INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	*FUNDAMENTOS DE PROGRAMACIÓN 2*				
TÍTULO DE LA PRÁCTICA:	*Arreglos de Objjetos*				
NÚMERO DE PRÁCTICA:	*3*	**AÑO LECTIVO:**	*2024*	**NRO. SEMESTRE:**	*II*
FECHA DE PRESENTACIÓN	*05/10/2024*	**HORA DE PRESENTACIÓN**	*19:40:59*		
INTEGRANTE (s) *Riveros Vilca Alberth Edwar*				**NOTA (0-20)**	
DOCENTE(s): *Ing. Lino Jose Pinto Oppe*					

RESULTADOS Y PRUEBAS

I. EJERCICIOS RESUELTOS:

Actividad 1: *Analice, complete y pruebe el Código de la clase DemoBatalla.*

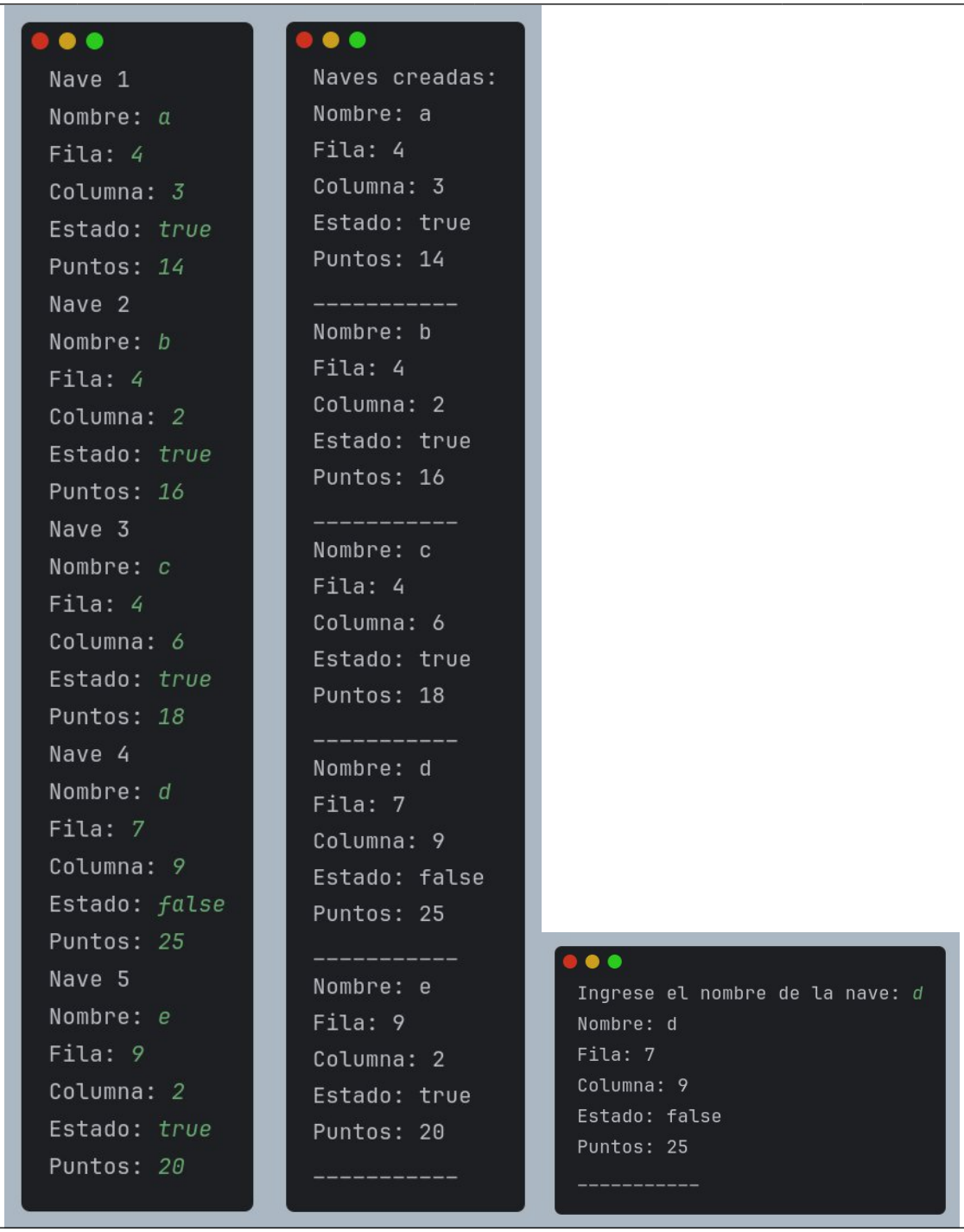
CLASE Nave:

```java
public class Nave {  15 usages  rivX241
    private String nombre;  3 usages
    private int fila;  3 usages
    private String columna;  3 usages
    private boolean estado;  3 usages
    private int puntos;  3 usages
    // Metodos mutadores
    public void setNombre( String n){  1 usage  rivX241
        nombre = n;
    }
    public void setFila(int f){  1 usage  rivX241
        fila = f;
    }
    public void setColumna(String c){  1 usage  rivX241
        columna = c;
    }
    public void setEstado(boolean e){  1 usage  rivX241
        estado = e;
    }
    public void setPuntos(int p){  1 usage  rivX241
        puntos = p;
    }
    // Metodos accesores
    public String getNombre(){  1 usage  rivX241
        return nombre;
    }
    public int getFila(){  no usages  rivX241
        return fila;
    }
    public String getColumna(){  no usages  rivX241
        return columna;
    }
    public boolean getEstado(){  1 usage  rivX241
        return estado;
    }
    public int getPuntos(){  3 usages  rivX241
        return puntos;
    }
    @Override  rivX241
    public String toString(){
        return "Nombre: "+nombre+"\nFila: "+fila+"\nColumna: "+columna+"\nEstado: "+estado+"\nPuntos: "+puntos
                +"\n__________";
    }
}
```

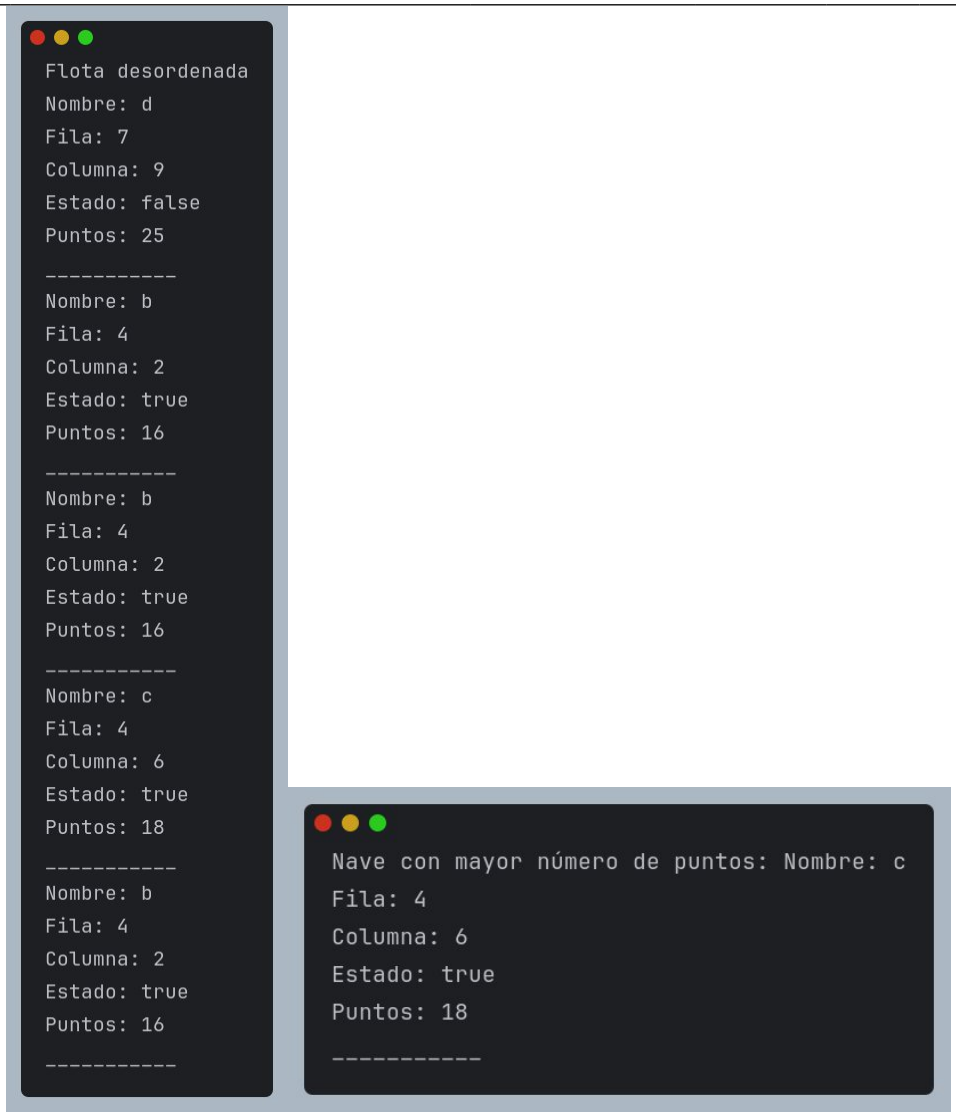
```java
import java.util.*;
public class DemoBatalla {  rivX241 *
    public static void main(String [] args){  rivX241 *
        Nave [] misNaves = new Nave[5];
        Scanner sc = new Scanner(System.in);
        String nomb, col;
        int fil, punt;
        Random rand = new Random();
        boolean est;
        for (int i = 0; i < misNaves.length; i++) {
            System.out.println("Nave " + (i+1));
            System.out.print("Nombre: ");
            nomb = sc.next();
            System.out.print("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("\nNaves creadas:");
        mostrarNaves(misNaves);
        mostrarPorNombre(misNaves);
        mostrarPorPuntos(misNaves);
        System.out.println("Flota desordenada");
        Nave[] flota2 = nuevaFlota(misNaves);
        mostrarNaves(flota2);
        System.out.println("\nNave con mayor número de puntos: " + mostrarMayorPuntos(misNaves));
    }
    public static void mostrarNaves(Nave [] flota){  2 usages  rivX241
        for (Nave nave : flota) {
            System.out.println(nave.toString());
        }
    }
```

```java
public static void mostrarPorNombre(Nave [] flota){  1 usage  rivX241
    Scanner sc = new Scanner(System.in);
    System.out.print("Ingrese el nombre de la nave: ");
    String name = sc.nextLine();
    for(Nave naveBuscada : flota){
        if(naveBuscada.getNombre().equals(name)){
            System.out.println(naveBuscada);
        }
    }
}
public static void mostrarPorPuntos(Nave [] flota){  1 usage  rivX241
    Scanner sc = new Scanner(System.in);
    System.out.print("Ingrese el número de puntos para buscar las naves inferiores a este: ");
    int puntos = sc.nextInt();
    for(Nave naveFiltrada : flota){
        if(naveFiltrada.getPuntos()<=puntos){
            System.out.println(naveFiltrada);
        }
    }
}
public static String mostrarMayorPuntos(Nave [] flota){  1 usage  rivX241
    int index = 0;
    for(int i=1;i< flota.length;i++){
        if(flota[i].getPuntos()>flota[i-1].getPuntos()&&flota[i].getEstado()){
            index = i;
        }
    }
    return flota[index].toString();
}
public static Nave[] nuevaFlota(Nave[] flota){  1 usage  rivX241
    Nave[] nuevaFlota = new Nave[flota.length];
    Random rand = new Random();
    int randomIndex;
    for(int i = 0;i< flota.length;i++){
        randomIndex = rand.nextInt(flota.length);
        nuevaFlota[i] = flota[randomIndex];
    }
    return nuevaFlota;
}
}
```

EJECUCIÓN:


Nave 1
Nombre: a
Fila: 4
Columna: 3
Estado: true
Puntos: 14
Nave 2
Nombre: b
Fila: 4
Columna: 2
Estado: true
Puntos: 16
Nave 3
Nombre: c
Fila: 4
Columna: 6
Estado: true
Puntos: 18
Nave 4
Nombre: d
Fila: 7
Columna: 9
Estado: false
Puntos: 25
Nave 5
Nombre: e
Fila: 9
Columna: 2
Estado: true
Puntos: 20
Naves creadas:
Nombre: a
Fila: 4
Columna: 3
Estado: true
Puntos: 14
Nombre: b
Fila: 4
Columna: 2
Estado: true
Puntos: 16
Nombre: c
Fila: 4
Columna: 6
Estado: true
Puntos: 18
Nombre: d
Fila: 7
Columna: 9
Estado: false
Puntos: 25
Nombre: e
Fila: 9
Columna: 2
Estado: true
Puntos: 20
Ingrese el nombre de la nave: d
Nombre: d
Fila: 7
Columna: 9
Estado: false
Puntos: 25

```
Ingrese el número de puntos para buscar las naves inferiores a este: 20
Nombre: a
Fila: 4
Columna: 3
Estado: true
Puntos: 14
-----------
Nombre: b
Fila: 4
Columna: 2
Estado: true
Puntos: 16
-----------
Nombre: c
Fila: 4
Columna: 6
Estado: true
Puntos: 18
-----------
Nombre: e
Fila: 9
Columna: 2
Estado: true
Puntos: 20
-----------
```


Flota desordenada
Nombre: d
Fila: 7
Columna: 9
Estado: false
Puntos: 25
Nombre: b
Fila: 4
Columna: 2
Estado: true
Puntos: 16
Nombre: b
Fila: 4
Columna: 2
Estado: true
Puntos: 16
Nombre: c
Fila: 4
Columna: 6
Estado: true
Puntos: 18
Nombre: b
Fila: 4
Columna: 2
Estado: true
Puntos: 16
Nave con mayor número de puntos: Nombre: c
Fila: 4
Columna: 6
Estado: true
Puntos: 18

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

```java
public class Soldado {  13 usages  new *
    private String nombre;  5 usages
    private int vida=0;  4 usages
    public Soldado(String nombre) {  1 usage  new *
        this.nombre = nombre;
    }
    public Soldado(){  1 usage  new *
        this.nombre = "";
        this.vida = 0;
    }
    public String getNombre() {  1 usage  new *
        return nombre;
    }
    public void setNombre(String nombre) {  1 usage  new *
        this.nombre = nombre;
    }
    public int getVida() {  no usages  new *
        return vida;
    }
    public void setVida(int vida) {  1 usage  new *
        this.vida = vida;
    }
    @Override  new *
    public String toString() {
        return "Soldado{" + "nombre=" + nombre + ", vida=" + vida + '}';
    }
}
```

```java
import java.util.*;
public class Actividad04L1v2 {  new *
    public static void main(String[] args){  new *
        Random rand = new Random();
        Scanner sc = new Scanner(System.in);
        Soldado[] nameSoldiers = new Soldado[5];
        for(int i = 0;i < nameSoldiers.length;i++){
            nameSoldiers[i] = new Soldado();
            System.out.printf("Ingrese el nombre del soldado nro %d:",i+1);
            nameSoldiers[i].setNombre(sc.next());
            nameSoldiers[i].setVida(rand.nextInt( bound: 5)+1);
        }
        System.out.println("Informacion de los soldados: ");
        for (Soldado nameSoldier : nameSoldiers) {
            System.out.printf(nameSoldier.toString() + "\n");
        }
    }
}
```

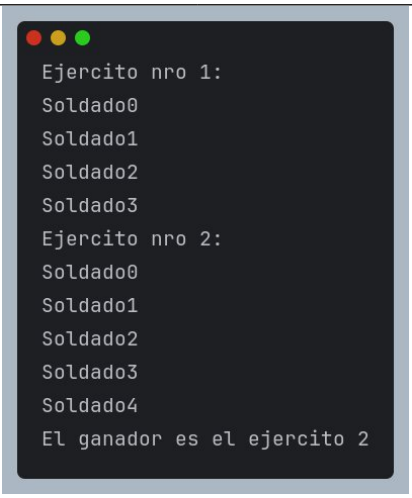
EJECUCIÓN:

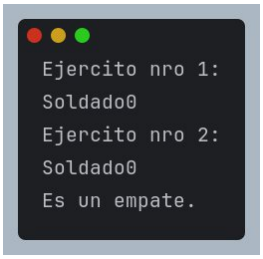
```
Ingrese el nombre del soldado nro 1:Juan
Ingrese el nombre del soldado nro 2:Pedro
Ingrese el nombre del soldado nro 3:Ricardo
Ingrese el nombre del soldado nro 4:Alex
Ingrese el nombre del soldado nro 5:Sandro
Informacion de los soldados:
Soldado{nombre=Juan, vida=4}
Soldado{nombre=Pedro, vida=1}
Soldado{nombre=Ricardo, vida=4}
Soldado{nombre=Alex, vida=5}
Soldado{nombre=Sandro, vida=3}
```

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

```java
import java.util.*;
public class Actividad05L1v2 {  new *
    public static void main(String[] args){  new *
        Random rand = new Random();
        int numSoldEjercito1 = rand.nextInt( bound: 5)+1,numSoldEjercito2 = rand.nextInt( bound: 5)+1;
        Soldado[] ejercito1 = new Soldado[numSoldEjercito1],ejercito2 = new Soldado[numSoldEjercito2];
        inicializarEjercito(ejercito1);
        inicializarEjercito(ejercito2);
        System.out.println("Ejercito nro 1:");
        mostrarEjercito(ejercito1);
        System.out.println("Ejercito nro 2:");
        mostrarEjercito(ejercito2);
        System.out.println(mostrarGanador(ejercito1,ejercito2));
    }
    public static void inicializarEjercito(Soldado[] ejercito){  2 usages  new *
        for(int i = 0;i < ejercito.length; i++){
            ejercito[i] = new Soldado( nombre: "Soldado" + i);
        }
    }
    public static void mostrarEjercito(Soldado[] ejercito){  2 usages  new *
        for(Soldado soldado : ejercito){
            System.out.print(soldado.getNombre()+"\n");
        }
    }
    public static String mostrarGanador(Soldado[] ejercito1,Soldado[] ejercito2){  1 usage  new *
        if(ejercito1.length==ejercito2.length){
            return "Es un empate.";
        }else{
            return ejercito1.length>ejercito2.length?"El ganador es el ejercito 1":"El ganador es el ejercito 2";
        }
    }
}
```

EJECUCIÓN:





II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?

De acuerdo al problema con valores de tipo String que equivalen a cadenas de caracteres,datos de tipo entero y booleanos.

¿Qué resultado esperabas obtener para cada valor de entrada?

Que se guardara en cada objeto el String para el nombre y el valor entero que era aleatorio al de vida,la generación del nuevo arreglo no tuvo mayores inconvenientes.

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

Los valores almacenados previamente se imprimieron por pantalla gracias a los métodos getters y setters desarrollados.

III. CUESTIONARIO:

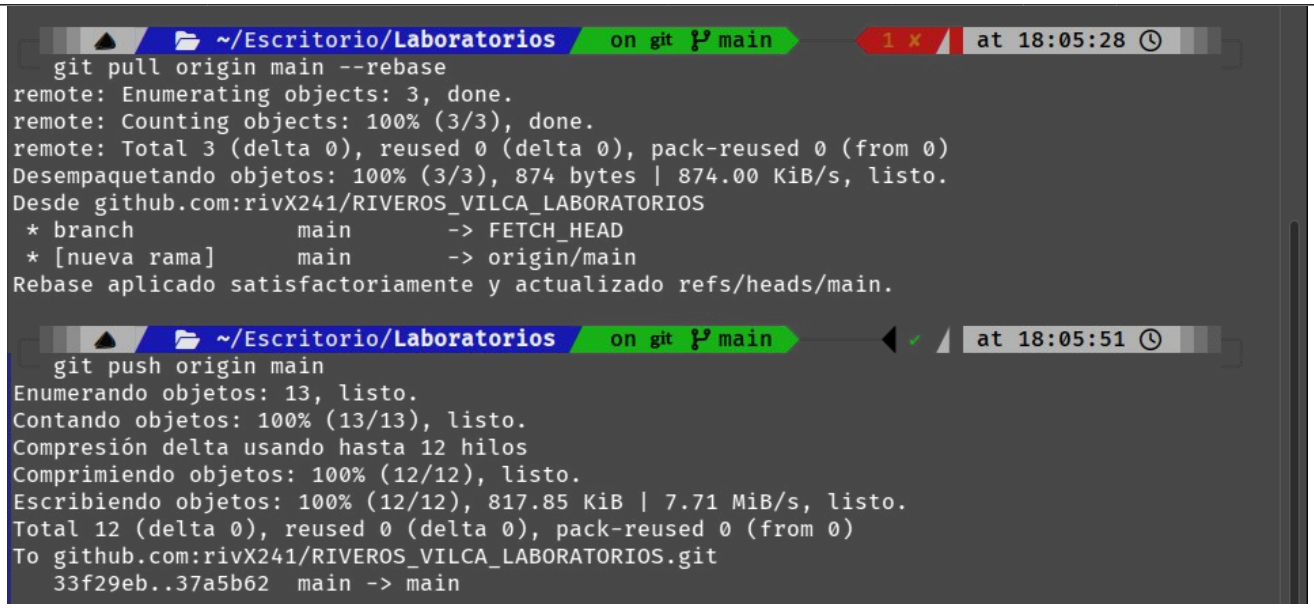
```
~/Escritorio/Laboratorios                                    at 17:59:43
  git init
hint: Usando 'master' como el nombre de la rama inicial. Este nombre de rama predeterminado
hint: está sujeto a cambios. Para configurar el nombre de la rama inicial para usar en todos
hint: de sus nuevos repositorios, reprimiendo esta advertencia, llama a:
hint:
hint:   git config --global init.defaultBranch <nombre>
hint:
hint: Los nombres comúnmente elegidos en lugar de 'master' son 'main', 'trunk' y
hint: 'development'. Se puede cambiar el nombre de la rama recién creada mediante este coman
do:
hint:
hint:   git branch -m <nombre>
Inicializado repositorio Git vacío en /home/riv/Escritorio/Laboratorios/.git/
```

Inicie el repositorio local.

```
~/Escritorio/Laboratorios  on git  master ?2                 at 17:59:54
  git branch -m main

~/Escritorio/Laboratorios  on git  main ?2                   at 18:03:01
  git add .

~/Escritorio/Laboratorios  on git  main +8                   at 18:03:29
  git remote add origin https://github.com/rivX241/RIVEROS_VILCA_LABORATORIOS.git

~/Escritorio/Laboratorios  on git  main +8                   at 18:04:41
  git commit -m "lab3-finalizado"
[main (commit-raíz) 1da6664] lab3-finalizado
 8 files changed, 345 insertions(+)
 create mode 100644 RIVEROS_VILCA_LABORATORIO_02/JuegoAhorcado.java
 create mode 100644 RIVEROS_VILCA_LABORATORIO_02/LINK_MI_REPOSITORIO.txt
 create mode 100644 RIVEROS_VILCA_LABORATORIO_02/RIVEROS_VILCA_LABORATORIO_02.pdf
 create mode 100644 RIVEROS_VILCA_LABORATORIO_03/Actividad04L1v2.java
 create mode 100644 RIVEROS_VILCA_LABORATORIO_03/Actividad05L1v2.java
 create mode 100755 RIVEROS_VILCA_LABORATORIO_03/DemoBatalla.java
 create mode 100755 RIVEROS_VILCA_LABORATORIO_03/Nave.java
 create mode 100644 RIVEROS_VILCA_LABORATORIO_03/Soldado.java
```

Cambie a la rama main debido a que al crear el repositorio en Github es la rama por defecto al crearlo, Añadí los archivos a stage,añadí el remoto con el link del repositorio de Github, y realice el commit con el mensaje lab3-finalizado.

LINK:https://github.com/rivX241/RIVEROS_VILCA_LABORATORIOS



Debido a que al crear el repositorio añadí un archivo readme tuve que hacer pull con —rebase para evitar problemas al realizar el push,luego de ello hice el push y no obtuve errores.

CONCLUSIONES

Los arreglos es una estructura de datos muy vérsatil que nos permite manejar grandes cantidades de datos de manera más sencilla que hacerlo individualmente,además junto al uso de los objetos nos permite emplear los métodos getters y setters para así no depender de varios arreglos gracias a los atributos de las clases que empleamos.

METODOLOGÍA DE TRABAJO

1.Realice la lectura del problema vi los requisitos y restricciones del mismo para plantear una solución.
2.Identifique las herramientas y la lógica requerida para la solución del mismo.
3.Codifique el problema para luego testearlo con los datos de entrada.
4.Analice y reestructure el código del laboratorio 01 de acuerdo al uso de clases e implemente la clase necesaria para los arreglos con clases.
4.Probé la solución y corregí los errores que estuvieron presentes.

REFERENCIAS Y BIBLIOGRAFÍA

Colocare las referencias utilizadas para el desarrollo de la práctica en formato IEEE

E. G. Castro Gutiérrez y M. W. Aedo López, *Fundamentos de programación 2: tópicos de programación orientada a objetos*, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021. ISBN: 978-612-5035-20-2. 170 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 20.5 x 29 cm.

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	4	
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 evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	2	
TOTAL		20		18	