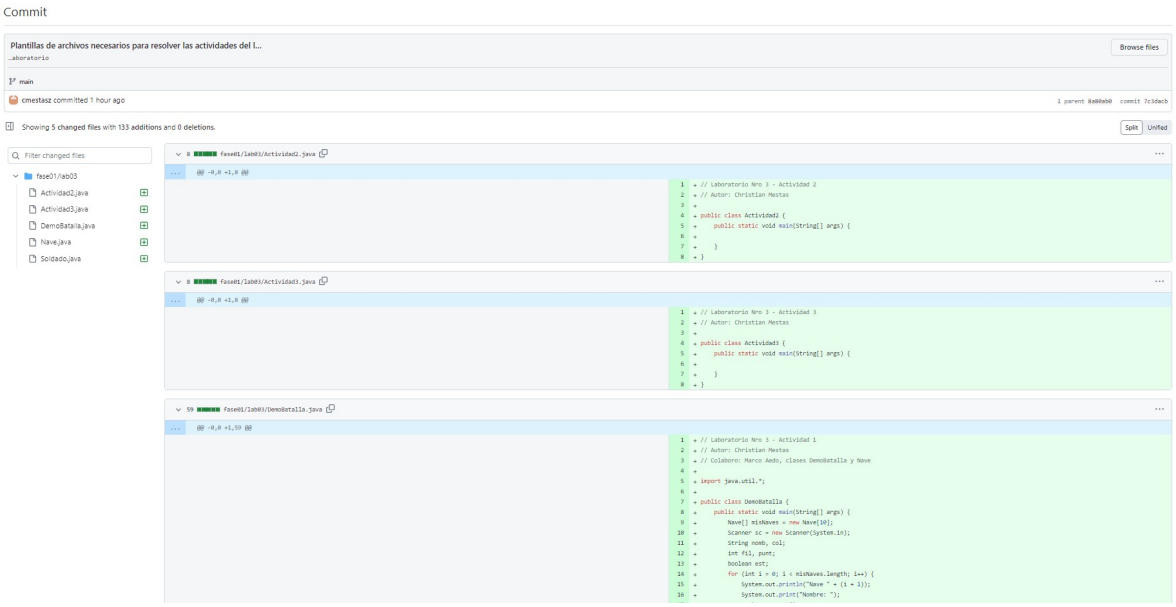
4. Actividades con el repositorio GitHub

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
```



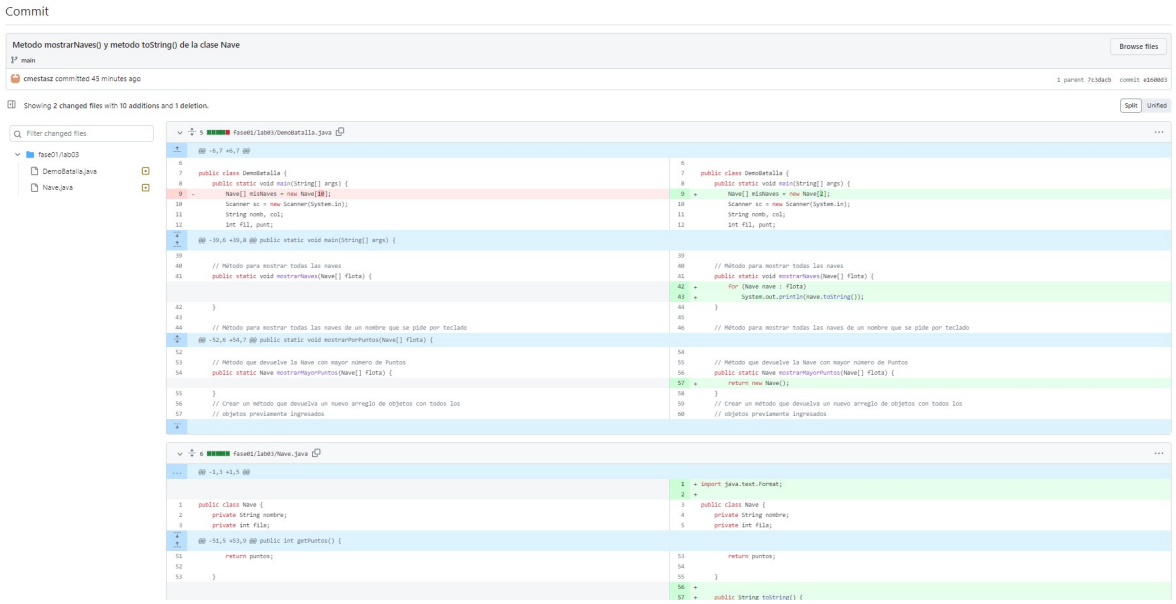
Primer Commit.

Actualizando Nave.java y DemoBatalla.java

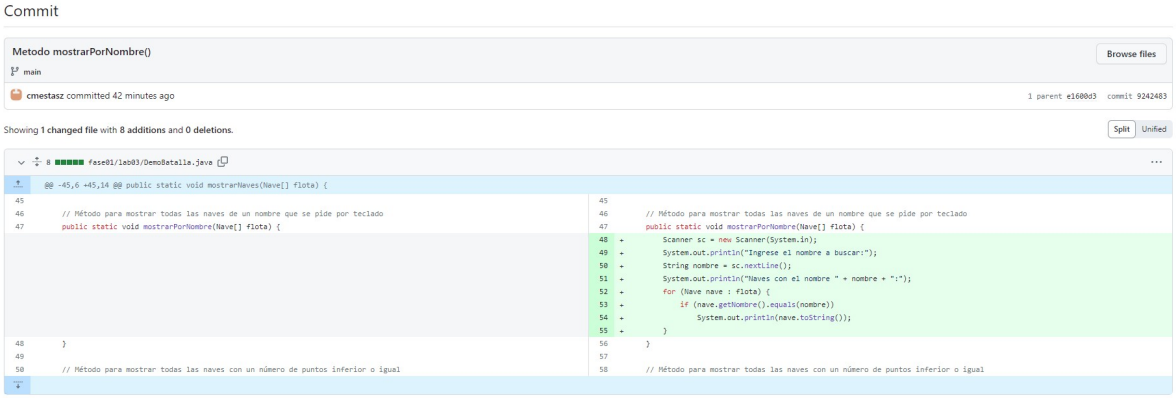
```
$ code Nave.java
$ code DemoBatalla.java
```

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

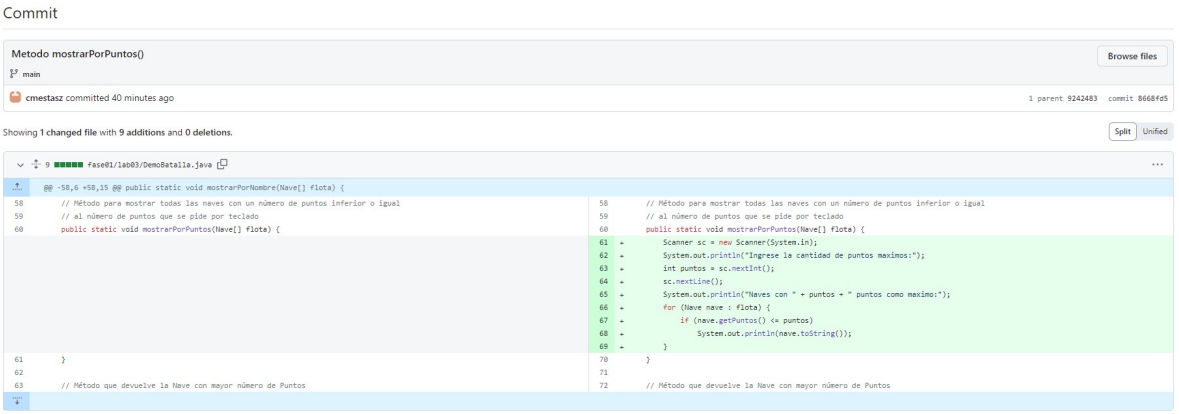
```
$ git add .
$ git commit -m "Metodo mostrarNaves() y metodo toString() de la clase Nave"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarPorNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarPorPuntos()"
$ code DemoBatalla.java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



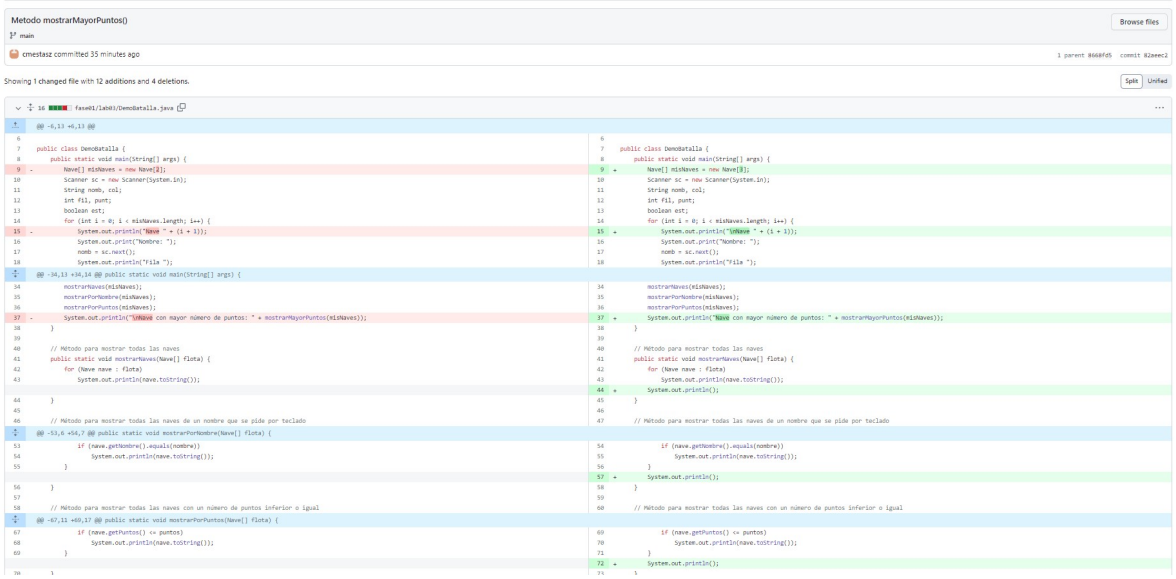
Segundo Commit.



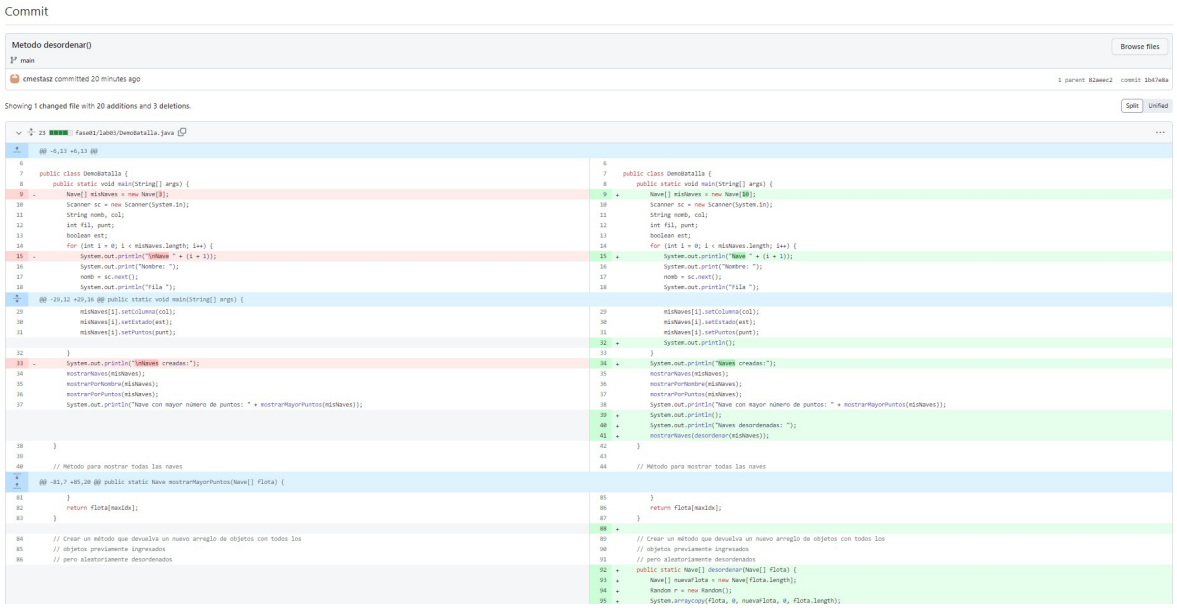
Tercer Commit.



Cuarto Commit.



Quinto Commit.



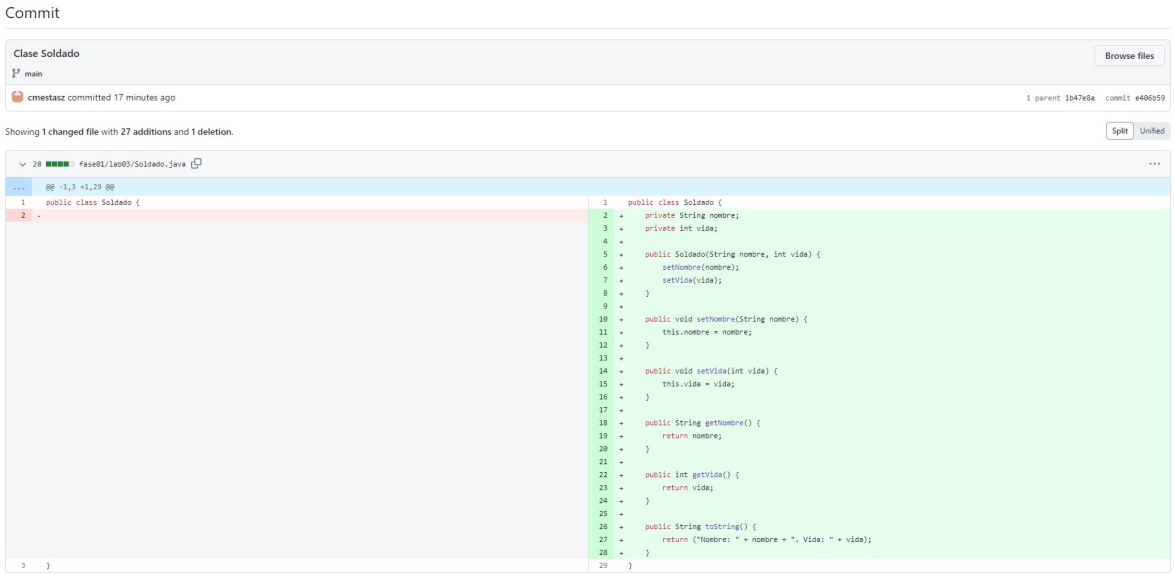
Sexto Commit.

Actualizando Soldado.java

```
$ code Soldado.java
```

Septimo Commit / Soldado.java

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



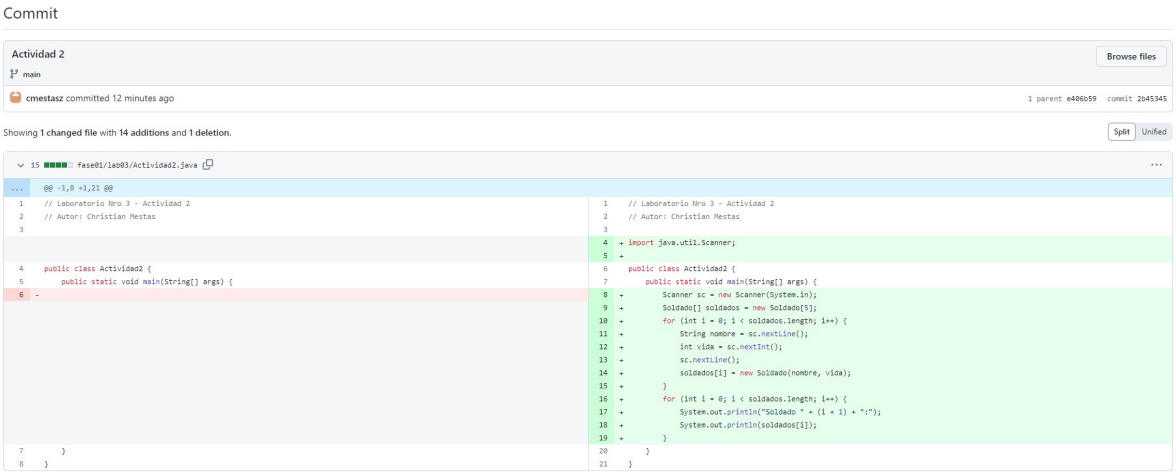
Septimo Commit.

Actualizando Actividad2.java

```
$ code Actividad2.java
```

Octavo Commit / Soldado.java

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



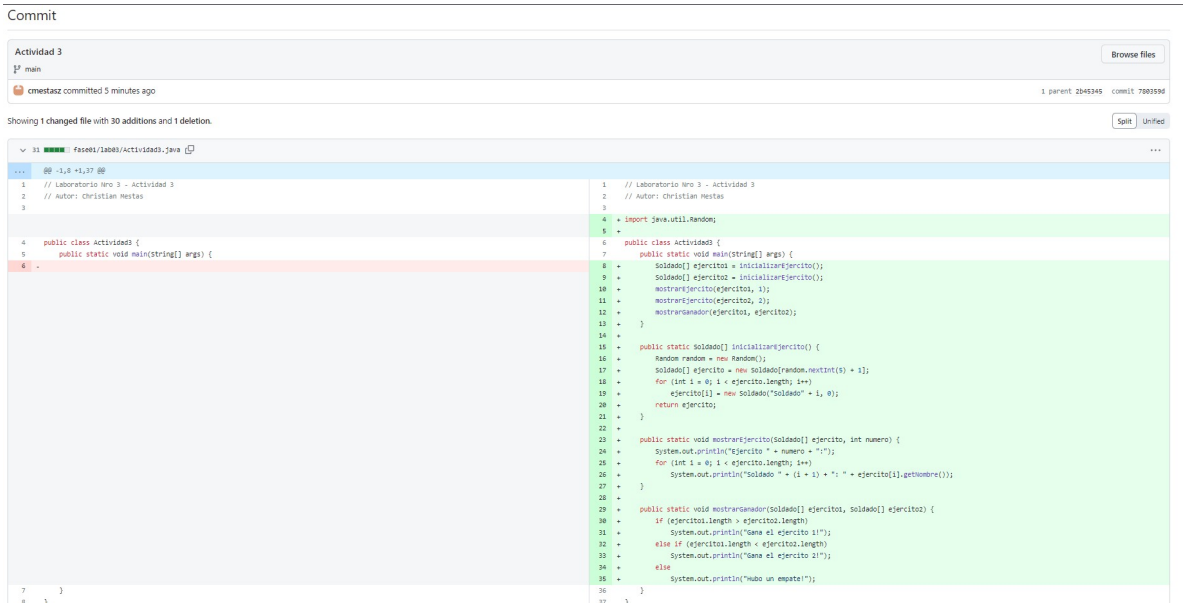
Octavo Commit.

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

```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) {
        nombre = n;
    }

    public void setFila(int f) {
        fila = f;
    }

    public void setColumna(String c) {
        columna = c;
    }

    public void setEstado(boolean e) {
        estado = e;
    }

    public void setPuntos(int p) {
        puntos = p;
    }

    // Metodos accesores
    public String getNombre() {
        return nombre;

    }

    public int getFila() {
        return fila;

    }

    public String getColumna() {
        return columna;

    }

    public boolean getEstado() {
        return estado;

    }

    public int getPuntos() {
        return puntos;

```

```
53      }
54
55      public String toString() {
56          return (nombre + ": (" + fila + ", " + columna + "). Estado: " + estado + ". Puntos:
                " + puntos);
57      }
58  }
```

- 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

```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;
        int fil, punt;
        boolean est;
        for (int i = 0; i < misNaves.length; i++) {
            System.out.println("Nave " + (i + 1));
            System.out.print("Nombre: ");
            nomb = sc.next();
            System.out.println("Fila ");
            fil = sc.nextInt();
            System.out.print("Columna: ");
            col = sc.next();
            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);
            misNaves[i].setPuntos(punt);
            System.out.println();
        }
        System.out.println("Naves creadas:");
        mostrarNaves(misNaves);
        mostrarPorNombre(misNaves);
        mostrarPorPuntos(misNaves);
        System.out.println("Nave con mayor numero de puntos: " +
            mostrarMayorPuntos(misNaves));
        System.out.println();
        System.out.println("Naves desordenadas: ");
        mostrarNaves(desordenar(misNaves));
    }

    // Metodo para mostrar todas las naves
    public static void mostrarNaves(Nave[] flota) {
        for (Nave nave : flota)
            System.out.println(nave.toString());
        System.out.println();
    }

    // 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);
```

```
54          System.out.println("Ingrese el nombre a buscar:");
55          String nombre = sc.nextLine();
56          System.out.println("Naves con el nombre " + nombre + ":");
57          for (Nave nave : flota) {
58              if (nave.getNombre().equals(nombre))
59                  System.out.println(nave.toString());
60          }
61          System.out.println();
62      }
63
64      // Metodo para mostrar todas las naves con un numero de puntos inferior o igual
65      // al numero de puntos que se pide por teclado
66      public static void mostrarPorPuntos(Nave[] flota) {
67          Scanner sc = new Scanner(System.in);
68          System.out.println("Ingrese la cantidad de puntos maximos:");
69          int puntos = sc.nextInt();
70          sc.nextLine();
71          System.out.println("Naves con " + puntos + " puntos como maximo:");
72          for (Nave nave : flota) {
73              if (nave.getPuntos() <= puntos)
74                  System.out.println(nave.toString());
75          }
76          System.out.println();
77      }
78
79      // Metodo que devuelve la Nave con mayor numero de Puntos
80      public static Nave mostrarMayorPuntos(Nave[] flota) {
81          int maxIdx = 0;
82          for (int i = 0; i < flota.length; i++) {
83              if (flota[i].getPuntos() > flota[maxIdx].getPuntos())
84                  maxIdx = i;
85          }
86          return flota[maxIdx];
87      }
88
89      // Crear un metodo que devuelva un nuevo arreglo de objetos con todos los
90      // objetos previamente ingresados
91      // pero aleatoriamente desordenados
92      public static Nave[] desordenar(Nave[] flota) {
93          Nave[] nuevaFlota = new Nave[flota.length];
94          Random r = new Random();
95          System.arraycopy(flota, 0, nuevaFlota, 0, flota.length);
96          for (int idx = 0; idx < nuevaFlota.length; idx++) {
97              int nIdx = r.nextInt(nuevaFlota.length);
98              Nave t = nuevaFlota[idx];
99              nuevaFlota[idx] = nuevaFlota[nIdx];
100             nuevaFlota[nIdx] = t;
101         }
102         return nuevaFlota;
103     }
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

```java
public class Soldado {
    private String nombre;
    private int vida;

    public Soldado(String nombre, int vida) {
        setNombre(nombre);
        setVida(vida);
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void setVida(int vida) {
        this.vida = vida;
    }

    public String getNombre() {
        return nombre;
    }

    public int getVida() {
        return vida;
    }

    public String toString() {
        return ("Nombre: " + nombre + ". Vida: " + vida);
    }
}
```

- 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

```java
// Laboratorio Nro 3 - Actividad 2
// Autor: Christian Mestas

import java.util.Scanner;

public class Actividad2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Soldado[] soldados = new Soldado[5];
        for (int i = 0; i < soldados.length; i++) {
            String nombre = sc.nextLine();
            int vida = sc.nextInt();
            sc.nextLine();
            soldados[i] = new Soldado(nombre, vida);
        }
        for (int i = 0; i < soldados.length; i++) {
            System.out.println("Soldado " + (i + 1) + ":");
            System.out.println(soldados[i]);
        }
    }
}
```

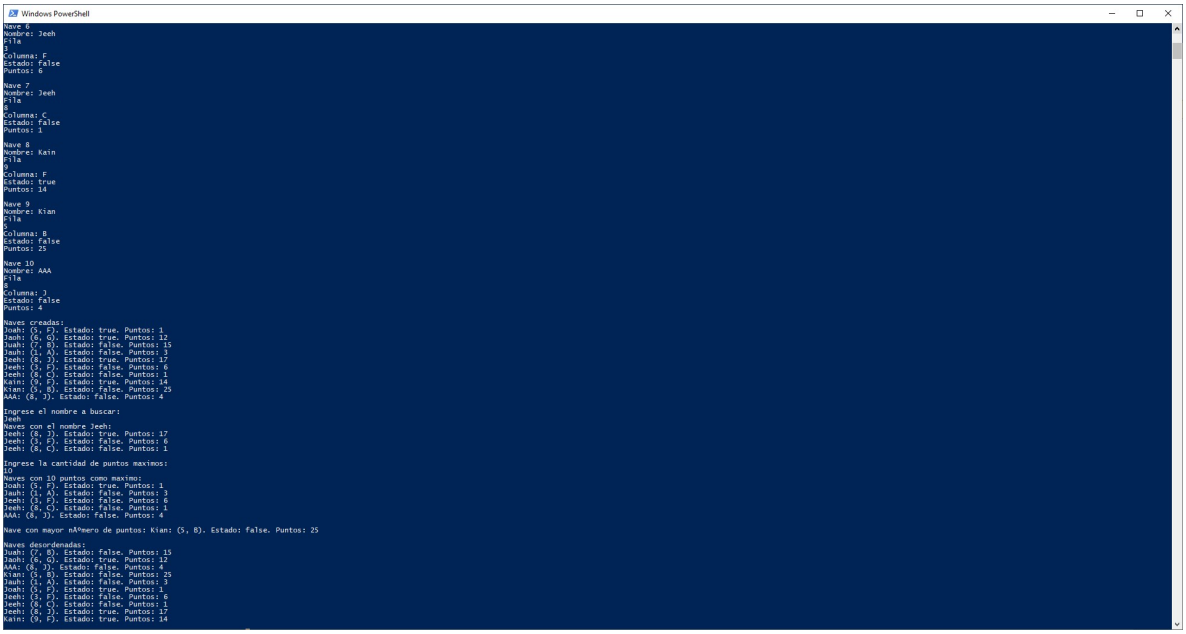
- Los 5 nombres y sus respectivas vidas son leídas, para luego instanciar objetos de la clase soldado con sus respectivos atributos.
- Luego los atributos son impresos en el orden en el que se ingresaron.

Actividad3.java

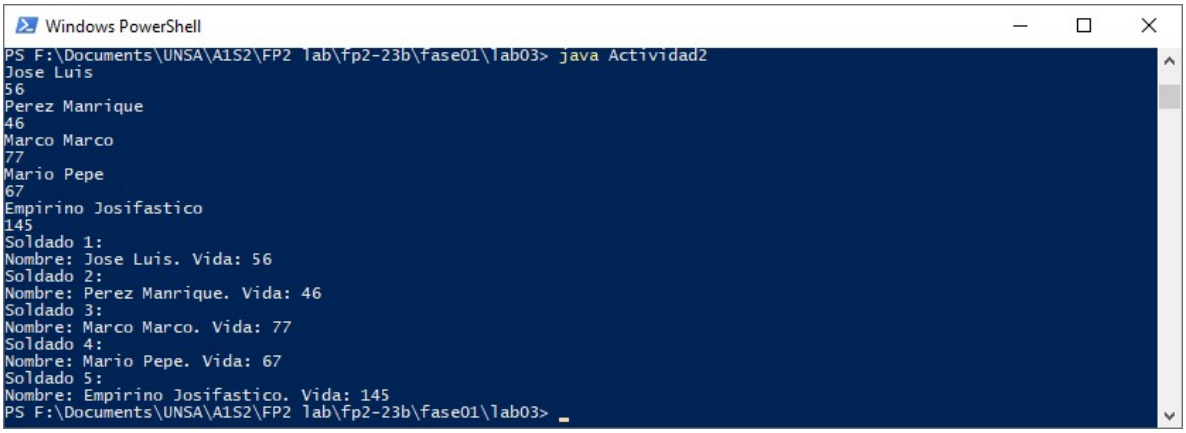
```java
// Laboratorio Nro 3 - Actividad 3
// Autor: Christian Mestas

import java.util.Random;

public class Actividad3 {
    public static void main(String[] args) {
        Soldado[] ejercito1 = inicializarEjercito();
        Soldado[] ejercito2 = inicializarEjercito();
        mostrarEjercito(ejercito1, 1);
        mostrarEjercito(ejercito2, 2);
        mostrarGanador(ejercito1, ejercito2);
    }

    public static Soldado[] inicializarEjercito() {
        Random random = new Random();
        Soldado[] ejercito = new Soldado[random.nextInt(5) + 1];
        for (int i = 0; i < ejercito.length; i++)
            ejercito[i] = new Soldado("Soldado" + i, 0);
        return ejercito;
    }

    public static void mostrarEjercito(Soldado[] ejercito, int numero) {
        System.out.println("Ejercito " + numero + ":");
        for (int i = 0; i < ejercito.length; i++)
            System.out.println("Soldado " + (i + 1) + ": " + ejercito[i].getNombre());
    }

    public static void mostrarGanador(Soldado[] ejercito1, Soldado[] ejercito2) {
        if (ejercito1.length > ejercito2.length)
            System.out.println("Gana el ejercito 1!");
        else if (ejercito1.length < ejercito2.length)
            System.out.println("Gana el ejercito 2!");
        else
            System.out.println("Hubo un empate!");
    }
}
```

- Se crean 2 ejércitos con el método inicializarEjercito(), que crea un arreglo de soldados de tamaño al azar entre 1 y 5.
- mostrarEjercito() 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.

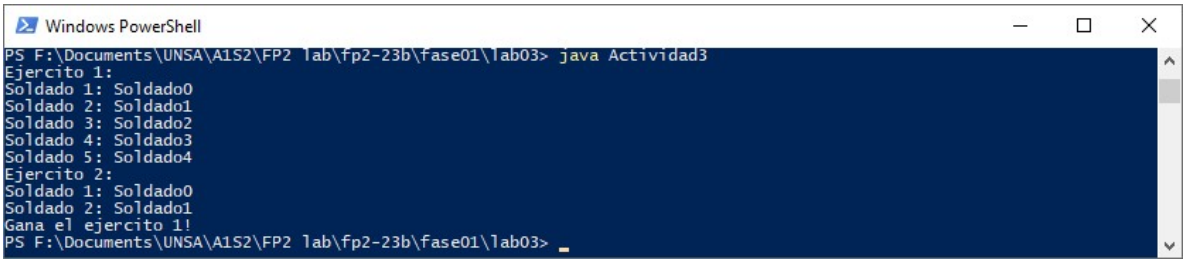
6. Ejecución del código



DemoBatalla.java



Actividad2.java



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
   |--- 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	2	
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		0	
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	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	4	
Total		20		16.5	

9. Referencias

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