

	<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

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de programación II-GRUPO F				
TÍTULO DE LA PRÁCTICA:	Arreglos de Objetos				
NÚMERO DE PRÁCTICA:	3	AÑO LECTIVO:	2024	NRO. SEMESTRE:	2
FECHA DE PRESENTACIÓN	4/10/2024	HORA DE PRESENTACIÓN	23/40/00		
INTEGRANTE (s) Usiel Surriel Quispe Puma				NOTA (0-20)	Nota colocada por el docente
DOCENTE(s): Pinto Oppe Lino José					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <p>EJERCICIOS RESUELTOS:</p> <p>EJERCICIO 1</p> <p>Analice, complete y pruebe el Código de la clase Demo Batalla</p> <p>CÓDIGO:</p> <ul style="list-style-type: none"> - Clase Principal(DemoBatalla):

```
package Laboratorio_03;
//Laboratorio N° 3- Ejercicio 1
//Autor: Usiel Surriel Quispe Puma

import java.util.*;

public class DemoBatalla {

    public static void main(String[] args) {
        Nave[] misNaves = new Nave[10];
        Nave[] navesRandom = new Nave[10];
        Scanner sc = new Scanner(System.in);
        String nomb, col;
        int fil, punt = 0;
        boolean est;

        for (int i = 0; i < misNaves.length; i++) {
            System.out.println("\nNave " + (i + 1));
            System.out.print("Nombre: ");
            nomb = sc.next();
            System.out.print("Fila (1-10): ");
            fil = sc.nextInt();
            System.out.print("Columna (A-J): ");
            col = sc.next().toUpperCase();
            System.out.print("Estado(true - false) : ");
            est = sc.nextBoolean();
            System.out.print("Puntos: ");
            punt = sc.nextInt();
            //creamos objeto nave e ingresamos los valores
            misNaves[i] = new Nave();

            misNaves[i].setNombre(nomb);
            misNaves[i].setFila(fil);
            misNaves[i].setColumna(col);
            misNaves[i].setEstado(est);
            misNaves[i].setPuntos(punt);
        }

        System.out.println("\n\tNAVES CREADAS");
        mostrarNaves(misNaves);
        System.out.println("-----\n");

        System.out.println("\tNAVES CON EL MISMO NOMBRE");
        System.out.print("Ingrese el nombre : ");
        nomb = sc.next();
        mostrarPorNombre(misNaves, nomb); //se agrego el argumento nombre
        System.out.println("-----\n");

        //mostramos los puntos de cada nave
        System.out.println("\tNAVES CON PUNTOS MENOR A PUNTOS INGRESADOS");
        mostrarPorPuntos(misNaves, punt);
    }
}
```

```
System.out.print("Ingrese la cantidad de puntos de referencia : ");
punt = sc.nextInt();
mostrarPorPuntos(misNaves, punt);

System.out.println("-----\n");
System.out.println("\tNAVE(S) CON MAYOR PUNTAJE");
mostrarMayorPuntos(misNaves);

//hacemos que se muestre los naves pero en orden desordenado por el cual
System.out.println("-----\n");
System.out.println("\tORDEN DE NAVES MODIFICADO");
navesRandom = desordenarNaves(misNaves); //Nos devuelve otro arreglo, lo asignamos
//a otro arreglo
mostrarNaves(navesRandom); //Mostramos el nuevo arreglo
}

//Método para mostrar todas las naves

public static void mostrarNaves(Nave[] flota) { //listo
    for (int i = 0; i < flota.length; i++) {
        System.out.println("NAVE " + (i + 1));
        flota[i].mostrarDatos();
        System.out.println("");
    }
}

//Método para mostrar todas las naves de un nombre que se pide por teclado
public static void mostrarPorNombre(Nave[] flota, String nombre) {
    for (int i = 0; i < flota.length; i++) {
        if (flota[i] != null && flota[i].getNombre().equals(nombre)) {

            flota[i].mostrarDatos();
            System.out.println("");
        }
    }
}

//Método para mostrar todas las naves con un número de puntos inferior o igual
//al número de puntos que se pide por teclado
public static void mostrarPorPuntos(Nave[] flota, int puntos) {
    for (int i = 0; i < flota.length; i++) {
        if (flota[i].getPuntos() <= puntos) {
            flota[i].mostrarDatos();
            System.out.println("");
        }
    }
}
```

```
}

//Método que devuelve la Nave con mayor número de Puntos
public static void mostrarMayorPuntos(Nave[] flota) {
    int mayorPuntos = 0;
    int nave = 0;
    for (int i = 0; i < flota.length; i++) {
        if (flota[i].getPuntos() > mayorPuntos) {
            mayorPuntos = flota[i].getPuntos();
            nave = i;
        }
    }
    flota[nave].mostrarDatos();
}

//Crear un método que devuelva un nuevo arreglo de objetos con todos los objetos
//previamente ingresados pero aleatoriamente desordenados
public static Nave[] desordenarNaves(Nave[] flota) {
    Random random = new Random();
    Nave[] flotaRandom = new Nave[flota.length];

    //copiamos lo elementos del arreglo orgininal a otro arreglo
    for (int i = 0; i < flota.length; i++) {
        flotaRandom[i] = flota[i];
    }

    //Aplicamos el algoritmo de Fisher-Yates para desordenar el arreglo
    for (int i = flotaRandom.length - 1; i > 0; i--) {
        int posAleatoria = random.nextInt(i + 1);
        Nave temp = flotaRandom[i];
        flotaRandom[i] = flotaRandom[posAleatoria];
        flotaRandom[posAleatoria] = temp;
    }

    return flotaRandom;
}
}
```

- Clase Nave:

```
package Laboratorio_03;

//Laboratorio N° 3- Ejercicio 1
//Autor: Uziel Surriel Quispe Puma
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 accesorios

    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;
    }

    public void mostrarDatos() {
        System.out.println("Nombre : " + nombre);
        System.out.println("fila: " + fila + "      Columna : " + columna);
        System.out.println("puntos : " + puntos);
        System.out.println("Estado : " + estado);
    }
}
```

PRUEBAS: (solo se uso 4 objetos nave para el ejemplo)

```
Nave 1
Nombre: F1
Fila (1-10): 4
Columna (A-J): j
Estado(true - false) : true
Puntos: 10
```

```
Nave 2
Nombre: F2
Fila (1-10): 2
Columna (A-J): a
Estado(true - false) : true
Puntos: 30
```

```
Nave 3
Nombre: F1
Fila (1-10): 7
Columna (A-J): g
Estado(true - false) : true
Puntos: 9
```

```
Nave 4
Nombre: F4
Fila (1-10): 5
Columna (A-J): c
Estado(true - false) : true
Puntos: 50
```

NAVES CREADAS

```
NAVE 1
Nombre : F1
fila: 4      Columna : J
puntos : 10
Estado : true
```

```
NAVE 2
Nombre : F2
fila: 2      Columna : A
puntos : 30
Estado : true
```

```
NAVE 3
Nombre : F1
fila: 7      Columna : G
puntos : 9
Estado : true

NAVE 4
Nombre : F4
fila: 5      Columna : C
puntos : 50
Estado : true

-----

      NAVES CON EL MISMO NOMBRE
Ingrese el nombre : F1
Nombre : F1
fila: 4      Columna : J
puntos : 10
Estado : true

Nombre : F1
fila: 7      Columna : G
puntos : 9
Estado : true

-----

      NAVES CON PUNTOS MENOR A PUNTOS INGRESADOS
Ingrese la cantidad de puntos de referencia : 40
Nombre : F1
fila: 4      Columna : J
puntos : 10
Estado : true

Nombre : F2
fila: 2      Columna : A
puntos : 30
Estado : true

Nombre : F1
fila: 7      Columna : G
puntos : 9
Estado : true
```

```
-----  
  
      NAVE(S)  CON MAYOR PUNTAJE  
Nombre : F4  
fila: 5      Columna : C  
puntos : 50  
Estado : true  
-----
```

ORDEN DE NAVES MODIFICADO

```
NAVE 1  
Nombre : F1  
fila: 7      Columna : G  
puntos : 9  
Estado : true
```

```
NAVE 2  
Nombre : F2  
fila: 2      Columna : A  
puntos : 30  
Estado : true
```

```
NAVE 3  
Nombre : F4  
fila: 5      Columna : C  
puntos : 50  
Estado : true
```

```
NAVE 4  
Nombre : F1  
fila: 4      Columna : J  
puntos : 10  
Estado : true
```


EJERCICIO 2

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

CÓDIGO:

```
package Laboratorio_03;
//Laboratorio N° 3- Ejercicio 2
//Autor: Usiel Surriel Quispe Puma

import java.util.*;

public class EJERCICIO_2 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        String nombre;
        int nivelVida;

        //Arreglo de objeto de la clase soldados
        Soldado[] soldados = new Soldado[5];
        ingresarDatos(soldados, sc);
        System.out.println("=====");
        System.out.println("\n\tSOLDADOS INGRESADOS\n");
        mostrarDatos(soldados);
    }

    //metodo para llenar datos del arreglo
    public static void ingresarDatos(Soldado[] soldados, Scanner sc) {
        String nombre;
        int nivelVida;

        for (int i = 0; i < soldados.length; i++) {
            System.out.println("\n\tSOLDADO " + (i+1));
            System.out.print("Ingrese el nombre : ");
            nombre = sc.next();
            System.out.print("Ingrese su nivel de vida : ");
            nivelVida = sc.nextInt();
            //Ingresmos los datos al objeto
            soldados[i] = new Soldado(nombre, nivelVida);
        }
    }

    //Metodo para mostrar los datos de cada soldado
    public static void mostrarDatos(Soldado [] soldados){
        for(Soldado a : soldados){
            System.out.println( a.toString());
            System.out.println("-----\n");
        }
    }
}
```

- Clase Soldado:

```
package Laboratorio_03;

public class Soldado {

    // Atributos de la clase Soldado
    private String nombre;
    private int nivelVida;

    // Constructor
    public Soldado(String nombre, int nivelVida) {
        this.nombre = nombre;
        this.nivelVida = nivelVida;
    }

    // Métodos getter y setter
    public String getNombre() {
        return nombre;
    }

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

    public int getNivelVida() {
        return nivelVida;
    }

    public void setNivelVida(int nivelVida) {
        this.nivelVida = nivelVida;
    }

    // Método toString para mostrar los detalles del soldado
    @Override
    public String toString() {
        return "Nombre='" + nombre + "\tNivel de vida=" + nivelVida;
    }
}
```

PRUEBAS:**SOLDADO 1**

Ingrese el nombre : Max

Ingrese su nivel de vida : 10

SOLDADO 2

Ingrese el nombre : Juan

Ingrese su nivel de vida : 30

SOLDADO 3

Ingrese el nombre : Jhon

Ingrese su nivel de vida : 2

SOLDADO 4

Ingrese el nombre : Maxin

Ingrese su nivel de vida : 23

SOLDADO 5

Ingrese el nombre : David

Ingrese su nivel de vida : 12

=====

SOLDADOS INGRESADOS

Nombre : Max Nivel de Vida : 10

Nombre : Juan Nivel de Vida : 30

Nombre : Jhon Nivel de Vida : 2

Nombre : Maxin Nivel de Vida : 23

Nombre : David Nivel de Vida : 12

EJERCICIO 3

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

CÓDIGO:

- Clase Principal:

```
package Laboratorio_03;
//Laboratorio N° 3- Ejercicio 3
//Autor: Usiel Surriel Quispe Puma

import java.util.*;

public class EJERCICIO_3 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Random rdn = new Random();
        int cantSoldados1, cantSoldados2;

        //De manera aleatoria generamos la cantidad de soldados
        cantSoldados1 = rdn.nextInt(5) + 1;
        cantSoldados2 = rdn.nextInt(5) + 1;

        //Creamos objeto de la clase Ejercito
        Ejercito e1 = new Ejercito(cantSoldados1);
        Ejercito e2 = new Ejercito(cantSoldados2);

        //Inicializamos los datos de cada soldado
        e1.generarNombres();
        e2.generarNombres();

        //Mostramos datos de todos los soldado de cada ejercito
        System.out.println("\nDATOS DE LOS SOLDADOS DEL EJERCITO 1\n");
        e1.mostrarDatos();
        System.out.println("\nDATOS DE LOS SOLDADOS DEL EJERCITO 2\n");
        e2.mostrarDatos();

        //mostramos al ganador
        System.out.println("\n-----");
        mostrarGanador(cantSoldados1, cantSoldados2);
    }

    //Método para designar al ganador considerando la cantidad de soldados
    //ejercito
    public static void mostrarGanador(int cant1, int cant2) {
        if (cant1 > cant2) {
            System.out.println("GANADOR : EJERCITO 1    \nCantidad de soldados : "+cant1);
        } else if (cant2 > cant1) {
            System.out.println("GANADOR : EJERCITO 2    \nCantidad de soldados : " + cant2);
        } else {
            System.out.println("EMPATE !!");
            System.out.println("EJERCITO 1 : "+cant1+"    EJERCITO 2 : "+cant2);
        }
    }
}
```

- Clase Ejercito:

```
package Laboratorio_03;
//Laboratorio N° 3- Ejercicio 3
//Autor: Uziel Surriel Quispe Puma



public class Ejercito {

    //Arreglo de objeto de la clase Soldado_2
    Soldado_2[] soldados;

    //Creamos un arreglo de objetos, e indicamos la cantidad de soldados que se generara por cada ejercito
    public Ejercito(int cantidad) {
        soldados = new Soldado_2[cantidad];
    }

    //metodo para generar nombres de los soldados
    public void generarNombres() {
        int id;
        for (int i = 0; i < soldados.length; i++) {
            id=(int)(Math.random()*100);
            String nombre;
            nombre = "Soldado" + id;
            soldados[i] = new Soldado_2(nombre);
        }
    }

    //Metodo para mostrar los datos de todos los soldados de cada ejercito
    public void mostrarDatos() {
        for (Soldado_2 e : soldados) {
            System.out.println(e.toString());
            System.out.println("\n");
        }
    }
}
```

	<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: 14</p>

- Clase Soldado:

```
package Laboratorio_03;
//Laboratorio N° 3- Ejercicio 3
//Autor: Usiel Surriel Quispe Puma

public class Soldado_2 {

    private String nombre;

    public Soldado_2(String nombre) {
        this.nombre = nombre;
    }

    //Metodo to String para mostrar datos de los soldados

    @Override
    public String toString() {
        return "Nombre : " + nombre;
    }

}
```

PRUEBAS:

Caso: Ejercito 1 gana

```
DATOS DE LOS SOLDADOS DEL EJERCITO 1

Nombre : Soldado34

Nombre : Soldado58

Nombre : Soldado71

Nombre : Soldado57

DATOS DE LOS SOLDADOS DEL EJERCITO 2

Nombre : Soldado22

-----
GANADOR : EJERCITO 1
Cantidad de soldados : 4
```

Caso : Empate

```

DATOS DE LOS SOLDADOS DEL EJERCITO 1
Nombre : Soldado81

Nombre : Soldado63

Nombre : Soldado8

Nombre : Soldado57

Nombre : Soldado66

DATOS DE LOS SOLDADOS DEL EJERCITO 2
Nombre : Soldado72

Nombre : Soldado31

Nombre : Soldado76

Nombre : Soldado59

Nombre : Soldado0

-----
EMPATE !!
EJERCITO 1 : 5    EJERCITO 2 : 5

```

Caso: Ejercito 2 gana

```

DATOS DE LOS SOLDADOS DEL EJERCITO 1
Nombre : Soldado70

DATOS DE LOS SOLDADOS DEL EJERCITO 2
Nombre : Soldado46

Nombre : Soldado35

Nombre : Soldado42

-----
GANADOR : EJERCITO 2
Cantidad de soldados : 3

```

	<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: 16

II. PRUEBAS

- Capturas del commit:

```
PS C:\Users\USIEL\OneDrive\Documentos\TEORIA_LABORATORIOS_PW_FP\LABORATORIOS\Lab_FP2> git add src/Laboratorio_03
PS C:\Users\USIEL\OneDrive\Documentos\TEORIA_LABORATORIOS_PW_FP\LABORATORIOS\Lab_FP2> git commit -m "Se agrego el labora
torio 3"
[master 7c11a34] Se agrego el laboratorio 3
PS C:\Users\USIEL\OneDrive\Documentos\TEORIA_LABORATORIOS_PW_FP\LABORATORIOS\Lab_FP2> git push origin master
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (14/14), 2.18 KiB | 1.09 MiB/s, done.
Total 14 (delta 9), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (9/9), completed with 7 local objects.
To https://github.com/usiel33/Laboratorios_FP2.git
 6b610d9..7c11a34 master -> master
PS C:\Users\USIEL\OneDrive\Documentos\TEORIA_LABORATORIOS_PW_FP\LABORATORIOS\Lab_FP2> git log
commit 7c11a340d42ee477a7a33dc62c201c0ef745acec (HEAD -> master, origin/master)
Merge: e8e11c0 6b610d9
Author: usiel33 <uquissep@unsa.edu.pe>
Date: Fri Oct 4 23:22:50 2024 -0500

Se agrego el laboratorio 3
```

III. CUESTIONARIO:

CONCLUSIONES

El uso de arreglos de objetos mejora la eficiencia del programa haciendo que la cantidad de código necesaria sea mínima.

METODOLOGÍA DE TRABAJO

La metodología que use para realizar este laboratorio fue , ver los ejercicios y sus restricciones después identificar las posibles soluciones para que el programa funcione.

REFERENCIAS Y BIBLIOGRAFÍA

	<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: 17</p>

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